# GNU/Linux

## THE MAN PAGES BOOK


**Maintainers:**

**Alejandro Colomar <alx@kernel.org> 2020 - present (5.09 - HEAD)**
**Michael Kerrisk <mtk.manpages@gmail.com> 2004 - 2021 (2.00 - 5.13)**
**Andries Brouwer <aeb@cwi.nl> 1995 - 2004 (1.6 - 1.70)**
**Rik Faith 1993 - 1995 (1.0 - 1.5)**

## NAME

perl5004delta − what's new for perl5.004

## DESCRIPTION

This document describes differences between the 5.003 release (as documented in *Programming Perl*, second edition — the Camel Book) and this one.

## Supported Environments

Perl5.004 builds out of the box on Unix, Plan 9, LynxOS, VMS, OS/2, QNX, AmigaOS, and Windows NT. Perl runs on Windows 95 as well, but it cannot be built there, for lack of a reasonable command interpreter.

## Core Changes

Most importantly, many bugs were fixed, including several security problems. See the *Changes* file in the distribution for details.

### List assignment to `%ENV` works

`%ENV = ()` and `%ENV = @list` now work as expected (except on VMS where it generates a fatal error).

### Change to "Can't locate Foo.pm in `@INC`" error

The error ''Can't locate Foo.pm in `@INC`'' now lists the contents of `@INC` for easier debugging.

### Compilation option: Binary compatibility with 5.003

There is a new Configure question that asks if you want to maintain binary compatibility with Perl 5.003. If you choose binary compatibility, you do not have to recompile your extensions, but you might have symbol conflicts if you embed Perl in another application, just as in the 5.003 release. By default, binary compatibility is preserved at the expense of symbol table pollution.

### `$PERL5OPT` environment variable

You may now put Perl options in the `$PERL5OPT` environment variable. Unless Perl is running with taint checks, it will interpret this variable as if its contents had appeared on a ''#!perl'' line at the beginning of your script, except that hyphens are optional. PERL5OPT may only be used to set the following switches: **−[DIMUdmw]**.

### Limitations on −M, −m, and −T options

The `−M` and `−m` options are no longer allowed on the `#!` line of a script. If a script needs a module, it should invoke it with the `use` pragma.

The **−T** option is also forbidden on the `#!` line of a script, unless it was present on the Perl command line. Due to the way `#!` works, this usually means that **−T** must be in the first argument. Thus:

```
#!/usr/bin/perl −T −w
```

will probably work for an executable script invoked as `scriptname`, while:

```
#!/usr/bin/perl −w −T
```

will probably fail under the same conditions. (Non-Unix systems will probably not follow this rule.) But `perl scriptname` is guaranteed to fail, since then there is no chance of **−T** being found on the command line before it is found on the `#!` line.

### More precise warnings

If you removed the **−w** option from your Perl 5.003 scripts because it made Perl too verbose, we recommend that you try putting it back when you upgrade to Perl 5.004. Each new perl version tends to remove some undesirable warnings, while adding new warnings that may catch bugs in your scripts.

### Deprecated: Inherited `AUTOLOAD` for non-methods

Before Perl 5.004, `AUTOLOAD` functions were looked up as methods (using the `@ISA` hierarchy), even when the function to be autoloaded was called as a plain function (e.g. `Foo::bar()`), not a method (e.g. `Foo−>bar()` or `$obj−>bar()`).

Perl 5.005 will use method lookup only for methods' `AUTOLOAD`s. However, there is a significant base of existing code that may be using the old behavior. So, as an interim step, Perl 5.004 issues an optional warning when a non-method uses an inherited `AUTOLOAD`.

The simple rule is: Inheritance will not work when autoloading non-methods. The simple fix for old code is: In any module that used to depend on inheriting `AUTOLOAD` for non-methods from a base class named `BaseClass`, execute `*AUTOLOAD = \&BaseClass::AUTOLOAD` during startup.

### Previously deprecated %OVERLOAD is no longer usable

Using %OVERLOAD to define overloading was deprecated in 5.003. Overloading is now defined using the overload pragma. %OVERLOAD is still used internally but should not be used by Perl scripts. See overload for more details.

### Subroutine arguments created only when they're modified

In Perl 5.004, nonexistent array and hash elements used as subroutine parameters are brought into existence only if they are actually assigned to (via @_).

Earlier versions of Perl vary in their handling of such arguments. Perl versions 5.002 and 5.003 always brought them into existence. Perl versions 5.000 and 5.001 brought them into existence only if they were not the first argument (which was almost certainly a bug). Earlier versions of Perl never brought them into existence.

For example, given this code:

```
undef @a; undef %a;
sub show { print $_[0] };
sub change { $_[0]++ };
show($a[2]);
change($a{b});
```

After this code executes in Perl 5.004, $a{b} exists but $a[2] does not. In Perl 5.002 and 5.003, both $a{b} and $a[2] would have existed (but $a[2]'s value would have been undefined).

### Group vector changeable with $)

The $) special variable has always (well, in Perl 5, at least) reflected not only the current effective group, but also the group list as returned by the getgroups() C function (if there is one). However, until this release, there has not been a way to call the setgroups() C function from Perl.

In Perl 5.004, assigning to $) is exactly symmetrical with examining it: The first number in its string value is used as the effective gid; if there are any numbers after the first one, they are passed to the setgroups() C function (if there is one).

### Fixed parsing of $$<digit>, &$<digit>, etc.

Perl versions before 5.004 misinterpreted any type marker followed by "$" and a digit. For example, "$$0" was incorrectly taken to mean "${$}0" instead of "${$0}". This bug is (mostly) fixed in Perl 5.004.

However, the developers of Perl 5.004 could not fix this bug completely, because at least two widely-used modules depend on the old meaning of "$$0" in a string. So Perl 5.004 still interprets "$$<digit>" in the old (broken) way inside strings; but it generates this message as a warning. And in Perl 5.005, this special treatment will cease.

### Fixed localization of $<digit>, $&, etc.

Perl versions before 5.004 did not always properly localize the regex-related special variables. Perl 5.004 does localize them, as the documentation has always said it should. This may result in $1, $2, etc. no longer being set where existing programs use them.

### No resetting of $. on implicit close

The documentation for Perl 5.0 has always stated that $. is *not* reset when an already-open file handle is reopened with no intervening call to close. Due to a bug, perl versions 5.000 through 5.003 *did* reset $. under that circumstance; Perl 5.004 does not.

### wantarray may return undef

The wantarray operator returns true if a subroutine is expected to return a list, and false otherwise. In Perl 5.004, wantarray can also return the undefined value if a subroutine's return value will not be used at all, which allows subroutines to avoid a time-consuming calculation of a return value if it isn't going to be used.

### eval EXPR determines value of EXPR in scalar context

Perl (version 5) used to determine the value of EXPR inconsistently, sometimes incorrectly using the surrounding context for the determination. Now, the value of EXPR (before being parsed by eval) is always determined in a scalar context. Once parsed, it is executed as before, by providing the context that the scope surrounding the eval provided. This change makes the behavior Perl4 compatible, besides fixing bugs resulting from the inconsistent behavior. This program:

```
@a = qw(time now is time);
print eval @a;
print '|', scalar eval @a;
```

used to print something like ‘‘timenowis881399109|4’’, but now (and in perl4) prints ‘‘4|4’’.

### Changes to tainting checks

A bug in previous versions may have failed to detect some insecure conditions when taint checks are turned on. (Taint checks are used in setuid or setgid scripts, or when explicitly turned on with the `-T` invocation option.) Although it's unlikely, this may cause a previously-working script to now fail, which should be construed as a blessing since that indicates a potentially-serious security hole was just plugged.

The new restrictions when tainting include:

No **glob()** or <*>

These operators may spawn the C shell (csh), which cannot be made safe. This restriction will be lifted in a future version of Perl when globbing is implemented without the use of an external program.

No spawning if tainted `$CDPATH`, `$ENV`, `$BASH_ENV`

These environment variables may alter the behavior of spawned programs (especially shells) in ways that subvert security. So now they are treated as dangerous, in the manner of `$IFS` and `$PATH`.

No spawning if tainted `$TERM` doesn't look like a terminal name

Some termcap libraries do unsafe things with `$TERM`. However, it would be unnecessarily harsh to treat all `$TERM` values as unsafe, since only shell metacharacters can cause trouble in `$TERM`. So a tainted `$TERM` is considered to be safe if it contains only alphanumerics, underscores, dashes, and colons, and unsafe if it contains other characters (including whitespace).

### New Opcode module and revised Safe module

A new Opcode module supports the creation, manipulation and application of opcode masks. The revised Safe module has a new API and is implemented using the new Opcode module. Please read the new Opcode and Safe documentation.

### Embedding improvements

In older versions of Perl it was not possible to create more than one Perl interpreter instance inside a single process without leaking like a sieve and/or crashing. The bugs that caused this behavior have all been fixed. However, you still must take care when embedding Perl in a C program. See the updated perlembed manpage for tips on how to manage your interpreters.

### Internal change: FileHandle class based on IO::* classes

File handles are now stored internally as type IO::Handle. The FileHandle module is still supported for backwards compatibility, but it is now merely a front end to the IO::* modules, specifically IO::Handle, IO::Seekable, and IO::File. We suggest, but do not require, that you use the IO::* modules in new code.

In harmony with this change, `*GLOB{FILEHANDLE}` is now just a backward-compatible synonym for `*GLOB{IO}`.

### Internal change: PerlIO abstraction interface

It is now possible to build Perl with AT&T's sfio IO package instead of stdio. See perlapio for more details, and the *INSTALL* file for how to use it.

### New and changed syntax

`$coderef->`(PARAMS)

A subroutine reference may now be suffixed with an arrow and a (possibly empty) parameter list. This syntax denotes a call of the referenced subroutine, with the given parameters (if any).

This new syntax follows the pattern of `$hashref->{FOO}` and `$aryref->[$foo]`: You may now write `&$subref($foo)` as `$subref->($foo)`. All these arrow terms may be chained; thus, `&{$table->{FOO}}($bar)` may now be written `$table->{FOO}->($bar)`.

### New and changed builtin constants

__PACKAGE__

The current package name at compile time, or the undefined value if there is no current package (due to a `package;` directive). Like `__FILE__` and `__LINE__`, `__PACKAGE__` does *not* interpolate into strings.

### New and changed builtin variables

$^E Extended error message on some platforms. (Also known as `$EXTENDED_OS_ERROR` if you `use English`).

$^H

The current set of syntax checks enabled by `use strict`. See the documentation of `strict` for more details. Not actually new, but newly documented. Because it is intended for internal use by Perl core components, there is no `use English` long name for this variable.

$^M

By default, running out of memory it is not trappable. However, if compiled for this, Perl may use the contents of `$^M` as an emergency pool after **die()**ing with this message. Suppose that your Perl were compiled with -DPERL_EMERGENCY_SBRK and used Perl's malloc. Then

```
$^M = 'a' x (1<<16);
```

would allocate a 64K buffer for use when in emergency. See the *INSTALL* file for information on how to enable this option. As a disincentive to casual use of this advanced feature, there is no `use English` long name for this variable.

### New and changed builtin functions

delete on slices

This now works. (e.g. `delete @ENV{'PATH', 'MANPATH'}`)

flock

is now supported on more platforms, prefers fcntl to lockf when emulating, and always flushes before (un)locking.

printf and sprintf

Perl now implements these functions itself; it doesn't use the C library function **sprintf()** any more, except for floating-point numbers, and even then only known flags are allowed. As a result, it is now possible to know which conversions and flags will work, and what they will do.

The new conversions in Perl's **sprintf()** are:

```
%i   a synonym for %d
%p   a pointer (the address of the Perl value, in hexadecimal)
%n   special: *stores* the number of characters output so far
     into the next variable in the parameter list
```

The new flags that go between the % and the conversion are:

```
#    prefix octal with "0", hex with "0x"
h    interpret integer as C type "short" or "unsigned short"
V    interpret integer as Perl's standard integer type
```

Also, where a number would appear in the flags, an asterisk ("*") may be used instead, in which case Perl uses the next item in the parameter list as the given number (that is, as the field width or precision). If a field width obtained through "*" is negative, it has the same effect as the '-' flag: left-justification.

See "sprintf" in perlfunc for a complete list of conversion and flags.

keys as an lvalue

As an lvalue, `keys` allows you to increase the number of hash buckets allocated for the given hash. This can gain you a measure of efficiency if you know the hash is going to get big. (This is similar to pre-extending an array by assigning a larger number to $#array.) If you say

```
keys %hash = 200;
```

then `%hash` will have at least 200 buckets allocated for it. These buckets will be retained even if you do `%hash = ()`; use `undef %hash` if you want to free the storage while `%hash` is still in

scope. You can't shrink the number of buckets allocated for the hash using `keys` in this way (but you needn't worry about doing this by accident, as trying has no effect).

**my()** in Control Structures

You can now use **my()** (with or without the parentheses) in the control expressions of control structures such as:

```
while (defined(my $line = <>)) {
    $line = lc $line;
} continue {
    print $line;
}

if ((my $answer = <STDIN>) =~ /^y(es)?$/i) {
    user_agrees();
} elsif ($answer =~ /^n(o)?$/i) {
    user_disagrees();
} else {
    chomp $answer;
    die "`$answer' is neither `yes' nor `no'";
}
```

Also, you can declare a foreach loop control variable as lexical by preceding it with the word "my". For example, in:

```
foreach my $i (1, 2, 3) {
    some_function();
}
```

`$i` is a lexical variable, and the scope of `$i` extends to the end of the loop, but not beyond it.

Note that you still cannot use **my()** on global punctuation variables such as `$_` and the like.

**pack()** and **unpack()**

A new format 'w' represents a BER compressed integer (as defined in ASN.1). Its format is a sequence of one or more bytes, each of which provides seven bits of the total value, with the most significant first. Bit eight of each byte is set, except for the last byte, in which bit eight is clear.

If 'p' or 'P' are given undef as values, they now generate a NULL pointer.

Both **pack()** and **unpack()** now fail when their templates contain invalid types. (Invalid types used to be ignored.)

**sysseek()**

The new **sysseek()** operator is a variant of **seek()** that sets and gets the file's system read/write position, using the **lseek**(2) system call. It is the only reliable way to seek before using **sysread()** or **syswrite()**. Its return value is the new position, or the undefined value on failure.

use VERSION

If the first argument to `use` is a number, it is treated as a version number instead of a module name. If the version of the Perl interpreter is less than VERSION, then an error message is printed and Perl exits immediately. Because `use` occurs at compile time, this check happens immediately during the compilation process, unlike `require VERSION`, which waits until runtime for the check. This is often useful if you need to check the current Perl version before `use`ing library modules which have changed in incompatible ways from older versions of Perl. (We try not to do this more than we have to.)

use Module VERSION LIST

If the VERSION argument is present between Module and LIST, then the `use` will call the VERSION method in class Module with the given version as an argument. The default VERSION method, inherited from the UNIVERSAL class, croaks if the given version is larger than the value of the variable `$Module::VERSION`. (Note that there is not a comma after VERSION!)

This version-checking mechanism is similar to the one currently used in the Exporter module, but it is faster and can be used with modules that don't use the Exporter. It is the recommended method for new code.

prototype(FUNCTION)

Returns the prototype of a function as a string (or `undef` if the function has no prototype). FUNCTION is a reference to or the name of the function whose prototype you want to retrieve. (Not actually new; just never documented before.)

srand

The default seed for `srand`, which used to be `time`, has been changed. Now it's a heady mix of difficult-to-predict system-dependent values, which should be sufficient for most everyday purposes.

Previous to version 5.004, calling `rand` without first calling `srand` would yield the same sequence of random numbers on most or all machines. Now, when perl sees that you're calling `rand` and haven't yet called `srand`, it calls `srand` with the default seed. You should still call `srand` manually if your code might ever be run on a pre−5.004 system, of course, or if you want a seed other than the default.

$_ as Default

Functions documented in the Camel to default to `$_` now in fact do, and all those that do are so documented in perlfunc.

`m//gc` does not reset search position on failure

The `m//g` match iteration construct has always reset its target string's search position (which is visible through the `pos` operator) when a match fails; as a result, the next `m//g` match after a failure starts again at the beginning of the string. With Perl 5.004, this reset may be disabled by adding the "c" (for "continue") modifier, i.e. `m//gc`. This feature, in conjunction with the `\G` zero-width assertion, makes it possible to chain matches together. See perlop and perlre.

`m//x` ignores whitespace before ?*+{}

The `m//x` construct has always been intended to ignore all unescaped whitespace. However, before Perl 5.004, whitespace had the effect of escaping repeat modifiers like "*" or "?"; for example, `/a *b/x` was (mis)interpreted as `/a\*b/x`. This bug has been fixed in 5.004.

nested `sub{}` closures work now

Prior to the 5.004 release, nested anonymous functions didn't work right. They do now.

formats work right on changing lexicals

Just like anonymous functions that contain lexical variables that change (like a lexical index variable for a `foreach` loop), formats now work properly. For example, this silently failed before (printed only zeros), but is fine now:

```
my $i;
foreach $i ( 1 .. 10 ) {
    write;
}
format =
    my i is @#
    $i
.
```

However, it still fails (without a warning) if the foreach is within a subroutine:

```
my $i;
sub foo {
  foreach $i ( 1 .. 10 ) {
    write;
  }
}
foo;
format =
    my i is @#
    $i
.
```

### New builtin methods

The `UNIVERSAL` package automatically contains the following methods that are inherited by all other classes:

isa(CLASS)

`isa` returns *true* if its object is blessed into a subclass of `CLASS`

`isa` is also exportable and can be called as a sub with two arguments. This allows the ability to check what a reference points to. Example:

```
use UNIVERSAL qw(isa);

if(isa($ref, 'ARRAY')) {
   ...
}
```

can(METHOD)

`can` checks to see if its object has a method called `METHOD`, if it does then a reference to the sub is returned; if it does not then *undef* is returned.

VERSION( [NEED] )

`VERSION` returns the version number of the class (package). If the NEED argument is given then it will check that the current version (as defined by the `$VERSION` variable in the given package) not less than NEED; it will die if this is not the case. This method is normally called as a class method. This method is called automatically by the `VERSION` form of `use`.

```
use A 1.2 qw(some imported subs);
# implies:
A->VERSION(1.2);
```

**NOTE:** `can` directly uses Perl's internal code for method lookup, and `isa` uses a very similar method and caching strategy. This may cause strange effects if the Perl code dynamically changes `@ISA` in any package.

You may add other methods to the UNIVERSAL class via Perl or XS code. You do not need to `use UNIVERSAL` in order to make these methods available to your program. This is necessary only if you wish to have `isa` available as a plain subroutine in the current package.

### TIEHANDLE now supported

See perltie for other kinds of **tie()**s.

TIEHANDLE classname, LIST

This is the constructor for the class. That means it is expected to return an object of some sort. The reference can be used to hold some internal information.

```
sub TIEHANDLE {
    print "<shout>\n";
    my $i;
    return bless \$i, shift;
}
```

PRINT this, LIST

This method will be triggered every time the tied handle is printed to. Beyond its self reference it also expects the list that was passed to the print function.

```
sub PRINT {
    $r = shift;
    $$r++;
    return print join( $, => map {uc} @_), $\;
}
```

PRINTF this, LIST

This method will be triggered every time the tied handle is printed to with the `printf()` function. Beyond its self reference it also expects the format and list that was passed to the printf function.

```
sub PRINTF {
    shift;
      my $fmt = shift;
    print sprintf($fmt, @_)."\n";
}
```

READ this LIST

This method will be called when the handle is read from via the `read` or `sysread` functions.

```
sub READ {
    $r = shift;
    my($buf,$len,$offset) = @_;
    print "READ called, \$buf=$buf, \$len=$len, \$offset=$offset";
}
```

READLINE this

This method will be called when the handle is read from. The method should return undef when there is no more data.

```
sub READLINE {
    $r = shift;
    return "PRINT called $$r times\n"
}
```

GETC this

This method will be called when the `getc` function is called.

```
sub GETC { print "Don't GETC, Get Perl"; return "a"; }
```

DESTROY this

As with the other types of ties, this method will be called when the tied handle is about to be destroyed. This is useful for debugging and possibly for cleaning up.

```
sub DESTROY {
    print "</shout>\n";
}
```

### Malloc enhancements

If perl is compiled with the malloc included with the perl distribution (that is, if `perl -V:d_mymalloc` is 'define') then you can print memory statistics at runtime by running Perl thusly:

```
env PERL_DEBUG_MSTATS=2 perl your_script_here
```

The value of 2 means to print statistics after compilation and on exit; with a value of 1, the statistics are printed only on exit. (If you want the statistics at an arbitrary time, you'll need to install the optional module Devel::Peek.)

Three new compilation flags are recognized by malloc.c. (They have no effect if perl is compiled with system **malloc()**.)

−DPERL_EMERGENCY_SBRK

If this macro is defined, running out of memory need not be a fatal error: a memory pool can allocated by assigning to the special variable `$^M`. See ''$^M''.

−DPACK_MALLOC

Perl memory allocation is by bucket with sizes close to powers of two. Because of these malloc overhead may be big, especially for data of size exactly a power of two. If `PACK_MALLOC` is defined, perl uses a slightly different algorithm for small allocations (up to 64 bytes long), which makes it possible to have overhead down to 1 byte for allocations which are powers of two (and appear quite often).

Expected memory savings (with 8−byte alignment in `alignbytes`) is about 20% for typical Perl usage. Expected slowdown due to additional malloc overhead is in fractions of a percent (hard to measure, because of the effect of saved memory on speed).

−DTWO_POT_OPTIMIZE

Similarly to `PACK_MALLOC`, this macro improves allocations of data with size close to a power of two; but this works for big allocations (starting with 16K by default). Such allocations are typical

for big hashes and special-purpose scripts, especially image processing.

On recent systems, the fact that perl requires 2M from system for 1M allocation will not affect speed of execution, since the tail of such a chunk is not going to be touched (and thus will not require real memory). However, it may result in a premature out-of-memory error. So if you will be manipulating very large blocks with sizes close to powers of two, it would be wise to define this macro.

Expected saving of memory is 0−100% (100% in applications which require most memory in such 2**n chunks); expected slowdown is negligible.

### Miscellaneous efficiency enhancements

Functions that have an empty prototype and that do nothing but return a fixed value are now inlined (e.g. `sub PI () { 3.14159 }`).

Each unique hash key is only allocated once, no matter how many hashes have an entry with that key. So even if you have 100 copies of the same hash, the hash keys never have to be reallocated.

## Support for More Operating Systems

Support for the following operating systems is new in Perl 5.004.

### Win32

Perl 5.004 now includes support for building a ''native'' perl under Windows NT, using the Microsoft Visual C++ compiler (versions 2.0 and above) or the Borland C++ compiler (versions 5.02 and above). The resulting perl can be used under Windows 95 (if it is installed in the same directory locations as it got installed in Windows NT). This port includes support for perl extension building tools like ExtUtils::MakeMaker and h2xs, so that many extensions available on the Comprehensive Perl Archive Network (CPAN) can now be readily built under Windows NT. See http://www.perl.com/ for more information on CPAN and *README.win32* in the perl distribution for more details on how to get started with building this port.

There is also support for building perl under the Cygwin32 environment. Cygwin32 is a set of GNU tools that make it possible to compile and run many Unix programs under Windows NT by providing a mostly Unix-like interface for compilation and execution. See *README.cygwin32* in the perl distribution for more details on this port and how to obtain the Cygwin32 toolkit.

### Plan 9

See *README.plan9* in the perl distribution.

### QNX

See *README.qnx* in the perl distribution.

### AmigaOS

See *README.amigaos* in the perl distribution.

## Pragmata

Six new pragmatic modules exist:

use autouse MODULE => qw(sub1 sub2 sub3)

Defers `require MODULE` until someone calls one of the specified subroutines (which must be exported by MODULE). This pragma should be used with caution, and only when necessary.

use blib
use blib 'dir'

Looks for MakeMaker-like *'blib'* directory structure starting in *dir* (or current directory) and working back up to five levels of parent directories.

Intended for use on command line with **−M** option as a way of testing arbitrary scripts against an uninstalled version of a package.

use constant NAME => VALUE

Provides a convenient interface for creating compile-time constants, See ''Constant Functions'' in perlsub.

use locale

Tells the compiler to enable (or disable) the use of POSIX locales for builtin operations.

When `use locale` is in effect, the current LC_CTYPE locale is used for regular expressions and

case mapping; LC_COLLATE for string ordering; and LC_NUMERIC for numeric formatting in printf and sprintf (but **not** in print). LC_NUMERIC is always used in write, since lexical scoping of formats is problematic at best.

Each `use locale` or `no locale` affects statements to the end of the enclosing BLOCK or, if not inside a BLOCK, to the end of the current file. Locales can be switched and queried with **POSIX::setlocale()**.

See perllocale for more information.

use ops
: Disable unsafe opcodes, or any named opcodes, when compiling Perl code.

use vmsish
: Enable VMS-specific language features. Currently, there are three VMS-specific features available: 'status', which makes `$?` and `system` return genuine VMS status values instead of emulating POSIX; 'exit', which makes `exit` take a genuine VMS status value instead of assuming that `exit 1` is an error; and 'time', which makes all times relative to the local time zone, in the VMS tradition.

## Modules

### Required Updates

Though Perl 5.004 is compatible with almost all modules that work with Perl 5.003, there are a few exceptions:

```
Module   Required Version for Perl 5.004
------   ------------------------------
Filter   Filter-1.12
LWP      libwww-perl-5.08
Tk       Tk400.202 (-w makes noise)
```

Also, the majordomo mailing list program, version 1.94.1, doesn't work with Perl 5.004 (nor with perl 4), because it executes an invalid regular expression. This bug is fixed in majordomo version 1.94.2.

### Installation directories

The *installperl* script now places the Perl source files for extensions in the architecture-specific library directory, which is where the shared libraries for extensions have always been. This change is intended to allow administrators to keep the Perl 5.004 library directory unchanged from a previous version, without running the risk of binary incompatibility between extensions' Perl source and shared libraries.

### Module information summary

Brand new modules, arranged by topic rather than strictly alphabetically:

```
CGI.pm              Web server interface ("Common Gateway Interface")
CGI/Apache.pm       Support for Apache's Perl module
CGI/Carp.pm         Log server errors with helpful context
CGI/Fast.pm         Support for FastCGI (persistent server process)
CGI/Push.pm         Support for server push
CGI/Switch.pm       Simple interface for multiple server types

CPAN                Interface to Comprehensive Perl Archive Network
CPAN::FirstTime     Utility for creating CPAN configuration file
CPAN::Nox           Runs CPAN while avoiding compiled extensions

IO.pm               Top-level interface to IO::* classes
IO/File.pm          IO::File extension Perl module
IO/Handle.pm        IO::Handle extension Perl module
IO/Pipe.pm          IO::Pipe extension Perl module
IO/Seekable.pm      IO::Seekable extension Perl module
IO/Select.pm        IO::Select extension Perl module
IO/Socket.pm        IO::Socket extension Perl module

Opcode.pm           Disable named opcodes when compiling Perl code
```

```
ExtUtils/Embed.pm    Utilities for embedding Perl in C programs
ExtUtils/testlib.pm  Fixes up @INC to use just-built extension

FindBin.pm           Find path of currently executing program

Class/Struct.pm      Declare struct-like datatypes as Perl classes
File/stat.pm         By-name interface to Perl's builtin stat
Net/hostent.pm       By-name interface to Perl's builtin gethost*
Net/netent.pm        By-name interface to Perl's builtin getnet*
Net/protoent.pm      By-name interface to Perl's builtin getproto*
Net/servent.pm       By-name interface to Perl's builtin getserv*
Time/gmtime.pm       By-name interface to Perl's builtin gmtime
Time/localtime.pm    By-name interface to Perl's builtin localtime
Time/tm.pm           Internal object for Time::{gm,local}time
User/grent.pm        By-name interface to Perl's builtin getgr*
User/pwent.pm        By-name interface to Perl's builtin getpw*

Tie/RefHash.pm       Base class for tied hashes with references as keys

UNIVERSAL.pm         Base class for *ALL* classes
```

### Fcntl

New constants in the existing Fcntl modules are now supported, provided that your operating system happens to support them:

```
F_GETOWN F_SETOWN
O_ASYNC O_DEFER O_DSYNC O_FSYNC O_SYNC
O_EXLOCK O_SHLOCK
```

These constants are intended for use with the Perl operators **sysopen()** and **fcntl()** and the basic database modules like SDBM_File. For the exact meaning of these and other Fcntl constants please refer to your operating system's documentation for **fcntl()** and **open()**.

In addition, the Fcntl module now provides these constants for use with the Perl operator **flock()**:

```
LOCK_SH LOCK_EX LOCK_NB LOCK_UN
```

These constants are defined in all environments (because where there is no **flock()** system call, Perl emulates it). However, for historical reasons, these constants are not exported unless they are explicitly requested with the ":flock" tag (e.g. `use Fcntl ':flock'`).

### IO

The IO module provides a simple mechanism to load all the IO modules at one go. Currently this includes:

```
IO::Handle
IO::Seekable
IO::File
IO::Pipe
IO::Socket
```

For more information on any of these modules, please see its respective documentation.

### Math::Complex

The Math::Complex module has been totally rewritten, and now supports more operations. These are overloaded:

```
+ - * / ** <=> neg ~ abs sqrt exp log sin cos atan2 "" (stringify)
```

And these functions are now exported:

```
pi i Re Im arg
log10 logn ln cbrt root
tan
csc sec cot
asin acos atan
acsc asec acot
sinh cosh tanh
csch sech coth
asinh acosh atanh
acsch asech acoth
cplx cplxe
```

### Math::Trig

This new module provides a simpler interface to parts of Math::Complex for those who need trigonometric functions only for real numbers.

### DB_File

There have been quite a few changes made to DB_File. Here are a few of the highlights:

- Fixed a handful of bugs.
- By public demand, added support for the standard hash function **exists()**.
- Made it compatible with Berkeley DB 1.86.
- Made negative subscripts work with RECNO interface.
- Changed the default flags from O_RDWR to O_CREAT|O_RDWR and the default mode from 0640 to 0666.
- Made DB_File automatically import the **open()** constants (O_RDWR, O_CREAT etc.) from Fcntl, if available.
- Updated documentation.

Refer to the HISTORY section in DB_File.pm for a complete list of changes. Everything after DB_File 1.01 has been added since 5.003.

### Net::Ping

Major rewrite − support added for both udp echo and real icmp pings.

### Object-oriented overrides for builtin operators

Many of the Perl builtins returning lists now have object-oriented overrides. These are:

```
File::stat
Net::hostent
Net::netent
Net::protoent
Net::servent
Time::gmtime
Time::localtime
User::grent
User::pwent
```

For example, you can now say

```
use File::stat;
use User::pwent;
$his = (stat($filename)->st_uid == pwent($whoever)->pw_uid);
```

## Utility Changes

### pod2html

Sends converted HTML to standard output

The *pod2html* utility included with Perl 5.004 is entirely new. By default, it sends the converted HTML to its standard output, instead of writing it to a file like Perl 5.003's *pod2html* did. Use the **−−outfile=FILENAME** option to write to a file.

### xsubpp

`void` XSUBs now default to returning nothing

Due to a documentation/implementation bug in previous versions of Perl, XSUBs with a return type of `void` have actually been returning one value. Usually that value was the GV for the XSUB, but sometimes it was some already freed or reused value, which would sometimes lead to program failure.

In Perl 5.004, if an XSUB is declared as returning `void`, it actually returns no value, i.e. an empty list (though there is a backward-compatibility exception; see below). If your XSUB really does return an SV, you should give it a return type of `SV *`.

For backward compatibility, *xsubpp* tries to guess whether a `void` XSUB is really `void` or if it wants to return an `SV *`. It does so by examining the text of the XSUB: if *xsubpp* finds what looks like an assignment to `ST(0)`, it assumes that the XSUB's return type is really `SV *`.

## C Language API Changes

`gv_fetchmethod` and `perl_call_sv`

The `gv_fetchmethod` function finds a method for an object, just like in Perl 5.003. The GV it returns may be a method cache entry. However, in Perl 5.004, method cache entries are not visible to users; therefore, they can no longer be passed directly to `perl_call_sv`. Instead, you should use the `GvCV` macro on the GV to extract its CV, and pass the CV to `perl_call_sv`.

The most likely symptom of passing the result of `gv_fetchmethod` to `perl_call_sv` is Perl's producing an ''Undefined subroutine called'' error on the *second* call to a given method (since there is no cache on the first call).

`perl_eval_pv`

A new function handy for eval'ing strings of Perl code inside C code. This function returns the value from the eval statement, which can be used instead of fetching globals from the symbol table. See perlguts, perlembed and perlcall for details and examples.

Extended API for manipulating hashes

Internal handling of hash keys has changed. The old hashtable API is still fully supported, and will likely remain so. The additions to the API allow passing keys as `SV*`s, so that `tied` hashes can be given real scalars as keys rather than plain strings (nontied hashes still can only use strings as keys). New extensions must use the new hash access functions and macros if they wish to use `SV*` keys. These additions also make it feasible to manipulate `HE*`s (hash entries), which can be more efficient. See perlguts for details.

## Documentation Changes

Many of the base and library pods were updated. These new pods are included in section 1:

perldelta
: This document.

perlfaq
: Frequently asked questions.

perllocale
: Locale support (internationalization and localization).

perltoot
: Tutorial on Perl OO programming.

perlapio
: Perl internal IO abstraction interface.

perlmodlib
: Perl module library and recommended practice for module creation. Extracted from perlmod (which is much smaller as a result).

perldebug
: Although not new, this has been massively updated.

perlsec
: Although not new, this has been massively updated.

## New Diagnostics

Several new conditions will trigger warnings that were silent before. Some only affect certain platforms. The following new warnings and errors outline these. These messages are classified as follows (listed in increasing order of desperation):

```
(W) A warning (optional).
(D) A deprecation (optional).
(S) A severe warning (mandatory).
(F) A fatal error (trappable).
(P) An internal error you should never see (trappable).
(X) A very fatal error (nontrappable).
(A) An alien error message (not generated by Perl).
```

‘‘my’’ variable %s masks earlier declaration in same scope

(W) A lexical variable has been redeclared in the same scope, effectively eliminating all access to the previous instance. This is almost always a typographical error. Note that the earlier variable will still exist until the end of the scope or until all closure referents to it are destroyed.

%s argument is not a HASH element or slice

(F) The argument to **delete()** must be either a hash element, such as

```
$foo{$bar}
$ref->[12]->{"susie"}
```

or a hash slice, such as

```
@foo{$bar, $baz, $xyzzy}
@{$ref->[12]}{"susie", "queue"}
```

Allocation too large: %lx

(X) You can’t allocate more than 64K on an MS-DOS machine.

Allocation too large

(F) You can’t allocate more than 2^31+‘‘small amount’’ bytes.

Applying %s to %s will act on scalar(%s)

(W) The pattern match (//), substitution (s///), and transliteration (tr///) operators work on scalar values. If you apply one of them to an array or a hash, it will convert the array or hash to a scalar value (the length of an array or the population info of a hash) and then work on that scalar value. This is probably not what you meant to do. See ‘‘grep’’ in perlfunc and ‘‘map’’ in perlfunc for alternatives.

Attempt to free nonexistent shared string

(P) Perl maintains a reference counted internal table of strings to optimize the storage and access of hash keys and other strings. This indicates someone tried to decrement the reference count of a string that can no longer be found in the table.

Attempt to use reference as lvalue in substr

(W) You supplied a reference as the first argument to **substr()** used as an lvalue, which is pretty strange. Perhaps you forgot to dereference it first. See ‘‘substr’’ in perlfunc.

Bareword ‘‘%s’’ refers to nonexistent package

(W) You used a qualified bareword of the form `Foo::`, but the compiler saw no other uses of that namespace before that point. Perhaps you need to predeclare a package?

Can’t redefine active sort subroutine %s

(F) Perl optimizes the internal handling of sort subroutines and keeps pointers into them. You tried to redefine one such sort subroutine when it was currently active, which is not allowed. If you really want to do this, you should write `sort { &func } @x` instead of `sort func @x`.

Can’t use bareword (‘‘%s’’) as %s ref while ‘‘strict refs’’ in use

(F) Only hard references are allowed by ‘‘strict refs’’. Symbolic references are disallowed. See perlref.

Cannot resolve method ‘%s’ overloading ‘%s’ in package ‘%s’
(P) Internal error trying to resolve overloading specified by a method name (as opposed to a subroutine reference).

Constant subroutine `%s` redefined
(S) You redefined a subroutine which had previously been eligible for inlining. See ‘‘Constant Functions’’ in perlsub for commentary and workarounds.

Constant subroutine `%s` undefined
(S) You undefined a subroutine which had previously been eligible for inlining. See ‘‘Constant Functions’’ in perlsub for commentary and workarounds.

Copy method did not return a reference
(F) The method which overloads ‘‘=’’ is buggy. See ‘‘Copy Constructor’’ in overload.

Died
(F) You passed **die()** an empty string (the equivalent of `die ""`) or you called it with no args and both `$@` and `$_` were empty.

Exiting pseudo-block via `%s`
(W) You are exiting a rather special block construct (like a sort block or subroutine) by unconventional means, such as a goto, or a loop control statement. See ‘‘sort’’ in perlfunc.

Identifier too long
(F) Perl limits identifiers (names for variables, functions, etc.) to 252 characters for simple names, somewhat more for compound names (like `$A::B`). You’ve exceeded Perl’s limits. Future versions of Perl are likely to eliminate these arbitrary limitations.

Illegal character `%s` (carriage return)
(F) A carriage return character was found in the input. This is an error, and not a warning, because carriage return characters can break multi-line strings, including here documents (e.g., `print <<EOF;`).

Illegal switch in PERL5OPT: `%s`
(X) The PERL5OPT environment variable may only be used to set the following switches: **−[DIMUdmw]**.

Integer overflow in hex number
(S) The literal hex number you have specified is too big for your architecture. On a 32−bit architecture the largest hex literal is 0xFFFFFFFF.

Integer overflow in octal number
(S) The literal octal number you have specified is too big for your architecture. On a 32−bit architecture the largest octal literal is 037777777777.

internal error: glob failed
(P) Something went wrong with the external program(s) used for `glob` and `<*.c>`. This may mean that your csh (C shell) is broken. If so, you should change all of the csh-related variables in config.sh: If you have tcsh, make the variables refer to it as if it were csh (e.g. `full_csh='/usr/bin/tcsh'`); otherwise, make them all empty (except that `d_csh` should be `'undef'`) so that Perl will think csh is missing. In either case, after editing config.sh, run `./Configure −S` and rebuild Perl.

Invalid conversion in `%s:` ‘‘%s’’
(W) Perl does not understand the given format conversion. See ‘‘sprintf’’ in perlfunc.

Invalid type in pack: ’%s’
(F) The given character is not a valid pack type. See ‘‘pack’’ in perlfunc.

Invalid type in unpack: ’%s’
(F) The given character is not a valid unpack type. See ‘‘unpack’’ in perlfunc.

Name ‘‘%s::%s’’ used only once: possible typo
(W) Typographical errors often show up as unique variable names. If you had a good reason for having a unique name, then just mention it again somehow to suppress the message (the `use vars` pragma is provided for just this purpose).

Null picture in formline
(F) The first argument to formline must be a valid format picture specification. It was found to be empty, which probably means you supplied it an uninitialized value. See perlform.

Offset outside string
(F) You tried to do a read/write/send/recv operation with an offset pointing outside the buffer. This is difficult to imagine. The sole exception to this is that `sysread()`ing past the buffer will extend the buffer and zero pad the new area.

Out of memory!
(X|F) The **malloc()** function returned 0, indicating there was insufficient remaining memory (or virtual memory) to satisfy the request.

The request was judged to be small, so the possibility to trap it depends on the way Perl was compiled. By default it is not trappable. However, if compiled for this, Perl may use the contents of `$^M` as an emergency pool after **die()**ing with this message. In this case the error is trappable *once*.

Out of memory during request for `%s`
(F) The **malloc()** function returned 0, indicating there was insufficient remaining memory (or virtual memory) to satisfy the request. However, the request was judged large enough (compile-time default is 64K), so a possibility to shut down by trapping this error is granted.

panic: frexp
(P) The library function **frexp()** failed, making printf(“%f”) impossible.

Possible attempt to put comments in **qw()** list
(W) **qw()** lists contain items separated by whitespace; as with literal strings, comment characters are not ignored, but are instead treated as literal data. (You may have used different delimiters than the parentheses shown here; braces are also frequently used.)

You probably wrote something like this:

```
@list = qw(
    a # a comment
    b # another comment
);
```

when you should have written this:

```
@list = qw(
    a
    b
);
```

If you really want comments, build your list the old-fashioned way, with quotes and commas:

```
@list = (
    'a',    # a comment
    'b',    # another comment
);
```

Possible attempt to separate words with commas
(W) **qw()** lists contain items separated by whitespace; therefore commas aren’t needed to separate the items. (You may have used different delimiters than the parentheses shown here; braces are also frequently used.)

You probably wrote something like this:

```
qw! a, b, c !;
```

which puts literal commas into some of the list items. Write it without commas if you don’t want them to appear in your data:

```
qw! a b c !;
```

Scalar value @%s{%s} better written as $%s{%s}

(W) You've used a hash slice (indicated by @) to select a single element of a hash. Generally it's better to ask for a scalar value (indicated by $). The difference is that `$foo{&bar}` always behaves like a scalar, both when assigning to it and when evaluating its argument, while `@foo{&bar}` behaves like a list when you assign to it, and provides a list context to its subscript, which can do weird things if you're expecting only one subscript.

Stub found while resolving method '%s' overloading '%s' in `%s`

(P) Overloading resolution over `@ISA` tree may be broken by importing stubs. Stubs should never be implicitly created, but explicit calls to `can` may break this.

Too late for "**-T**" option

(X) The #! line (or local equivalent) in a Perl script contains the **-T** option, but Perl was not invoked with **-T** in its argument list. This is an error because, by the time Perl discovers a **-T** in a script, it's too late to properly taint everything from the environment. So Perl gives up.

untie attempted while `%d` inner references still exist

(W) A copy of the object returned from `tie` (or `tied`) was still valid when `untie` was called.

Unrecognized character `%s`

(F) The Perl parser has no idea what to do with the specified character in your Perl script (or eval). Perhaps you tried to run a compressed script, a binary program, or a directory as a Perl program.

Unsupported function fork

(F) Your version of executable does not support forking.

Note that under some systems, like OS/2, there may be different flavors of Perl executables, some of which may support fork, some not. Try changing the name you call Perl by to `perl_`, `perl__`, and so on.

Use of "$$<digit>" to mean "${$}<digit>" is deprecated

(D) Perl versions before 5.004 misinterpreted any type marker followed by "$" and a digit. For example, "$$0" was incorrectly taken to mean "${$}0" instead of "${$0}". This bug is (mostly) fixed in Perl 5.004.

However, the developers of Perl 5.004 could not fix this bug completely, because at least two widely-used modules depend on the old meaning of "$$0" in a string. So Perl 5.004 still interprets "$$<digit>" in the old (broken) way inside strings; but it generates this message as a warning. And in Perl 5.005, this special treatment will cease.

Value of `%s` can be "0"; test with **defined()**

(W) In a conditional expression, you used <HANDLE>, <*> (glob), `each()`, or `readdir()` as a boolean value. Each of these constructs can return a value of "0"; that would make the conditional expression false, which is probably not what you intended. When using these constructs in conditional expressions, test their values with the `defined` operator.

Variable "%s" may be unavailable

(W) An inner (nested) *anonymous* subroutine is inside a *named* subroutine, and outside that is another subroutine; and the anonymous (innermost) subroutine is referencing a lexical variable defined in the outermost subroutine. For example:

```
sub outermost { my $a; sub middle { sub { $a } } }
```

If the anonymous subroutine is called or referenced (directly or indirectly) from the outermost subroutine, it will share the variable as you would expect. But if the anonymous subroutine is called or referenced when the outermost subroutine is not active, it will see the value of the shared variable as it was before and during the *first* call to the outermost subroutine, which is probably not what you want.

In these circumstances, it is usually best to make the middle subroutine anonymous, using the `sub {}` syntax. Perl has specific support for shared variables in nested anonymous subroutines; a named subroutine in between interferes with this feature.

Variable "%s" will not stay shared

(W) An inner (nested) *named* subroutine is referencing a lexical variable defined in an outer subroutine.

When the inner subroutine is called, it will probably see the value of the outer subroutine's variable as it was before and during the *first* call to the outer subroutine; in this case, after the first call to the outer subroutine is complete, the inner and outer subroutines will no longer share a common value for the variable. In other words, the variable will no longer be shared.

Furthermore, if the outer subroutine is anonymous and references a lexical variable outside itself, then the outer and inner subroutines will *never* share the given variable.

This problem can usually be solved by making the inner subroutine anonymous, using the `sub {}` syntax. When inner anonymous subs that reference variables in outer subroutines are called or referenced, they are automatically rebound to the current values of such variables.

Warning: something's wrong
(W) You passed **warn()** an empty string (the equivalent of `warn ""`) or you called it with no args and `$_` was empty.

Ill-formed logical name |%s| in prime_env_iter
(W) A warning peculiar to VMS. A logical name was encountered when preparing to iterate over `%ENV` which violates the syntactic rules governing logical names. Since it cannot be translated normally, it is skipped, and will not appear in `%ENV`. This may be a benign occurrence, as some software packages might directly modify logical name tables and introduce nonstandard names, or it may indicate that a logical name table has been corrupted.

Got an error from DosAllocMem
(P) An error peculiar to OS/2. Most probably you're using an obsolete version of Perl, and this should not happen anyway.

Malformed PERLLIB_PREFIX
(F) An error peculiar to OS/2. PERLLIB_PREFIX should be of the form

```
prefix1;prefix2
```

or

```
prefix1 prefix2
```

with nonempty prefix1 and prefix2. If `prefix1` is indeed a prefix of a builtin library search path, prefix2 is substituted. The error may appear if components are not found, or are too long. See ''PERLLIB_PREFIX'' in *README.os2*.

PERL_SH_DIR too long
(F) An error peculiar to OS/2. PERL_SH_DIR is the directory to find the `sh`-shell in. See ''PERL_SH_DIR'' in *README.os2*.

Process terminated by SIG%s
(W) This is a standard message issued by OS/2 applications, while *nix applications die in silence. It is considered a feature of the OS/2 port. One can easily disable this by appropriate sighandlers, see ''Signals'' in perlipc. See also ''Process terminated by SIGTERM/SIGINT'' in *README.os2*.

## BUGS

If you find what you think is a bug, you might check the headers of recently posted articles in the comp.lang.perl.misc newsgroup. There may also be information at http://www.perl.com/perl/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Make sure you trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to <*perlbug@perl.com*> to be analysed by the Perl porting team.

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl. This file has been significantly updated for 5.004, so even veteran users should look through it.

The *README* file for general stuff.

The *Copying* file for copyright information.

## HISTORY

Constructed by Tom Christiansen, grabbing material with permission from innumerable contributors, with kibitzing by more than a few Perl porters.

Last update: Wed May 14 11:14:09 EDT 1997

## NAME

perl5005delta − what's new for perl5.005

## DESCRIPTION

This document describes differences between the 5.004 release and this one.

## About the new versioning system

Perl is now developed on two tracks: a maintenance track that makes small, safe updates to released production versions with emphasis on compatibility; and a development track that pursues more aggressive evolution. Maintenance releases (which should be considered production quality) have subversion numbers that run from `1` to `49`, and development releases (which should be considered ''alpha'' quality) run from `50` to `99`.

Perl 5.005 is the combined product of the new dual-track development scheme.

## Incompatible Changes

### WARNING: This version is not binary compatible with Perl 5.004.

Starting with Perl 5.004_50 there were many deep and far-reaching changes to the language internals. If you have dynamically loaded extensions that you built under perl 5.003 or 5.004, you can continue to use them with 5.004, but you will need to rebuild and reinstall those extensions to use them 5.005. See *INSTALL* for detailed instructions on how to upgrade.

### Default installation structure has changed

The new Configure defaults are designed to allow a smooth upgrade from 5.004 to 5.005, but you should read *INSTALL* for a detailed discussion of the changes in order to adapt them to your system.

### Perl Source Compatibility

When none of the experimental features are enabled, there should be very few user-visible Perl source compatibility issues.

If threads are enabled, then some caveats apply. `@_` and `$_` become lexical variables. The effect of this should be largely transparent to the user, but there are some boundary conditions under which user will need to be aware of the issues. For example, `local(@_)` results in a ''Can't localize lexical variable `@_` ...'' message. This may be enabled in a future version.

Some new keywords have been introduced. These are generally expected to have very little impact on compatibility. See "New `INIT` keyword", "New `lock` keyword", and "New `qr//` operator".

Certain barewords are now reserved. Use of these will provoke a warning if you have asked for them with the `-w` switch. See "`our` is now a reserved word".

### C Source Compatibility

There have been a large number of changes in the internals to support the new features in this release.

- Core sources now require ANSI C compiler

  An ANSI C compiler is now **required** to build perl. See *INSTALL*.

- All Perl global variables must now be referenced with an explicit prefix

  All Perl global variables that are visible for use by extensions now have a `PL_` prefix. New extensions should `not` refer to perl globals by their unqualified names. To preserve sanity, we provide limited backward compatibility for globals that are being widely used like `sv_undef` and `na` (which should now be written as `PL_sv_undef`, `PL_na` etc.)

  If you find that your XS extension does not compile anymore because a perl global is not visible, try adding a `PL_` prefix to the global and rebuild.

  It is strongly recommended that all functions in the Perl API that don't begin with `perl` be referenced with a `Perl_` prefix. The bare function names without the `Perl_` prefix are supported with macros, but this support may cease in a future release.

  See perlapi.

- Enabling threads has source compatibility issues

  Perl built with threading enabled requires extensions to use the new `dTHR` macro to initialize the handle to access per-thread data. If you see a compiler error that talks about the variable `thr` not being declared (when building a module that has XS code), you need to add `dTHR;` at the

beginning of the block that elicited the error.

The API function `perl_get_sv("@",GV_ADD)` should be used instead of directly accessing perl globals as `GvSV(errgv)`. The API call is backward compatible with existing perls and provides source compatibility with threading is enabled.

See ''C Source Compatibility'' for more information.

### Binary Compatibility

This version is NOT binary compatible with older versions. All extensions will need to be recompiled. Further binaries built with threads enabled are incompatible with binaries built without. This should largely be transparent to the user, as all binary incompatible configurations have their own unique architecture name, and extension binaries get installed at unique locations. This allows coexistence of several configurations in the same directory hierarchy. See *INSTALL*.

### Security fixes may affect compatibility

A few taint leaks and taint omissions have been corrected. This may lead to ''failure'' of scripts that used to work with older versions. Compiling with -DINCOMPLETE_TAINTS provides a perl with minimal amounts of changes to the tainting behavior. But note that the resulting perl will have known insecurities.

Oneliners with the `-e` switch do not create temporary files anymore.

### Relaxed new mandatory warnings introduced in 5.004

Many new warnings that were introduced in 5.004 have been made optional. Some of these warnings are still present, but perl's new features make them less often a problem. See ''New Diagnostics''.

### Licensing

Perl has a new Social Contract for contributors. See *Porting/Contract*.

The license included in much of the Perl documentation has changed. Most of the Perl documentation was previously under the implicit GNU General Public License or the Artistic License (at the user's choice). Now much of the documentation unambiguously states the terms under which it may be distributed. Those terms are in general much less restrictive than the GNU GPL. See perl and the individual perl manpages listed therein.

## Core Changes

### Threads

WARNING: Threading is considered an **experimental** feature. Details of the implementation may change without notice. There are known limitations and some bugs. These are expected to be fixed in future versions.

See *README.threads*.

### Compiler

WARNING: The Compiler and related tools are considered **experimental**. Features may change without notice, and there are known limitations and bugs. Since the compiler is fully external to perl, the default configuration will build and install it.

The Compiler produces three different types of transformations of a perl program. The C backend generates C code that captures perl's state just before execution begins. It eliminates the compile-time overheads of the regular perl interpreter, but the run-time performance remains comparatively the same. The CC backend generates optimized C code equivalent to the code path at run-time. The CC backend has greater potential for big optimizations, but only a few optimizations are implemented currently. The Bytecode backend generates a platform independent bytecode representation of the interpreter's state just before execution. Thus, the Bytecode back end also eliminates much of the compilation overhead of the interpreter.

The compiler comes with several valuable utilities.

`B::Lint` is an experimental module to detect and warn about suspicious code, especially the cases that the `-w` switch does not detect.

`B::Deparse` can be used to demystify perl code, and understand how perl optimizes certain constructs.

`B::Xref` generates cross reference reports of all definition and use of variables, subroutines and formats in a program.

`B::Showlex` show the lexical variables used by a subroutine or file at a glance.

`perlcc` is a simple frontend for compiling perl.

See `ext/B/README`, B, and the respective compiler modules.

### Regular Expressions

Perl's regular expression engine has been seriously overhauled, and many new constructs are supported. Several bugs have been fixed.

Here is an itemized summary:

Many new and improved optimizations

Changes in the RE engine:

```
Unneeded nodes removed;
Substrings merged together;
New types of nodes to process (SUBEXPR)* and similar expressions
    quickly, used if the SUBEXPR has no side effects and matches
    strings of the same length;
Better optimizations by lookup for constant substrings;
Better search for constants substrings anchored by $ ;
```

Changes in Perl code using RE engine:

```
More optimizations to s/longer/short/;
study() was not working;
/blah/ may be optimized to an analogue of index() if $& $` $' not seen
Unneeded copying of matched-against string removed;
Only matched part of the string is copying if $` $' were not seen;
```

Many bug fixes

Note that only the major bug fixes are listed here. See *Changes* for others.

```
Backtracking might not restore start of $3.
No feedback if max count for * or + on "complex" subexpression
    was reached, similarly (but at compile time) for {3,34567}
Primitive restrictions on max count introduced to decrease a
    possibility of a segfault;
(ZERO-LENGTH)* could segfault;
(ZERO-LENGTH)* was prohibited;
Long REs were not allowed;
/RE/g could skip matches at the same position after a
  zero-length match;
```

New regular expression constructs

The following new syntax elements are supported:

```
(?<=RE)
(?<!RE)
(?{ CODE })
(?i-x)
(?i:RE)
(?(COND)YES_RE|NO_RE)
(?>RE)
\z
```

New operator for precompiled regular expressions

See "New `qr//` operator".

Other improvements

```
Better debugging output (possibly with colors),
    even from non-debugging Perl;
RE engine code now looks like C, not like assembler;
Behaviour of RE modifiable by `use re' directive;
Improved documentation;
Test suite significantly extended;
Syntax [:^upper:] etc., reserved inside character classes;
```

Incompatible changes

```
(?i) localized inside enclosing group;
$( is not interpolated into RE any more;
/RE/g may match at the same position (with non-zero length)
    after a zero-length match (bug fix).
```

See perlre and perlop.

### Improved malloc()

See banner at the beginning of `malloc.c` for details.

### Quicksort is internally implemented

Perl now contains its own highly optimized **qsort()** routine. The new **qsort()** is resistant to inconsistent comparison functions, so Perl's `sort()` will not provoke coredumps any more when given poorly written sort subroutines. (Some C library `qsort()`s that were being used before used to have this problem.) In our testing, the new `qsort()` required the minimal number of pair-wise compares on average, among all known `qsort()` implementations.

See `perlfunc/sort`.

### Reliable signals

Perl's signal handling is susceptible to random crashes, because signals arrive asynchronously, and the Perl runtime is not reentrant at arbitrary times.

However, one experimental implementation of reliable signals is available when threads are enabled. See `Thread::Signal`. Also see *INSTALL* for how to build a Perl capable of threads.

### Reliable stack pointers

The internals now reallocate the perl stack only at predictable times. In particular, magic calls never trigger reallocations of the stack, because all reentrancy of the runtime is handled using a ''stack of stacks''. This should improve reliability of cached stack pointers in the internals and in XSUBs.

### More generous treatment of carriage returns

Perl used to complain if it encountered literal carriage returns in scripts. Now they are mostly treated like whitespace within program text. Inside string literals and here documents, literal carriage returns are ignored if they occur paired with linefeeds, or get interpreted as whitespace if they stand alone. This behavior means that literal carriage returns in files should be avoided. You can get the older, more compatible (but less generous) behavior by defining the preprocessor symbol `PERL_STRICT_CR` when building perl. Of course, all this has nothing whatever to do with how escapes like `\r` are handled within strings.

Note that this doesn't somehow magically allow you to keep all text files in DOS format. The generous treatment only applies to files that perl itself parses. If your C compiler doesn't allow carriage returns in files, you may still be unable to build modules that need a C compiler.

### Memory leaks

`substr`, `pos` and `vec` don't leak memory anymore when used in lvalue context. Many small leaks that impacted applications that embed multiple interpreters have been fixed.

### Better support for multiple interpreters

The build-time option `-DMULTIPLICITY` has had many of the details reworked. Some previously global variables that should have been per-interpreter now are. With care, this allows interpreters to call each other. See the `PerlInterp` extension on CPAN.

### Behavior of local() on array and hash elements is now well-defined

See ''Temporary Values via **local()**'' in perlsub.

**`%!` is transparently tied to the Errno module**
See perlvar, and Errno.

**Pseudo-hashes are supported**
See perlref.

**`EXPR foreach EXPR` is supported**
See perlsyn.

**Keywords can be globally overridden**
See perlsub.

**`$^E` is meaningful on Win32**
See perlvar.

**`foreach (1..1000000)` optimized**
`foreach (1..1000000)` is now optimized into a counting loop. It does not try to allocate a 1000000−size list anymore.

**`Foo::` can be used as implicitly quoted package name**
Barewords caused unintuitive behavior when a subroutine with the same name as a package happened to be defined. Thus, `new Foo @args`, use the result of the call to `Foo()` instead of `Foo` being treated as a literal. The recommended way to write barewords in the indirect object slot is `new Foo:: @args`. Note that the method `new()` is called with a first argument of `Foo`, not `Foo::` when you do that.

**`exists $Foo::{Bar::}` tests existence of a package**
It was impossible to test for the existence of a package without actually creating it before. Now `exists $Foo::{Bar::}` can be used to test if the `Foo::Bar` namespace has been created.

**Better locale support**
See perllocale.

**Experimental support for 64−bit platforms**
Perl5 has always had 64−bit support on systems with 64−bit longs. Starting with 5.005, the beginnings of experimental support for systems with 32−bit long and 64−bit 'long long' integers has been added. If you add −DUSE_LONG_LONG to your ccflags in config.sh (or manually define it in perl.h) then perl will be built with 'long long' support. There will be many compiler warnings, and the resultant perl may not work on all systems. There are many other issues related to third-party extensions and libraries. This option exists to allow people to work on those issues.

**prototype() returns useful results on builtins**
See ''prototype'' in perlfunc.

**Extended support for exception handling**
`die()` now accepts a reference value, and `$@` gets set to that value in exception traps. This makes it possible to propagate exception objects. This is an undocumented **experimental** feature.

**Re-blessing in DESTROY() supported for chaining DESTROY() methods**
See ''Destructors'' in perlobj.

**All `printf` format conversions are handled internally**
See ''printf'' in perlfunc.

**New `INIT` keyword**
`INIT` subs are like `BEGIN` and `END`, but they get run just before the perl runtime begins execution. e.g., the Perl Compiler makes use of `INIT` blocks to initialize and resolve pointers to XSUBs.

**New `lock` keyword**
The `lock` keyword is the fundamental synchronization primitive in threaded perl. When threads are not enabled, it is currently a noop.

To minimize impact on source compatibility this keyword is ''weak'', i.e., any user-defined subroutine of the same name overrides it, unless a `use Thread` has been seen.

**New `qr//` operator**
The `qr//` operator, which is syntactically similar to the other quote-like operators, is used to create precompiled regular expressions. This compiled form can now be explicitly passed around in variables, and interpolated in other regular expressions. See perlop.

### `our` is now a reserved word

Calling a subroutine with the name `our` will now provoke a warning when using the `-w` switch.

### Tied arrays are now fully supported

See Tie::Array.

### Tied handles support is better

Several missing hooks have been added. There is also a new base class for TIEARRAY implementations. See Tie::Array.

### 4th argument to substr

**substr()** can now both return and replace in one operation. The optional 4th argument is the replacement string. See ‘‘substr’’ in perlfunc.

### Negative LENGTH argument to splice

**splice()** with a negative LENGTH argument now work similar to what the LENGTH did for **substr()**. Previously a negative LENGTH was treated as 0. See ‘‘splice’’ in perlfunc.

### Magic lvalues are now more magical

When you say something like `substr($x, 5) = "hi"`, the scalar returned by **substr()** is special, in that any modifications to it affect `$x`. (This is called a ’magic lvalue’ because an ’lvalue’ is something on the left side of an assignment.) Normally, this is exactly what you would expect to happen, but Perl uses the same magic if you use **substr()**, **pos()**, or **vec()** in a context where they might be modified, like taking a reference with `\` or as an argument to a sub that modifies `@_`. In previous versions, this ’magic’ only went one way, but now changes to the scalar the magic refers to ($x in the above example) affect the magic lvalue too. For instance, this code now acts differently:

```
$x = "hello";
sub printit {
    $x = "g'bye";
    print $_[0], "\n";
}
printit(substr($x, 0, 5));
```

In previous versions, this would print ‘‘hello’’, but it now prints ‘‘g’bye’’.

### <> now reads in records

If `$/` is a reference to an integer, or a scalar that holds an integer, <> will read in records instead of lines. For more info, see ‘‘$/’’ in perlvar.

## Supported Platforms

Configure has many incremental improvements. Site-wide policy for building perl can now be made persistent, via Policy.sh. Configure also records the command-line arguments used in *config.sh*.

### New Platforms

BeOS is now supported. See *README.beos*.

DOS is now supported under the DJGPP tools. See *README.dos* (installed as perldos on some systems).

MiNT is now supported. See *README.mint*.

MPE/iX is now supported. See README.mpeix.

MVS (aka OS390, aka Open Edition) is now supported. See *README.os390* (installed as perlos390 on some systems).

Stratus VOS is now supported. See *README.vos*.

### Changes in existing support

Win32 support has been vastly enhanced. Support for Perl Object, a C++ encapsulation of Perl. GCC and EGCS are now supported on Win32. See *README.win32*, aka perlwin32.

VMS configuration system has been rewritten. See *README.vms* (installed as *README_vms* on some systems).

The hints files for most Unix platforms have seen incremental improvements.

## Modules and Pragmata

### New Modules

B Perl compiler and tools. See B.

Data::Dumper
: A module to pretty print Perl data. See Data::Dumper.

Dumpvalue
: A module to dump perl values to the screen. See Dumpvalue.

Errno
: A module to look up errors more conveniently. See Errno.

File::Spec
: A portable API for file operations.

ExtUtils::Installed
: Query and manage installed modules.

ExtUtils::Packlist
: Manipulate .packlist files.

Fatal
: Make functions/builtins succeed or die.

IPC::SysV
: Constants and other support infrastructure for System V IPC operations in perl.

Test
: A framework for writing test suites.

Tie::Array
: Base class for tied arrays.

Tie::Handle
: Base class for tied handles.

Thread
: Perl thread creation, manipulation, and support.

attrs
: Set subroutine attributes.

fields
: Compile-time class fields.

re Various pragmata to control behavior of regular expressions.

### Changes in existing modules

Benchmark
: You can now run tests for *x* seconds instead of guessing the right number of tests to run.

  Keeps better time.

Carp
: Carp has a new function **cluck()**. **cluck()** warns, like **carp()**, but also adds a stack backtrace to the error message, like **confess()**.

CGI
: CGI has been updated to version 2.42.

Fcntl
: More Fcntl constants added: F_SETLK64, F_SETLKW64, O_LARGEFILE for large (more than 4G) file access (the 64−bit support is not yet working, though, so no need to get overly excited), Free/Net/OpenBSD locking behaviour flags F_FLOCK, F_POSIX, Linux F_SHLCK, and O_ACCMODE: the mask of O_RDONLY, O_WRONLY, and O_RDWR.

Math::Complex
: The accessors methods Re, Im, arg, abs, rho, theta, methods can ($z−>**Re()**) now also act as mutators ($z−>**Re**(3)).

Math::Trig
: A little bit of radial trigonometry (cylindrical and spherical) added, for example the great circle distance.

POSIX
: POSIX now has its own platform-specific hints files.

DB_File
: DB_File supports version 2.x of Berkeley DB. See `ext/DB_File/Changes`.

MakeMaker
: MakeMaker now supports writing empty makefiles, provides a way to specify that site **umask()** policy should be honored. There is also better support for manipulation of .packlist files, and getting information about installed modules.

  Extensions that have both architecture-dependent and architecture-independent files are now always installed completely in the architecture-dependent locations. Previously, the shareable parts were shared both across architectures and across perl versions and were therefore liable to be overwritten with newer versions that might have subtle incompatibilities.

CPAN
: See perlmodinstall and CPAN.

Cwd
: Cwd::cwd is faster on most platforms.

## Utility Changes

`h2ph` and related utilities have been vastly overhauled.

`perlcc`, a new experimental front end for the compiler is available.

The crude GNU `configure` emulator is now called `configure.gnu` to avoid trampling on `Configure` under case-insensitive filesystems.

`perldoc` used to be rather slow. The slower features are now optional. In particular, case-insensitive searches need the `-i` switch, and recursive searches need `-r`. You can set these switches in the `PERLDOC` environment variable to get the old behavior.

## Documentation Changes

Config.pm now has a glossary of variables.

*Porting/patching.pod* has detailed instructions on how to create and submit patches for perl.

perlport specifies guidelines on how to write portably.

perlmodinstall describes how to fetch and install modules from `CPAN` sites.

Some more Perl traps are documented now. See perltrap.

perlopentut gives a tutorial on using **open()**.

perlreftut gives a tutorial on references.

perlthrtut gives a tutorial on threads.

## New Diagnostics

Ambiguous call resolved as CORE::%s(), qualify as such or use &
: (W) A subroutine you have declared has the same name as a Perl keyword, and you have used the name without qualification for calling one or the other. Perl decided to call the builtin because the subroutine is not imported.

  To force interpretation as a subroutine call, either put an ampersand before the subroutine name, or qualify the name with its package. Alternatively, you can import the subroutine (or pretend that it's imported with the `use subs` pragma).

  To silently interpret it as the Perl operator, use the `CORE::` prefix on the operator (e.g. `CORE::log($x)`) or by declaring the subroutine to be an object method (see ‘‘attrs’’).

Bad index while coercing array into hash
: (F) The index looked up in the hash found as the 0'th element of a pseudo-hash is not legal. Index values must be at 1 or greater. See perlref.

Bareword ‘‘%s’’ refers to nonexistent package
(W) You used a qualified bareword of the form `Foo::`, but the compiler saw no other uses of that namespace before that point. Perhaps you need to predeclare a package?

Can’t call method ‘‘%s’’ on an undefined value
(F) You used the syntax of a method call, but the slot filled by the object reference or package name contains an undefined value. Something like this will reproduce the error:

```
$BADREF = 42;
process $BADREF 1,2,3;
$BADREF->process(1,2,3);
```

Can’t check filesystem of script ‘‘%s’’ for nosuid
(P) For some reason you can’t check the filesystem of the script for nosuid.

Can’t coerce array into hash
(F) You used an array where a hash was expected, but the array has no information on how to map from keys to array indices. You can do that only with arrays that have a hash reference at index 0.

Can’t goto subroutine from an eval-string
(F) The ‘‘goto subroutine’’ call can’t be used to jump out of an eval ‘‘string’’. (You can use it to jump out of an eval {BLOCK}, but you probably don’t want to.)

Can’t localize pseudo-hash element
(F) You said something like `local $ar->{'key'}`, where `$ar` is a reference to a pseudo-hash. That hasn’t been implemented yet, but you can get a similar effect by localizing the corresponding array element directly: `local $ar->[$ar->[0]{'key'}]`.

Can’t use %%! because Errno.pm is not available
(F) The first time the %! hash is used, perl automatically loads the Errno.pm module. The Errno module is expected to tie the %! hash to provide symbolic names for `$!` errno values.

Cannot find an opnumber for ‘‘%s’’
(F) A string of a form `CORE::word` was given to **prototype()**, but there is no builtin with the name `word`.

Character class syntax [. .] is reserved for future extensions
(W) Within regular expression character classes ([]) the syntax beginning with ‘‘[.’’ and ending with ‘‘.]’’ is reserved for future extensions. If you need to represent those character sequences inside a regular expression character class, just quote the square brackets with the backslash: ‘‘\[.’’ and ‘‘.\]’’.

Character class syntax [: :] is reserved for future extensions
(W) Within regular expression character classes ([]) the syntax beginning with ‘‘[:’’ and ending with ‘‘:]’’ is reserved for future extensions. If you need to represent those character sequences inside a regular expression character class, just quote the square brackets with the backslash: ‘‘\[:’’ and ‘‘:\]’’.

Character class syntax [= =] is reserved for future extensions
(W) Within regular expression character classes ([]) the syntax beginning with ‘‘[=’’ and ending with ‘‘=]’’ is reserved for future extensions. If you need to represent those character sequences inside a regular expression character class, just quote the square brackets with the backslash: ‘‘\[=’’ and ‘‘=\]’’.

`%s`: Eval-group in insecure regular expression
(F) Perl detected tainted data when trying to compile a regular expression that contains the `(?{ ... })` zero-width assertion, which is unsafe. See ‘‘(?{ code })’’ in perlre, and perlsec.

`%s`: Eval-group not allowed, use re ’eval’
(F) A regular expression contained the `(?{ ... })` zero-width assertion, but that construct is only allowed when the `use re 'eval'` pragma is in effect. See ‘‘(?{ code })’’ in perlre.

`%s`: Eval-group not allowed at run time
(F) Perl tried to compile a regular expression containing the `(?{ ... })` zero-width assertion at run time, as it would when the pattern contains interpolated values. Since that is a security risk, it is not allowed. If you insist, you may still do this by explicitly building the pattern from an interpolated string at run time and using that in an **eval()**. See ‘‘(?{ code })’’ in perlre.

Explicit blessing to '' (assuming package main)

(W) You are blessing a reference to a zero length string. This has the effect of blessing the reference into the package main. This is usually not what you want. Consider providing a default target package, e.g. bless($ref, $p || 'MyPackage');

Illegal hex digit ignored

(W) You may have tried to use a character other than 0 − 9 or A − F in a hexadecimal number. Interpretation of the hexadecimal number stopped before the illegal character.

No such array field

(F) You tried to access an array as a hash, but the field name used is not defined. The hash at index 0 should map all valid field names to array indices for that to work.

No such field "%s" in variable %s of type %s

(F) You tried to access a field of a typed variable where the type does not know about the field name. The field names are looked up in the %FIELDS hash in the type package at compile time. The %FIELDS hash is usually set up with the 'fields' pragma.

Out of memory during ridiculously large request

(F) You can't allocate more than 2^31+"small amount" bytes. This error is most likely to be caused by a typo in the Perl program. e.g., $arr[time] instead of $arr[$time].

Range iterator outside integer range

(F) One (or both) of the numeric arguments to the range operator ".." are outside the range which can be represented by integers internally. One possible workaround is to force Perl to use magical string increment by prepending "0" to your numbers.

Recursive inheritance detected while looking for method '%s' %s

(F) More than 100 levels of inheritance were encountered while invoking a method. Probably indicates an unintended loop in your inheritance hierarchy.

Reference found where even-sized list expected

(W) You gave a single reference where Perl was expecting a list with an even number of elements (for assignment to a hash). This usually means that you used the anon hash constructor when you meant to use parens. In any case, a hash requires key/value **pairs**.

```
%hash = { one => 1, two => 2, };    # WRONG
%hash = [ qw/ an anon array / ];    # WRONG
%hash = ( one => 1, two => 2, );    # right
%hash = qw( one 1 two 2 );                  # also fine
```

Undefined value assigned to typeglob

(W) An undefined value was assigned to a typeglob, a la *foo = undef. This does nothing. It's possible that you really mean undef *foo.

Use of reserved word "%s" is deprecated

(D) The indicated bareword is a reserved word. Future versions of perl may use it as a keyword, so you're better off either explicitly quoting the word in a manner appropriate for its context of use, or using a different name altogether. The warning can be suppressed for subroutine names by either adding a & prefix, or using a package qualifier, e.g. &our(), or Foo::our().

perl: warning: Setting locale failed.

(S) The whole warning message will look something like:

```
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LC_ALL = "En_US",
        LANG = (unset)
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
```

Exactly what were the failed locale settings varies. In the above the settings were that the LC_ALL was "En_US" and the LANG had no value. This error means that Perl detected that you and/or your system administrator have set up the so-called variable system but Perl could not use those settings. This was not dead serious, fortunately: there is a "default locale" called "C" that Perl can and will use, the script will be run. Before you really fix the problem, however, you will get

the same error message each time you run Perl. How to really fix the problem can be found in ''LOCALE PROBLEMS'' in perllocale.

## Obsolete Diagnostics

Can't **mktemp()**

(F) The **mktemp()** routine failed for some reason while trying to process a **−e** switch. Maybe your /tmp partition is full, or clobbered.

Removed because **−e** doesn't use temporary files any more.

Can't write to temp file for **−e**: `%s`

(F) The write routine failed for some reason while trying to process a **−e** switch. Maybe your /tmp partition is full, or clobbered.

Removed because **−e** doesn't use temporary files any more.

Cannot open temporary file

(F) The create routine failed for some reason while trying to process a **−e** switch. Maybe your /tmp partition is full, or clobbered.

Removed because **−e** doesn't use temporary files any more.

regexp too big

(F) The current implementation of regular expressions uses shorts as address offsets within a string. Unfortunately this means that if the regular expression compiles to longer than 32767, it'll blow up. Usually when you want a regular expression this big, there is a better way to do it with multiple statements. See perlre.

## Configuration Changes

You can use ''Configure −Uinstallusrbinperl'' which causes installperl to skip installing perl also as /usr/bin/perl. This is useful if you prefer not to modify /usr/bin for some reason or another but harmful because many scripts assume to find Perl in /usr/bin/perl.

## BUGS

If you find what you think is a bug, you might check the headers of recently posted articles in the comp.lang.perl.misc newsgroup. There may also be information at http://www.perl.com/perl/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Make sure you trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to <*perlbug@perl.com*> to be analysed by the Perl porting team.

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## HISTORY

Written by Gurusamy Sarathy <*gsar@activestate.com*>, with many contributions from The Perl Porters.

Send omissions or corrections to <*perlbug@perl.com*>.

## NAME

perl5100delta − what is new for perl 5.10.0

## DESCRIPTION

This document describes the differences between the 5.8.8 release and the 5.10.0 release.

Many of the bug fixes in 5.10.0 were already seen in the 5.8.X maintenance releases; they are not duplicated here and are documented in the set of man pages named perl58[1−8]?delta.

## Core Enhancements

### The `feature` pragma

The `feature` pragma is used to enable new syntax that would break Perl's backwards-compatibility with older releases of the language. It's a lexical pragma, like `strict` or `warnings`.

Currently the following new features are available: `switch` (adds a switch statement), `say` (adds a `say` built-in function), and `state` (adds a `state` keyword for declaring ''static'' variables). Those features are described in their own sections of this document.

The `feature` pragma is also implicitly loaded when you require a minimal perl version (with the `use VERSION` construct) greater than, or equal to, 5.9.5. See feature for details.

### New −E command-line switch

**−E** is equivalent to **−e**, but it implicitly enables all optional features (like `use feature ":5.10"`).

### Defined-or operator

A new operator `//` (defined-or) has been implemented. The following expression:

```
$a // $b
```

is merely equivalent to

```
defined $a ? $a : $b
```

and the statement

```
$c //= $d;
```

can now be used instead of

```
$c = $d unless defined $c;
```

The `//` operator has the same precedence and associativity as `||`. Special care has been taken to ensure that this operator Do What You Mean while not breaking old code, but some edge cases involving the empty regular expression may now parse differently. See perlop for details.

### Switch and Smart Match operator

Perl 5 now has a switch statement. It's available when `use feature 'switch'` is in effect. This feature introduces three new keywords, `given`, `when`, and `default`:

```
given ($foo) {
    when (/^abc/) { $abc = 1; }
    when (/^def/) { $def = 1; }
    when (/^xyz/) { $xyz = 1; }
    default { $nothing = 1; }
}
```

A more complete description of how Perl matches the switch variable against the `when` conditions is given in ''Switch statements'' in perlsyn.

This kind of match is called *smart match*, and it's also possible to use it outside of switch statements, via the new `~~` operator. See ''Smart matching in detail'' in perlsyn.

This feature was contributed by Robin Houston.

### Regular expressions

Recursive Patterns

It is now possible to write recursive patterns without using the `(??{})` construct. This new way is more efficient, and in many cases easier to read.

Each capturing parenthesis can now be treated as an independent pattern that can be entered by using the `(?PARNO)` syntax (`PARNO` standing for ''parenthesis number''). For example, the following pattern will match nested balanced angle brackets:

```
/
 ^                    # start of line
 (                    # start capture buffer 1
    <                 #   match an opening angle bracket
    (?:               #   match one of:
        (?>           #     don't backtrack over the inside of this grou
            [^<>]+    #       one or more non angle brackets
        )             #     end non backtracking group
    |                 #     ... or ...
        (?1)          #     recurse to bracket 1 and try it again
    )*                #   0 or more times.
    >                 #   match a closing angle bracket
 )                    # end capture buffer one
 $                    # end of line
/x
```

PCRE users should note that Perl's recursive regex feature allows backtracking into a recursed pattern, whereas in PCRE the recursion is atomic or ''possessive'' in nature. As in the example above, you can add (?>) to control this selectively. (Yves Orton)

Named Capture Buffers

It is now possible to name capturing parenthesis in a pattern and refer to the captured contents by name. The naming syntax is `(?<NAME>....)`. It's possible to backreference to a named buffer with the `\k<NAME>` syntax. In code, the new magical hashes `%+` and `%-` can be used to access the contents of the capture buffers.

Thus, to replace all doubled chars with a single copy, one could write

```
s/(?<letter>.)\k<letter>/$+{letter}/g
```

Only buffers with defined contents will be ''visible'' in the `%+` hash, so it's possible to do something like

```
foreach my $name (keys %+) {
    print "content of buffer '$name' is $+{$name}\n";
}
```

The `%-` hash is a bit more complete, since it will contain array refs holding values from all capture buffers similarly named, if there should be many of them.

`%+` and `%-` are implemented as tied hashes through the new module `Tie::Hash::NamedCapture`.

Users exposed to the .NET regex engine will find that the perl implementation differs in that the numerical ordering of the buffers is sequential, and not ''unnamed first, then named''. Thus in the pattern

```
/(A)(?<B>B)(C)(?<D>D)/
```

`$1` will be 'A', `$2` will be 'B', `$3` will be 'C' and `$4` will be 'D' and not `$1` is 'A', `$2` is 'C' and `$3` is 'B' and `$4` is 'D' that a .NET programmer would expect. This is considered a feature. :-) (Yves Orton)

Possessive Quantifiers

Perl now supports the ''possessive quantifier'' syntax of the ''atomic match'' pattern. Basically a possessive quantifier matches as much as it can and never gives any back. Thus it can be used to control backtracking. The syntax is similar to non-greedy matching, except instead of using a '?' as the modifier the '+' is used. Thus `?+`, `*+`, `++`, `{min,max}+` are now legal quantifiers. (Yves Orton)

Backtracking control verbs

The regex engine now supports a number of special-purpose backtrack control verbs: (*THEN), (*PRUNE), (*MARK), (*SKIP), (*COMMIT), (*FAIL) and (*ACCEPT). See perlre for their descriptions. (Yves Orton)

Relative backreferences
: A new syntax `\g{N}` or `\gN` where "N" is a decimal integer allows a safer form of back-reference notation as well as allowing relative backreferences. This should make it easier to generate and embed patterns that contain backreferences. See "Capture buffers" in perlre. (Yves Orton)

`\K` escape
: The functionality of Jeff Pinyan's module Regexp::Keep has been added to the core. In regular expressions you can now use the special escape `\K` as a way to do something like floating length positive lookbehind. It is also useful in substitutions like:

```
s/(foo)bar/$1/g
```

that can now be converted to

```
s/foo\Kbar//g
```

which is much more efficient. (Yves Orton)

Vertical and horizontal whitespace, and linebreak
: Regular expressions now recognize the `\v` and `\h` escapes that match vertical and horizontal whitespace, respectively. `\V` and `\H` logically match their complements.

`\R` matches a generic linebreak, that is, vertical whitespace, plus the multi-character sequence `"\x0D\x0A"`.

Optional pre-match and post-match captures with the /p flag
: There is a new flag `/p` for regular expressions. Using this makes the engine preserve a copy of the part of the matched string before the matching substring to the new special variable `${^PREMATCH}`, the part after the matching substring to `${^POSTMATCH}`, and the matched substring itself to `${^MATCH}`.

Perl is still able to store these substrings to the special variables `$``, `$'`, `$&`, but using these variables anywhere in the program adds a penalty to all regular expression matches, whereas if you use the `/p` flag and the new special variables instead, you pay only for the regular expressions where the flag is used.

For more detail on the new variables, see perlvar; for the use of the regular expression flag, see perlop and perlre.

`say()`
: **say()** is a new built-in, only available when `use feature 'say'` is in effect, that is similar to **print()**, but that implicitly appends a newline to the printed string. See "say" in perlfunc. (Robin Houston)

**Lexical** `$_`
: The default variable `$_` can now be lexicalized, by declaring it like any other lexical variable, with a simple

```
my $_;
```

The operations that default on `$_` will use the lexically-scoped version of `$_` when it exists, instead of the global `$_`.

In a `map` or a `grep` block, if `$_` was previously my'ed, then the `$_` inside the block is lexical as well (and scoped to the block).

In a scope where `$_` has been lexicalized, you can still have access to the global version of `$_` by using `$::_`, or, more simply, by overriding the lexical declaration with `our $_`. (Rafael Garcia-Suarez)

**The _ prototype**
: A new prototype character has been added. _ is equivalent to `$` but defaults to `$_` if the corresponding argument isn't supplied (both `$` and _ denote a scalar). Due to the optional nature of the argument, you can only use it at the end of a prototype, or before a semicolon.

This has a small incompatible consequence: the **prototype()** function has been adjusted to return _ for some built-ins in appropriate cases (for example, `prototype('CORE::rmdir')`). (Rafael Garcia-Suarez)

### UNITCHECK blocks

UNITCHECK, a new special code block has been introduced, in addition to BEGIN, CHECK, INIT and END.

CHECK and INIT blocks, while useful for some specialized purposes, are always executed at the transition between the compilation and the execution of the main program, and thus are useless whenever code is loaded at runtime. On the other hand, UNITCHECK blocks are executed just after the unit which defined them has been compiled. See perlmod for more information. (Alex Gough)

### New Pragma, mro

A new pragma, mro (for Method Resolution Order) has been added. It permits to switch, on a per-class basis, the algorithm that perl uses to find inherited methods in case of a multiple inheritance hierarchy. The default MRO hasn't changed (DFS, for Depth First Search). Another MRO is available: the C3 algorithm. See mro for more information. (Brandon Black)

Note that, due to changes in the implementation of class hierarchy search, code that used to undef the *ISA glob will most probably break. Anyway, undef'ing *ISA had the side-effect of removing the magic on the @ISA array and should not have been done in the first place. Also, the cache *::ISA::CACHE:: no longer exists; to force reset the @ISA cache, you now need to use the mro API, or more simply to assign to @ISA (e.g. with @ISA = @ISA).

### readdir() may return a "short filename" on Windows

The **readdir()** function may return a ''short filename'' when the long filename contains characters outside the ANSI codepage. Similarly **Cwd::cwd()** may return a short directory name, and **glob()** may return short names as well. On the NTFS file system these short names can always be represented in the ANSI codepage. This will not be true for all other file system drivers; e.g. the FAT filesystem stores short filenames in the OEM codepage, so some files on FAT volumes remain inaccessible through the ANSI APIs.

Similarly, $^X, @INC, and $ENV{PATH} are preprocessed at startup to make sure all paths are valid in the ANSI codepage (if possible).

The **Win32::GetLongPathName()** function now returns the UTF-8 encoded correct long file name instead of using replacement characters to force the name into the ANSI codepage. The new **Win32::GetANSIPathName()** function can be used to turn a long pathname into a short one only if the long one cannot be represented in the ANSI codepage.

Many other functions in the Win32 module have been improved to accept UTF-8 encoded arguments. Please see Win32 for details.

### readpipe() is now overridable

The built-in function **readpipe()** is now overridable. Overriding it permits also to override its operator counterpart, qx// (a.k.a. ``). Moreover, it now defaults to $_ if no argument is provided. (Rafael Garcia-Suarez)

### Default argument for readline()

**readline()** now defaults to *ARGV if no argument is provided. (Rafael Garcia-Suarez)

### state() variables

A new class of variables has been introduced. State variables are similar to my variables, but are declared with the state keyword in place of my. They're visible only in their lexical scope, but their value is persistent: unlike my variables, they're not undefined at scope entry, but retain their previous value. (Rafael Garcia-Suarez, Nicholas Clark)

To use state variables, one needs to enable them by using

```
use feature 'state';
```

or by using the -E command-line switch in one-liners. See ''Persistent Private Variables'' in perlsub.

### Stacked filetest operators

As a new form of syntactic sugar, it's now possible to stack up filetest operators. You can now write -f -w -x $file in a row to mean -x $file && -w _ && -f _. See ''-X'' in perlfunc.

### UNIVERSAL::DOES()

The UNIVERSAL class has a new method, DOES(). It has been added to solve semantic problems with the isa() method. isa() checks for inheritance, while DOES() has been designed to be overridden when module authors use other types of relations between classes (in addition to

inheritance). (chromatic)

See ‘‘$obj->DOES( ROLE )’’ in UNIVERSAL.

### Formats

Formats were improved in several ways. A new field, `^*`, can be used for variable-width, one-line-at-a-time text. Null characters are now handled correctly in picture lines. Using `@#` and `~~` together will now produce a compile-time error, as those format fields are incompatible. perlform has been improved, and miscellaneous bugs fixed.

### Byte-order modifiers for pack() and unpack()

There are two new byte-order modifiers, `>` (big-endian) and `<` (little-endian), that can be appended to most **pack()** and **unpack()** template characters and groups to force a certain byte-order for that type or group. See ‘‘pack’’ in perlfunc and perlpacktut for details.

### `no VERSION`

You can now use `no` followed by a version number to specify that you want to use a version of perl older than the specified one.

### `chdir`, `chmod` and `chown` on filehandles

`chdir`, `chmod` and `chown` can now work on filehandles as well as filenames, if the system supports respectively `fchdir`, `fchmod` and `fchown`, thanks to a patch provided by Gisle Aas.

### OS groups

`$(` and `$)` now return groups in the order where the OS returns them, thanks to Gisle Aas. This wasn’t previously the case.

### Recursive sort subs

You can now use recursive subroutines with **sort()**, thanks to Robin Houston.

### Exceptions in constant folding

The constant folding routine is now wrapped in an exception handler, and if folding throws an exception (such as attempting to evaluate 0/0), perl now retains the current optree, rather than aborting the whole program. Without this change, programs would not compile if they had expressions that happened to generate exceptions, even though those expressions were in code that could never be reached at runtime. (Nicholas Clark, Dave Mitchell)

### Source filters in `@INC`

It’s possible to enhance the mechanism of subroutine hooks in `@INC` by adding a source filter on top of the filehandle opened and returned by the hook. This feature was planned a long time ago, but wasn’t quite working until now. See ‘‘require’’ in perlfunc for details. (Nicholas Clark)

### New internal variables

`${^RE_DEBUG_FLAGS}`

This variable controls what debug flags are in effect for the regular expression engine when running under `use re "debug"`. See re for details.

`${^CHILD_ERROR_NATIVE}`

This variable gives the native status returned by the last pipe close, backtick command, successful call to **wait()** or **waitpid()**, or from the **system()** operator. See perlvar for details. (Contributed by Gisle Aas.)

`${^RE_TRIE_MAXBUF}`

See ‘‘Trie optimisation of literal string alternations’’.

`${^WIN32_SLOPPY_STAT}`

See ‘‘Sloppy stat on Windows’’.

### Miscellaneous

`unpack()` now defaults to unpacking the `$_` variable.

`mkdir()` without arguments now defaults to `$_`.

The internal dump output has been improved, so that non-printable characters such as newline and backspace are output in `\x` notation, rather than octal.

The **-C** option can no longer be used on the `#!` line. It wasn’t working there anyway, since the standard streams are already set up at this point in the execution of the perl interpreter. You can use **binmode()** instead to get the desired behaviour.

#### UCD 5.0.0

The copy of the Unicode Character Database included in Perl 5 has been updated to version 5.0.0.

#### MAD

MAD, which stands for *Miscellaneous Attribute Decoration*, is a still-in-development work leading to a Perl 5 to Perl 6 converter. To enable it, it's necessary to pass the argument `-Dmad` to Configure. The obtained perl isn't binary compatible with a regular perl 5.10, and has space and speed penalties; moreover not all regression tests still pass with it. (Larry Wall, Nicholas Clark)

#### kill() on Windows

On Windows platforms, `kill(-9, $pid)` now kills a process tree. (On Unix, this delivers the signal to all processes in the same process group.)

## Incompatible Changes

#### Packing and UTF-8 strings

The semantics of **pack()** and **unpack()** regarding UTF-8-encoded data has been changed. Processing is now by default character per character instead of byte per byte on the underlying encoding. Notably, code that used things like `pack("a*", $string)` to see through the encoding of string will now simply get back the original `$string`. Packed strings can also get upgraded during processing when you store upgraded characters. You can get the old behaviour by using `use bytes`.

To be consistent with **pack()**, the `C0` in **unpack()** templates indicates that the data is to be processed in character mode, i.e. character by character; on the contrary, `U0` in **unpack()** indicates UTF-8 mode, where the packed string is processed in its UTF-8-encoded Unicode form on a byte by byte basis. This is reversed with regard to perl 5.8.X, but now consistent between **pack()** and **unpack()**.

Moreover, `C0` and `U0` can also be used in **pack()** templates to specify respectively character and byte modes.

`C0` and `U0` in the middle of a pack or unpack format now switch to the specified encoding mode, honoring parens grouping. Previously, parens were ignored.

Also, there is a new **pack()** character format, `W`, which is intended to replace the old `C`. `C` is kept for unsigned chars coded as bytes in the strings internal representation. `W` represents unsigned (logical) character values, which can be greater than 255. It is therefore more robust when dealing with potentially UTF-8-encoded data (as `C` will wrap values outside the range 0..255, and not respect the string encoding).

In practice, that means that pack formats are now encoding-neutral, except `C`.

For consistency, `A` in **unpack()** format now trims all Unicode whitespace from the end of the string. Before perl 5.9.2, it used to strip only the classical ASCII space characters.

#### Byte/character count feature in unpack()

A new **unpack()** template character, `"."`, returns the number of bytes or characters (depending on the selected encoding mode, see above) read so far.

#### The $* and $# variables have been removed

`$*`, which was deprecated in favor of the `/s` and `/m` regexp modifiers, has been removed.

The deprecated $# variable (output format for numbers) has been removed.

Two new severe warnings, `$#/$* is no longer supported`, have been added.

#### substr() lvalues are no longer fixed-length

The lvalues returned by the three argument form of **substr()** used to be a ''fixed length window'' on the original string. In some cases this could cause surprising action at distance or other undefined behaviour. Now the length of the window adjusts itself to the length of the string assigned to it.

#### Parsing of -f _

The identifier _ is now forced to be a bareword after a filetest operator. This solves a number of misparsing issues when a global _ subroutine is defined.

#### :unique

The `:unique` attribute has been made a no-op, since its current implementation was fundamentally flawed and not threadsafe.

### Effect of pragmas in eval

The compile-time value of the `%^H` hint variable can now propagate into eval("")uated code. This makes it more useful to implement lexical pragmas.

As a side-effect of this, the overloaded-ness of constants now propagates into eval("").

### chdir FOO

A bareword argument to **chdir()** is now recognized as a file handle. Earlier releases interpreted the bareword as a directory name. (Gisle Aas)

### Handling of .pmc files

An old feature of perl was that before `require` or `use` look for a file with a *.pm* extension, they will first look for a similar filename with a *.pmc* extension. If this file is found, it will be loaded in place of any potentially existing file ending in a *.pm* extension.

Previously, *.pmc* files were loaded only if more recent than the matching *.pm* file. Starting with 5.9.4, they'll be always loaded if they exist.

### $^V is now a `version` object instead of a v-string

$^V can still be used with the `%vd` format in printf, but any character-level operations will now access the string representation of the `version` object and not the ordinals of a v-string. Expressions like `substr($^V, 0, 2)` or `split //, $^V` no longer work and must be rewritten.

### @- and @+ in patterns

The special arrays `@-` and `@+` are no longer interpolated in regular expressions. (Sadahiro Tomoyuki)

### `$AUTOLOAD` can now be tainted

If you call a subroutine by a tainted name, and if it defers to an AUTOLOAD function, then `$AUTOLOAD` will be (correctly) tainted. (Rick Delaney)

### Tainting and printf

When perl is run under taint mode, `printf()` and `sprintf()` will now reject any tainted format argument. (Rafael Garcia-Suarez)

### undef and signal handlers

Undefining or deleting a signal handler via `undef $SIG{FOO}` is now equivalent to setting it to `'DEFAULT'`. (Rafael Garcia-Suarez)

### strictures and dereferencing in defined()

`use strict 'refs'` was ignoring taking a hard reference in an argument to **defined()**, as in :

```
use strict 'refs';
my $x = 'foo';
if (defined $$x) {...}
```

This now correctly produces the run-time error `Can't use string as a SCALAR ref while "strict refs" in use`.

`defined @$foo` and `defined %$bar` are now also subject to `strict 'refs'` (that is, `$foo` and `$bar` shall be proper references there.) (`defined(@foo)` and `defined(%bar)` are discouraged constructs anyway.) (Nicholas Clark)

### `(?p{})` has been removed

The regular expression construct `(?p{})`, which was deprecated in perl 5.8, has been removed. Use `(??{})` instead. (Rafael Garcia-Suarez)

### Pseudo-hashes have been removed

Support for pseudo-hashes has been removed from Perl 5.9. (The `fields` pragma remains here, but uses an alternate implementation.)

### Removal of the bytecode compiler and of perlcc

`perlcc`, the byteloader and the supporting modules (B::C, B::CC, B::Bytecode, etc.) are no longer distributed with the perl sources. Those experimental tools have never worked reliably, and, due to the lack of volunteers to keep them in line with the perl interpreter developments, it was decided to remove them instead of shipping a broken version of those. The last version of those modules can be found with perl 5.9.4.

However the B compiler framework stays supported in the perl core, as with the more useful modules it has permitted (among others, B::Deparse and B::Concise).

### Removal of the JPL

The JPL (Java-Perl Lingo) has been removed from the perl sources tarball.

### Recursive inheritance detected earlier

Perl will now immediately throw an exception if you modify any package's `@ISA` in such a way that it would cause recursive inheritance.

Previously, the exception would not occur until Perl attempted to make use of the recursive inheritance while resolving a method or doing a `$foo->isa($bar)` lookup.

### warnings::enabled and warnings::warnif changed to favor users of modules

The behaviour in 5.10.x favors the person using the module; The behaviour in 5.8.x favors the module writer;

Assume the following code:

```
  main calls Foo::Bar::baz()
  Foo::Bar inherits from Foo::Base
  Foo::Bar::baz() calls Foo::Base::_bazbaz()
  Foo::Base::_bazbaz() calls: warnings::warnif('substr', 'some warning
message');
```

On 5.8.x, the code warns when Foo::Bar contains `use warnings;` It does not matter if Foo::Base or main have warnings enabled to disable the warning one has to modify Foo::Bar.

On 5.10.0 and newer, the code warns when main contains `use warnings;` It does not matter if Foo::Base or Foo::Bar have warnings enabled to disable the warning one has to modify main.

## Modules and Pragmata

### Upgrading individual core modules

Even more core modules are now also available separately through the CPAN. If you wish to update one of these modules, you don't need to wait for a new perl release. From within the cpan shell, running the 'r' command will report on modules with upgrades available. See `perldoc CPAN` for more information.

### Pragmata Changes

`feature`

The new pragma `feature` is used to enable new features that might break old code. See "The `feature` pragma" above.

`mro`

This new pragma enables to change the algorithm used to resolve inherited methods. See "New Pragma, `mro`" above.

Scoping of the `sort` pragma

The `sort` pragma is now lexically scoped. Its effect used to be global.

Scoping of `bignum`, `bigint`, `bigrat`

The three numeric pragmas `bignum`, `bigint` and `bigrat` are now lexically scoped. (Tels)

`base`

The `base` pragma now warns if a class tries to inherit from itself. (Curtis ''Ovid'' Poe)

`strict` and `warnings`

`strict` and `warnings` will now complain loudly if they are loaded via incorrect casing (as in `use Strict;`). (Johan Vromans)

`version`

The `version` module provides support for version objects.

`warnings`

The `warnings` pragma doesn't load `Carp` anymore. That means that code that used `Carp` routines without having loaded it at compile time might need to be adjusted; typically, the following (faulty) code won't work anymore, and will require parentheses to be added after the function name:

```
    use warnings;
    require Carp;
    Carp::confess 'argh';
```

`less`

`less` now does something useful (or at least it tries to). In fact, it has been turned into a lexical pragma. So, in your modules, you can now test whether your users have requested to use less CPU, or less memory, less magic, or maybe even less fat. See less for more. (Joshua ben Jore)

### New modules

- `encoding::warnings`, by Audrey Tang, is a module to emit warnings whenever an ASCII character string containing high-bit bytes is implicitly converted into UTF−8. It's a lexical pragma since Perl 5.9.4; on older perls, its effect is global.
- `Module::CoreList`, by Richard Clamp, is a small handy module that tells you what versions of core modules ship with any versions of Perl 5. It comes with a command-line frontend, `corelist`.
- `Math::BigInt::FastCalc` is an XS-enabled, and thus faster, version of `Math::BigInt::Calc`.
- `Compress::Zlib` is an interface to the zlib compression library. It comes with a bundled version of zlib, so having a working zlib is not a prerequisite to install it. It's used by `Archive::Tar` (see below).
- `IO::Zlib` is an `IO::`−style interface to `Compress::Zlib`.
- `Archive::Tar` is a module to manipulate `tar` archives.
- `Digest::SHA` is a module used to calculate many types of SHA digests, has been included for SHA support in the CPAN module.
- `ExtUtils::CBuilder` and `ExtUtils::ParseXS` have been added.
- `Hash::Util::FieldHash`, by Anno Siegel, has been added. This module provides support for *field hashes*: hashes that maintain an association of a reference with a value, in a thread-safe garbage-collected way. Such hashes are useful to implement inside-out objects.
- `Module::Build`, by Ken Williams, has been added. It's an alternative to `ExtUtils::MakeMaker` to build and install perl modules.
- `Module::Load`, by Jos Boumans, has been added. It provides a single interface to load Perl modules and *.pl* files.
- `Module::Loaded`, by Jos Boumans, has been added. It's used to mark modules as loaded or unloaded.
- `Package::Constants`, by Jos Boumans, has been added. It's a simple helper to list all constants declared in a given package.
- `Win32API::File`, by Tye McQueen, has been added (for Windows builds). This module provides low-level access to Win32 system API calls for files/dirs.
- `Locale::Maketext::Simple`, needed by CPANPLUS, is a simple wrapper around `Locale::Maketext::Lexicon`. Note that `Locale::Maketext::Lexicon` isn't included in the perl core; the behaviour of `Locale::Maketext::Simple` gracefully degrades when the later isn't present.
- `Params::Check` implements a generic input parsing/checking mechanism. It is used by CPANPLUS.
- `Term::UI` simplifies the task to ask questions at a terminal prompt.
- `Object::Accessor` provides an interface to create per-object accessors.
- `Module::Pluggable` is a simple framework to create modules that accept pluggable sub-modules.
- `Module::Load::Conditional` provides simple ways to query and possibly load installed modules.

- `Time::Piece` provides an object oriented interface to time functions, overriding the built-ins **localtime()** and **gmtime()**.
- `IPC::Cmd` helps to find and run external commands, possibly interactively.
- `File::Fetch` provide a simple generic file fetching mechanism.
- `Log::Message` and `Log::Message::Simple` are used by the log facility of `CPANPLUS`.
- `Archive::Extract` is a generic archive extraction mechanism for *.tar* (plain, gzipped or bzipped) or *.zip* files.
- `CPANPLUS` provides an API and a command-line tool to access the CPAN mirrors.
- `Pod::Escapes` provides utilities that are useful in decoding Pod E<...> sequences.
- `Pod::Simple` is now the backend for several of the Pod-related modules included with Perl.

### Selected Changes to Core Modules

`Attribute::Handlers`

`Attribute::Handlers` can now report the caller's file and line number. (David Feldman)

All interpreted attributes are now passed as array references. (Damian Conway)

`B::Lint`

`B::Lint` is now based on `Module::Pluggable`, and so can be extended with plugins. (Joshua ben Jore)

`B` It's now possible to access the lexical pragma hints (`%^H`) by using the method **B::COP::hints_hash()**. It returns a `B::RHE` object, which in turn can be used to get a hash reference via the method **B::RHE::HASH()**. (Joshua ben Jore)

`Thread`

As the old 5005thread threading model has been removed, in favor of the ithreads scheme, the `Thread` module is now a compatibility wrapper, to be used in old code only. It has been removed from the default list of dynamic extensions.

## Utility Changes

perl −d

The Perl debugger can now save all debugger commands for sourcing later; notably, it can now emulate stepping backwards, by restarting and rerunning all bar the last command from a saved command history.

It can also display the parent inheritance tree of a given class, with the `i` command.

ptar

`ptar` is a pure perl implementation of `tar` that comes with `Archive::Tar`.

ptardiff

`ptardiff` is a small utility used to generate a diff between the contents of a tar archive and a directory tree. Like `ptar`, it comes with `Archive::Tar`.

shasum

`shasum` is a command-line utility, used to print or to check SHA digests. It comes with the new `Digest::SHA` module.

corelist

The `corelist` utility is now installed with perl (see ''New modules'' above).

h2ph and h2xs

`h2ph` and `h2xs` have been made more robust with regard to ''modern'' C code.

`h2xs` implements a new option `--use-xsloader` to force use of `XSLoader` even in backwards compatible modules.

The handling of authors' names that had apostrophes has been fixed.

Any enums with negative values are now skipped.

perlivp

`perlivp` no longer checks for **.ph* files by default. Use the new `-a` option to run *all* tests.

find2perl
: `find2perl` now assumes `-print` as a default action. Previously, it needed to be specified explicitly.

  Several bugs have been fixed in `find2perl`, regarding `-exec` and `-eval`. Also the options `-path`, `-ipath` and `-iname` have been added.

config_data
: `config_data` is a new utility that comes with `Module::Build`. It provides a command-line interface to the configuration of Perl modules that use Module::Build's framework of configurability (that is, `*::ConfigData` modules that contain local configuration information for their parent modules.)

cpanp
: `cpanp`, the CPANPLUS shell, has been added. (`cpanp-run-perl`, a helper for CPANPLUS operation, has been added too, but isn't intended for direct use).

cpan2dist
: `cpan2dist` is a new utility that comes with CPANPLUS. It's a tool to create distributions (or packages) from CPAN modules.

pod2html
: The output of `pod2html` has been enhanced to be more customizable via CSS. Some formatting problems were also corrected. (Jari Aalto)

## New Documentation

The perlpragma manpage documents how to write one's own lexical pragmas in pure Perl (something that is possible starting with 5.9.4).

The new perlglossary manpage is a glossary of terms used in the Perl documentation, technical and otherwise, kindly provided by O'Reilly Media, Inc.

The perlreguts manpage, courtesy of Yves Orton, describes internals of the Perl regular expression engine.

The perlreapi manpage describes the interface to the perl interpreter used to write pluggable regular expression engines (by Ævar Arnfjörð Bjarmason).

The perlunitut manpage is a tutorial for programming with Unicode and string encodings in Perl, courtesy of Juerd Waalboer.

A new manual page, perlunifaq (the Perl Unicode FAQ), has been added (Juerd Waalboer).

The perlcommunity manpage gives a description of the Perl community on the Internet and in real life. (Edgar ''Trizor'' Bering)

The CORE manual page documents the `CORE::` namespace. (Tels)

The long-existing feature of `/(?{...})/` regexps setting `$_` and **pos()** is now documented.

## Performance Enhancements

### In-place sorting

Sorting arrays in place (`@a = sort @a`) is now optimized to avoid making a temporary copy of the array.

Likewise, `reverse sort ...` is now optimized to sort in reverse, avoiding the generation of a temporary intermediate list.

### Lexical array access

Access to elements of lexical arrays via a numeric constant between 0 and 255 is now faster. (This used to be only the case for global arrays.)

### XS-assisted SWASHGET

Some pure-perl code that perl was using to retrieve Unicode properties and transliteration mappings has been reimplemented in XS.

### Constant subroutines

The interpreter internals now support a far more memory efficient form of inlineable constants. Storing a reference to a constant value in a symbol table is equivalent to a full typeglob referencing a constant subroutine, but using about 400 bytes less memory. This proxy constant subroutine is automatically

upgraded to a real typeglob with subroutine if necessary. The approach taken is analogous to the existing space optimisation for subroutine stub declarations, which are stored as plain scalars in place of the full typeglob.

Several of the core modules have been converted to use this feature for their system dependent constants − as a result `use POSIX;` now takes about 200K less memory.

`PERL_DONT_CREATE_GVSV`

The new compilation flag `PERL_DONT_CREATE_GVSV`, introduced as an option in perl 5.8.8, is turned on by default in perl 5.9.3. It prevents perl from creating an empty scalar with every new typeglob. See perl589delta for details.

### Weak references are cheaper

Weak reference creation is now *O(1)* rather than *O(n)*, courtesy of Nicholas Clark. Weak reference deletion remains *O(n)*, but if deletion only happens at program exit, it may be skipped completely.

### sort() enhancements

Salvador Fandiño provided improvements to reduce the memory usage of `sort` and to speed up some cases.

### Memory optimisations

Several internal data structures (typeglobs, GVs, CVs, formats) have been restructured to use less memory. (Nicholas Clark)

### UTF−8 cache optimisation

The UTF−8 caching code is now more efficient, and used more often. (Nicholas Clark)

### Sloppy stat on Windows

On Windows, perl's **stat()** function normally opens the file to determine the link count and update attributes that may have been changed through hard links. Setting ${^WIN32_SLOPPY_STAT} to a true value speeds up **stat()** by not performing this operation. (Jan Dubois)

### Regular expressions optimisations

Engine de-recursivised

The regular expression engine is no longer recursive, meaning that patterns that used to overflow the stack will either die with useful explanations, or run to completion, which, since they were able to blow the stack before, will likely take a very long time to happen. If you were experiencing the occasional stack overflow (or segfault) and upgrade to discover that now perl apparently hangs instead, look for a degenerate regex. (Dave Mitchell)

Single char char-classes treated as literals

Classes of a single character are now treated the same as if the character had been used as a literal, meaning that code that uses char-classes as an escaping mechanism will see a speedup. (Yves Orton)

Trie optimisation of literal string alternations

Alternations, where possible, are optimised into more efficient matching structures. String literal alternations are merged into a trie and are matched simultaneously. This means that instead of O(N) time for matching N alternations at a given point, the new code performs in O(1) time. A new special variable, ${^RE_TRIE_MAXBUF}, has been added to fine-tune this optimization. (Yves Orton)

**Note:** Much code exists that works around perl's historic poor performance on alternations. Often the tricks used to do so will disable the new optimisations. Hopefully the utility modules used for this purpose will be educated about these new optimisations.

Aho-Corasick start-point optimisation

When a pattern starts with a trie-able alternation and there aren't better optimisations available, the regex engine will use Aho-Corasick matching to find the start point. (Yves Orton)

## Installation and Configuration Improvements

### Configuration improvements

`−Dusesitecustomize`

Run-time customization of `@INC` can be enabled by passing the `−Dusesitecustomize` flag to Configure. When enabled, this will make perl run *`$sitelibexp`/sitecustomize.pl* before anything else. This script can then be set up to add additional entries to `@INC`.

Relocatable installations
: There is now Configure support for creating a relocatable perl tree. If you Configure with `-Duserelocatableinc`, then the paths in `@INC` (and everything else in `%Config`) can be optionally located via the path of the perl executable.

  That means that, if the string `"../"` is found at the start of any path, it's substituted with the directory of $^X. So, the relocation can be configured on a per-directory basis, although the default with `-Duserelocatableinc` is that everything is relocated. The initial install is done to the original configured prefix.

**strlcat()** and **strlcpy()**
: The configuration process now detects whether **strlcat()** and **strlcpy()** are available. When they are not available, perl's own version is used (from Russ Allbery's public domain implementation). Various places in the perl interpreter now use them. (Steve Peters)

`d_pseudofork` and `d_printf_format_null`
: A new configuration variable, available as `$Config{d_pseudofork}` in the Config module, has been added, to distinguish real **fork()** support from fake pseudofork used on Windows platforms.

  A new configuration variable, `d_printf_format_null`, has been added, to see if printf-like formats are allowed to be NULL.

Configure help
: `Configure -h` has been extended with the most commonly used options.

### Compilation improvements

Parallel build
: Parallel makes should work properly now, although there may still be problems if `make test` is instructed to run in parallel.

Borland's compilers support
: Building with Borland's compilers on Win32 should work more smoothly. In particular Steve Hay has worked to side step many warnings emitted by their compilers and at least one C compiler internal error.

Static build on Windows
: Perl extensions on Windows now can be statically built into the Perl DLL.

  Also, it's now possible to build a `perl-static.exe` that doesn't depend on the Perl DLL on Win32. See the Win32 makefiles for details. (Vadim Konovalov)

ppport.h files
: All *ppport.h* files in the XS modules bundled with perl are now autogenerated at build time. (Marcus Holland-Moritz)

C++ compatibility
: Efforts have been made to make perl and the core XS modules compilable with various C++ compilers (although the situation is not perfect with some of the compilers on some of the platforms tested.)

Support for Microsoft 64-bit compiler
: Support for building perl with Microsoft's 64-bit compiler has been improved. (ActiveState)

Visual C++
: Perl can now be compiled with Microsoft Visual C++ 2005 (and 2008 Beta 2).

Win32 builds
: All win32 builds (MS-Win, WinCE) have been merged and cleaned up.

### Installation improvements

Module auxiliary files
: README files and changelogs for CPAN modules bundled with perl are no longer installed.

### New Or Improved Platforms

Perl has been reported to work on Symbian OS. See perlsymbian for more information.

Many improvements have been made towards making Perl work correctly on z/OS.

Perl has been reported to work on DragonFlyBSD and MidnightBSD.

Perl has also been reported to work on NexentaOS ( http://www.gnusolaris.org/ ).

The VMS port has been improved. See perlvms.

Support for Cray XT4 Catamount/Qk has been added. See *hints/catamount.sh* in the source code distribution for more information.

Vendor patches have been merged for RedHat and Gentoo.

**DynaLoader::dl_unload_file()** now works on Windows.

## Selected Bug Fixes

strictures in regexp-eval blocks
: `strict` wasn't in effect in regexp-eval blocks (`/(?{...})/`).

Calling **CORE::require()**
: **CORE::require()** and **CORE::do()** were always parsed as **require()** and **do()** when they were overridden. This is now fixed.

Subscripts of slices
: You can now use a non-arrowed form for chained subscripts after a list slice, like in:

```
({foo => "bar"})[0]{foo}
```

: This used to be a syntax error; a `->` was required.

`no warnings 'category'` works correctly with -w
: Previously when running with warnings enabled globally via `-w`, selective disabling of specific warning categories would actually turn off all warnings. This is now fixed; now `no warnings 'io';` will only turn off warnings in the `io` class. Previously it would erroneously turn off all warnings.

threads improvements
: Several memory leaks in ithreads were closed. Also, ithreads were made less memory-intensive.

: `threads` is now a dual-life module, also available on CPAN. It has been expanded in many ways. A **kill()** method is available for thread signalling. One can get thread status, or the list of running or joinable threads.

: A new `threads->exit()` method is used to exit from the application (this is the default for the main thread) or from the current thread only (this is the default for all other threads). On the other hand, the **exit()** built-in now always causes the whole application to terminate. (Jerry D. Hedden)

**chr()** and negative values
: **chr()** on a negative value now gives `\x{FFFD}`, the Unicode replacement character, unless when the `bytes` pragma is in effect, where the low eight bits of the value are used.

PERL5SHELL and tainting
: On Windows, the PERL5SHELL environment variable is now checked for taintedness. (Rafael Garcia-Suarez)

Using *FILE{IO}
: `stat()` and `-X` filetests now treat *FILE{IO} filehandles like *FILE filehandles. (Steve Peters)

Overloading and reblessing
: Overloading now works when references are reblessed into another class. Internally, this has been implemented by moving the flag for ''overloading'' from the reference to the referent, which logically is where it should always have been. (Nicholas Clark)

Overloading and UTF-8
: A few bugs related to UTF-8 handling with objects that have stringification overloaded have been fixed. (Nicholas Clark)

eval memory leaks fixed
: Traditionally, `eval 'syntax error'` has leaked badly. Many (but not all) of these leaks have now been eliminated or reduced. (Dave Mitchell)

Random device on Windows
: In previous versions, perl would read the file */dev/urandom* if it existed when seeding its random number generator. That file is unlikely to exist on Windows, and if it did would probably not contain appropriate data, so perl no longer tries to read it on Windows. (Alex Davies)

PERLIO_DEBUG
: The `PERLIO_DEBUG` environment variable no longer has any effect for setuid scripts and for scripts run with **-T**.

  Moreover, with a thread-enabled perl, using `PERLIO_DEBUG` could lead to an internal buffer overflow. This has been fixed.

PerlIO::scalar and read-only scalars
: PerlIO::scalar will now prevent writing to read-only scalars. Moreover, **seek()** is now supported with PerlIO::scalar-based filehandles, the underlying string being zero-filled as needed. (Rafael, Jarkko Hietaniemi)

**study()** and UTF-8
: **study()** never worked for UTF-8 strings, but could lead to false results. It's now a no-op on UTF-8 data. (Yves Orton)

Critical signals
: The signals SIGILL, SIGBUS and SIGSEGV are now always delivered in an "unsafe" manner (contrary to other signals, that are deferred until the perl interpreter reaches a reasonably stable state; see "Deferred Signals (Safe Signals)" in perlipc). (Rafael)

`@INC`-hook fix
: When a module or a file is loaded through an `@INC`-hook, and when this hook has set a filename entry in `%INC`, `__FILE__` is now set for this module accordingly to the contents of that `%INC` entry. (Rafael)

`-t` switch fix
: The `-w` and `-t` switches can now be used together without messing up which categories of warnings are activated. (Rafael)

Duping UTF-8 filehandles
: Duping a filehandle which has the `:utf8` PerlIO layer set will now properly carry that layer on the duped filehandle. (Rafael)

Localisation of hash elements
: Localizing a hash element whose key was given as a variable didn't work correctly if the variable was changed while the **local()** was in effect (as in `local $h{$x}; ++$x`). (Bo Lindbergh)

## New or Changed Diagnostics

Use of uninitialized value
: Perl will now try to tell you the name of the variable (if any) that was undefined.

Deprecated use of **my()** in false conditional
: A new deprecation warning, *Deprecated use of **my()** in false conditional*, has been added, to warn against the use of the dubious and deprecated construct

```
my $x if 0;
```

  See perldiag. Use `state` variables instead.

!=~ should be !~
: A new warning, `!=~ should be !~`, is emitted to prevent this misspelling of the non-matching operator.

Newline in left-justified string
: The warning *Newline in left-justified string* has been removed.

Too late for "-T" option
: The error *Too late for "-T" option* has been reformulated to be more descriptive.

"%s" variable `%s` masks earlier declaration
: This warning is now emitted in more consistent cases; in short, when one of the declarations involved is a `my` variable:

```
my $x;   my $x;     # warns
my $x;  our $x;     # warns
our $x;  my $x;     # warns
```

On the other hand, the following:

```
our $x; our $x;
```

now gives a `"our" variable %s redeclared` warning.

**readdir()/closedir()**/etc. attempted on invalid dirhandle
These new warnings are now emitted when a dirhandle is used but is either closed or not really a dirhandle.

Opening dirhandle/filehandle %s also as a file/directory
Two deprecation warnings have been added: (Rafael)

```
Opening dirhandle %s also as a file
Opening filehandle %s also as a directory
```

Use of −P is deprecated
Perl's command-line switch `-P` is now deprecated.

v−string in use/require is non-portable
Perl will warn you against potential backwards compatibility problems with the `use VERSION` syntax.

perl −V
`perl -V` has several improvements, making it more useable from shell scripts to get the value of configuration variables. See perlrun for details.

## Changed Internals

In general, the source code of perl has been refactored, tidied up, and optimized in many places. Also, memory management and allocation has been improved in several points.

When compiling the perl core with gcc, as many gcc warning flags are turned on as is possible on the platform. (This quest for cleanliness doesn't extend to XS code because we cannot guarantee the tidiness of code we didn't write.) Similar strictness flags have been added or tightened for various other C compilers.

### Reordering of SVt_* constants

The relative ordering of constants that define the various types of SV have changed; in particular, `SVt_PVGV` has been moved before `SVt_PVLV`, `SVt_PVAV`, `SVt_PVHV` and `SVt_PVCV`. This is unlikely to make any difference unless you have code that explicitly makes assumptions about that ordering. (The inheritance hierarchy of `B::*` objects has been changed to reflect this.)

### Elimination of SVt_PVBM

Related to this, the internal type `SVt_PVBM` has been removed. This dedicated type of SV was used by the `index` operator and parts of the regexp engine to facilitate fast Boyer-Moore matches. Its use internally has been replaced by SVs of type `SVt_PVGV`.

### New type SVt_BIND

A new type `SVt_BIND` has been added, in readiness for the project to implement Perl 6 on 5. There deliberately is no implementation yet, and they cannot yet be created or destroyed.

### Removal of CPP symbols

The C preprocessor symbols `PERL_PM_APIVERSION` and `PERL_XS_APIVERSION`, which were supposed to give the version number of the oldest perl binary-compatible (resp. source-compatible) with the present one, were not used, and sometimes had misleading values. They have been removed.

### Less space is used by ops

The `BASEOP` structure now uses less space. The `op_seq` field has been removed and replaced by a single bit bit-field `op_opt`. `op_type` is now 9 bits long. (Consequently, the `B::OP` class doesn't provide an `seq` method anymore.)

### New parser

perl's parser is now generated by bison (it used to be generated by byacc.) As a result, it seems to be a bit more robust.

Also, Dave Mitchell improved the lexer debugging output under `-DT`.

### Use of `const`

Andy Lester supplied many improvements to determine which function parameters and local variables could actually be declared `const` to the C compiler. Steve Peters provided new `*_set` macros and reworked the core to use these rather than assigning to macros in LVALUE context.

### Mathoms

A new file, *mathoms.c*, has been added. It contains functions that are no longer used in the perl core, but that remain available for binary or source compatibility reasons. However, those functions will not be compiled in if you add `-DNO_MATHOMS` in the compiler flags.

### `AvFLAGS` has been removed

The `AvFLAGS` macro has been removed.

### `av_*` changes

The `av_*()` functions, used to manipulate arrays, no longer accept null `AV*` parameters.

### $^H and %^H

The implementation of the special variables $^H and %^H has changed, to allow implementing lexical pragmas in pure Perl.

### B:: modules inheritance changed

The inheritance hierarchy of `B::` modules has changed; `B::NV` now inherits from `B::SV` (it used to inherit from `B::IV`).

### Anonymous hash and array constructors

The anonymous hash and array constructors now take 1 op in the optree instead of 3, now that pp_anonhash and pp_anonlist return a reference to a hash/array when the op is flagged with OPf_SPECIAL. (Nicholas Clark)

## Known Problems

There's still a remaining problem in the implementation of the lexical `$_`: it doesn't work inside `/(?{...})/` blocks. (See the TODO test in *t/op/mydef.t*.)

Stacked filetest operators won't work when the `filetest` pragma is in effect, because they rely on the **stat()** buffer _ being populated, and filetest bypasses **stat()**.

### UTF-8 problems

The handling of Unicode still is unclean in several places, where it's dependent on whether a string is internally flagged as UTF-8. This will be made more consistent in perl 5.12, but that won't be possible without a certain amount of backwards incompatibility.

## Platform Specific Problems

When compiled with g++ and thread support on Linux, it's reported that the `$!` stops working correctly. This is related to the fact that the glibc provides two **strerror_r**(3) implementation, and perl selects the wrong one.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/rt3/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

## SEE ALSO

The *Changes* file and the perl590delta to perl595delta man pages for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5101delta − what is new for perl v5.10.1

## DESCRIPTION

This document describes differences between the 5.10.0 release and the 5.10.1 release.

If you are upgrading from an earlier release such as 5.8.8, first read the perl5100delta, which describes differences between 5.8.8 and 5.10.0

## Incompatible Changes

### Switch statement changes

The handling of complex expressions by the `given`/`when` switch statement has been enhanced. There are two new cases where `when` now interprets its argument as a boolean, instead of an expression to be used in a smart match:

flip-flop operators

The `..` and `...` flip-flop operators are now evaluated in boolean context, following their usual semantics; see ‘‘Range Operators’’ in perlop.

Note that, as in perl 5.10.0, `when (1..10)` will not work to test whether a given value is an integer between 1 and 10; you should use `when ([1..10])` instead (note the array reference).

However, contrary to 5.10.0, evaluating the flip-flop operators in boolean context ensures it can now be useful in a `when()`, notably for implementing bistable conditions, like in:

```
when (/^=begin/ .. /^=end/) {
  # do something
}
```

defined-or operator

A compound expression involving the defined-or operator, as in `when (expr1 // expr2)`, will be treated as boolean if the first expression is boolean. (This just extends the existing rule that applies to the regular or operator, as in `when (expr1 || expr2)`.)

The next section details more changes brought to the semantics to the smart match operator, that naturally also modify the behaviour of the switch statements where smart matching is implicitly used.

### Smart match changes

*Changes to type-based dispatch*

The smart match operator `~~` is no longer commutative. The behaviour of a smart match now depends primarily on the type of its right hand argument. Moreover, its semantics have been adjusted for greater consistency or usefulness in several cases. While the general backwards compatibility is maintained, several changes must be noted:

- Code references with an empty prototype are no longer treated specially. They are passed an argument like the other code references (even if they choose to ignore it).
- `%hash ~~ sub {}` and `@array ~~ sub {}` now test that the subroutine returns a true value for each key of the hash (or element of the array), instead of passing the whole hash or array as a reference to the subroutine.
- Due to the commutativity breakage, code references are no longer treated specially when appearing on the left of the `~~` operator, but like any vulgar scalar.
- `undef ~~ %hash` is always false (since `undef` can’t be a key in a hash). No implicit conversion to `""` is done (as was the case in perl 5.10.0).
- `$scalar ~~ @array` now always distributes the smart match across the elements of the array. It’s true if one element in `@array` verifies `$scalar ~~ $element`. This is a generalization of the old behaviour that tested whether the array contained the scalar.

The full dispatch table for the smart match operator is given in ‘‘Smart matching in detail’’ in perlsyn.

*Smart match and overloading*

According to the rule of dispatch based on the rightmost argument type, when an object overloading `~~` appears on the right side of the operator, the overload routine will always be called (with a 3rd argument set to a true value, see overload.) However, when the object will appear on the left, the overload routine will be called only when the rightmost argument is a simple scalar. This way

distributivity of smart match across arrays is not broken, as well as the other behaviours with complex types (coderefs, hashes, regexes). Thus, writers of overloading routines for smart match mostly need to worry only with comparing against a scalar, and possibly with stringification overloading; the other common cases will be automatically handled consistently.

`~~` will now refuse to work on objects that do not overload it (in order to avoid relying on the object's underlying structure). (However, if the object overloads the stringification or the numification operators, and if overload fallback is active, it will be used instead, as usual.)

### Other incompatible changes

- The semantics of `use feature :5.10*` have changed slightly. See ''Modules and Pragmata'' for more information.
- It is now a run-time error to use the smart match operator `~~` with an object that has no overload defined for it. (This way `~~` will not break encapsulation by matching against the object's internal representation as a reference.)
- The version control system used for the development of the perl interpreter has been switched from Perforce to git. This is mainly an internal issue that only affects people actively working on the perl core; but it may have minor external visibility, for example in some of details of the output of `perl -V`. See perlrepository for more information.
- The internal structure of the `ext/` directory in the perl source has been reorganised. In general, a module `Foo::Bar` whose source was stored under *ext/Foo/Bar/* is now located under *ext/Foo-Bar/*. Also, some modules have been moved from *lib/* to *ext/*. This is purely a source tarball change, and should make no difference to the compilation or installation of perl, unless you have a very customised build process that explicitly relies on this structure, or which hard-codes the `nonxs_ext` *Configure* parameter. Specifically, this change does not by default alter the location of any files in the final installation.
- As part of the `Test::Harness` 2.x to 3.x upgrade, the experimental `Test::Harness::Straps` module has been removed. See ''Updated Modules'' for more details.
- As part of the `ExtUtils::MakeMaker` upgrade, the `ExtUtils::MakeMaker::bytes` and `ExtUtils::MakeMaker::vmsish` modules have been removed from this distribution.
- `Module::CoreList` no longer contains the `%:patchlevel` hash.
- This one is actually a change introduced in 5.10.0, but it was missed from that release's perldelta, so it is mentioned here instead.

  A bugfix related to the handling of the `/m` modifier and `qr` resulted in a change of behaviour between 5.8.x and 5.10.0:

```
# matches in 5.8.x, doesn't match in 5.10.0
$re = qr/^bar/; "foo\nbar" =~ /$re/m;
```

## Core Enhancements

### Unicode Character Database 5.1.0

The copy of the Unicode Character Database included in Perl 5.10.1 has been updated to 5.1.0 from 5.0.0. See <http://www.unicode.org/versions/Unicode5.1.0/#Notable_Changes> for the notable changes.

### A proper interface for pluggable Method Resolution Orders

As of Perl 5.10.1 there is a new interface for plugging and using method resolution orders other than the default (linear depth first search). The C3 method resolution order added in 5.10.0 has been re-implemented as a plugin, without changing its Perl-space interface. See perlmroapi for more information.

### The `overloading` pragma

This pragma allows you to lexically disable or enable overloading for some or all operations. (Yuval Kogman)

### Parallel tests

The core distribution can now run its regression tests in parallel on Unix-like platforms. Instead of running `make test`, set `TEST_JOBS` in your environment to the number of tests to run in parallel,

and run `make test_harness`. On a Bourne-like shell, this can be done as

```
TEST_JOBS=3 make test_harness  # Run 3 tests in parallel
```

An environment variable is used, rather than parallel make itself, because TAP::Harness needs to be able to schedule individual non-conflicting test scripts itself, and there is no standard interface to `make` utilities to interact with their job schedulers.

Note that currently some test scripts may fail when run in parallel (most notably `ext/IO/t/io_dir.t`). If necessary run just the failing scripts again sequentially and see if the failures go away.

### DTrace support

Some support for DTrace has been added. See ''DTrace support'' in *INSTALL*.

### Support for `configure_requires` in CPAN module metadata

Both `CPAN` and `CPANPLUS` now support the `configure_requires` keyword in the `META.yml` metadata file included in most recent CPAN distributions. This allows distribution authors to specify configuration prerequisites that must be installed before running *Makefile.PL* or *Build.PL*.

See the documentation for `ExtUtils::MakeMaker` or `Module::Build` for more on how to specify `configure_requires` when creating a distribution for CPAN.

## Modules and Pragmata

### New Modules and Pragmata

`autodie`

This is a new lexically-scoped alternative for the `Fatal` module. The bundled version is 2.06_01. Note that in this release, using a string eval when `autodie` is in effect can cause the autodie behaviour to leak into the surrounding scope. See ''BUGS'' in autodie for more details.

`Compress::Raw::Bzip2`

This has been added to the core (version 2.020).

`parent`

This pragma establishes an ISA relationship with base classes at compile time. It provides the key feature of `base` without the feature creep.

`Parse::CPAN::Meta`

This has been added to the core (version 1.39).

### Pragmata Changes

`attributes`

Upgraded from version 0.08 to 0.09.

`attrs`

Upgraded from version 1.02 to 1.03.

`base`

Upgraded from version 2.13 to 2.14. See parent for a replacement.

`bigint`

Upgraded from version 0.22 to 0.23.

`bignum`

Upgraded from version 0.22 to 0.23.

`bigrat`

Upgraded from version 0.22 to 0.23.

`charnames`

Upgraded from version 1.06 to 1.07.

The Unicode *NameAliases.txt* database file has been added. This has the effect of adding some extra `\N` character names that formerly wouldn't have been recognised; for example, `"\N{LATIN CAPITAL LETTER GHA}"`.

`constant`

Upgraded from version 1.13 to 1.17.

`feature`
: The meaning of the `:5.10` and `:5.10.X` feature bundles has changed slightly. The last component, if any (i.e. `X`) is simply ignored. This is predicated on the assumption that new features will not, in general, be added to maintenance releases. So `:5.10` and `:5.10.X` have identical effect. This is a change to the behaviour documented for 5.10.0.

`fields`
: Upgraded from version 2.13 to 2.14 (this was just a version bump; there were no functional changes).

`lib`
: Upgraded from version 0.5565 to 0.62.

`open`
: Upgraded from version 1.06 to 1.07.

`overload`
: Upgraded from version 1.06 to 1.07.

`overloading`
: See "The `overloading` pragma" above.

`version`
: Upgraded from version 0.74 to 0.77.

### Updated Modules

`Archive::Extract`
: Upgraded from version 0.24 to 0.34.

`Archive::Tar`
: Upgraded from version 1.38 to 1.52.

`Attribute::Handlers`
: Upgraded from version 0.79 to 0.85.

`AutoLoader`
: Upgraded from version 5.63 to 5.68.

`AutoSplit`
: Upgraded from version 1.05 to 1.06.

`B` Upgraded from version 1.17 to 1.22.

`B::Debug`
: Upgraded from version 1.05 to 1.11.

`B::Deparse`
: Upgraded from version 0.83 to 0.89.

`B::Lint`
: Upgraded from version 1.09 to 1.11.

`B::Xref`
: Upgraded from version 1.01 to 1.02.

`Benchmark`
: Upgraded from version 1.10 to 1.11.

`Carp`
: Upgraded from version 1.08 to 1.11.

`CGI`
: Upgraded from version 3.29 to 3.43. (also includes the ‘‘default_value for **popup_menu()**’’ fix from 3.45).

`Compress::Zlib`
: Upgraded from version 2.008 to 2.020.

`CPAN`
: Upgraded from version 1.9205 to 1.9402. `CPAN::FTP` has a local fix to stop it being too verbose on download failure.

`CPANPLUS`
Upgraded from version 0.84 to 0.88.

`CPANPLUS::Dist::Build`
Upgraded from version 0.06_02 to 0.36.

`Cwd`
Upgraded from version 3.25_01 to 3.30.

`Data::Dumper`
Upgraded from version 2.121_14 to 2.124.

`DB` Upgraded from version 1.01 to 1.02.

`DB_File`
Upgraded from version 1.816_1 to 1.820.

`Devel::PPPort`
Upgraded from version 3.13 to 3.19.

`Digest::MD5`
Upgraded from version 2.36_01 to 2.39.

`Digest::SHA`
Upgraded from version 5.45 to 5.47.

`DirHandle`
Upgraded from version 1.01 to 1.03.

`Dumpvalue`
Upgraded from version 1.12 to 1.13.

`DynaLoader`
Upgraded from version 1.08 to 1.10.

`Encode`
Upgraded from version 2.23 to 2.35.

`Errno`
Upgraded from version 1.10 to 1.11.

`Exporter`
Upgraded from version 5.62 to 5.63.

`ExtUtils::CBuilder`
Upgraded from version 0.21 to 0.2602.

`ExtUtils::Command`
Upgraded from version 1.13 to 1.16.

`ExtUtils::Constant`
Upgraded from 0.20 to 0.22. (Note that neither of these versions are available on CPAN.)

`ExtUtils::Embed`
Upgraded from version 1.27 to 1.28.

`ExtUtils::Install`
Upgraded from version 1.44 to 1.54.

`ExtUtils::MakeMaker`
Upgraded from version 6.42 to 6.55_02.

Note that `ExtUtils::MakeMaker::bytes` and `ExtUtils::MakeMaker::vmsish` have been removed from this distribution.

`ExtUtils::Manifest`
Upgraded from version 1.51_01 to 1.56.

`ExtUtils::ParseXS`
Upgraded from version 2.18_02 to 2.2002.

`Fatal`
Upgraded from version 1.05 to 2.06_01. See also the new pragma `autodie`.

`File::Basename`
Upgraded from version 2.76 to 2.77.

`File::Compare`
Upgraded from version 1.1005 to 1.1006.

`File::Copy`
Upgraded from version 2.11 to 2.14.

`File::Fetch`
Upgraded from version 0.14 to 0.20.

`File::Find`
Upgraded from version 1.12 to 1.14.

`File::Path`
Upgraded from version 2.04 to 2.07_03.

`File::Spec`
Upgraded from version 3.2501 to 3.30.

`File::stat`
Upgraded from version 1.00 to 1.01.

`File::Temp`
Upgraded from version 0.18 to 0.22.

`FileCache`
Upgraded from version 1.07 to 1.08.

`FileHandle`
Upgraded from version 2.01 to 2.02.

`Filter::Simple`
Upgraded from version 0.82 to 0.84.

`Filter::Util::Call`
Upgraded from version 1.07 to 1.08.

`FindBin`
Upgraded from version 1.49 to 1.50.

`GDBM_File`
Upgraded from version 1.08 to 1.09.

`Getopt::Long`
Upgraded from version 2.37 to 2.38.

`Hash::Util::FieldHash`
Upgraded from version 1.03 to 1.04. This fixes a memory leak.

`I18N::Collate`
Upgraded from version 1.00 to 1.01.

`IO` Upgraded from version 1.23_01 to 1.25.

This makes non-blocking mode work on Windows in `IO::Socket::INET` [CPAN #43573].

`IO::Compress::*`
Upgraded from version 2.008 to 2.020.

`IO::Dir`
Upgraded from version 1.06 to 1.07.

`IO::Handle`
Upgraded from version 1.27 to 1.28.

`IO::Socket`
Upgraded from version 1.30_01 to 1.31.

`IO::Zlib`
Upgraded from version 1.07 to 1.09.

`IPC::Cmd`
Upgraded from version 0.40_1 to 0.46.

`IPC::Open3`
Upgraded from version 1.02 to 1.04.

`IPC::SysV`
Upgraded from version 1.05 to 2.01.

`lib`
Upgraded from version 0.5565 to 0.62.

`List::Util`
Upgraded from version 1.19 to 1.21.

`Locale::MakeText`
Upgraded from version 1.12 to 1.13.

`Log::Message`
Upgraded from version 0.01 to 0.02.

`Math::BigFloat`
Upgraded from version 1.59 to 1.60.

`Math::BigInt`
Upgraded from version 1.88 to 1.89.

`Math::BigInt::FastCalc`
Upgraded from version 0.16 to 0.19.

`Math::BigRat`
Upgraded from version 0.21 to 0.22.

`Math::Complex`
Upgraded from version 1.37 to 1.56.

`Math::Trig`
Upgraded from version 1.04 to 1.20.

`Memoize`
Upgraded from version 1.01_02 to 1.01_03 (just a minor documentation change).

`Module::Build`
Upgraded from version 0.2808_01 to 0.34_02.

`Module::CoreList`
Upgraded from version 2.13 to 2.18. This release no longer contains the `%Module::CoreList::patchlevel` hash.

`Module::Load`
Upgraded from version 0.12 to 0.16.

`Module::Load::Conditional`
Upgraded from version 0.22 to 0.30.

`Module::Loaded`
Upgraded from version 0.01 to 0.02.

`Module::Pluggable`
Upgraded from version 3.6 to 3.9.

`NDBM_File`
Upgraded from version 1.07 to 1.08.

`Net::Ping`
Upgraded from version 2.33 to 2.36.

`NEXT`
Upgraded from version 0.60_01 to 0.64.

`Object::Accessor`
Upgraded from version 0.32 to 0.34.

`OS2::REXX`
Upgraded from version 1.03 to 1.04.

`Package::Constants`
Upgraded from version 0.01 to 0.02.

`PerlIO`
Upgraded from version 1.04 to 1.06.

`PerlIO::via`
Upgraded from version 0.04 to 0.07.

`Pod::Man`
Upgraded from version 2.16 to 2.22.

`Pod::Parser`
Upgraded from version 1.35 to 1.37.

`Pod::Simple`
Upgraded from version 3.05 to 3.07.

`Pod::Text`
Upgraded from version 3.08 to 3.13.

`POSIX`
Upgraded from version 1.13 to 1.17.

`Safe`
Upgraded from 2.12 to 2.18.

`Scalar::Util`
Upgraded from version 1.19 to 1.21.

`SelectSaver`
Upgraded from 1.01 to 1.02.

`SelfLoader`
Upgraded from 1.11 to 1.17.

`Socket`
Upgraded from 1.80 to 1.82.

`Storable`
Upgraded from 2.18 to 2.20.

`Switch`
Upgraded from version 2.13 to 2.14. Please see ‘‘Deprecations’’.

`Symbol`
Upgraded from version 1.06 to 1.07.

`Sys::Syslog`
Upgraded from version 0.22 to 0.27.

`Term::ANSIColor`
Upgraded from version 1.12 to 2.00.

`Term::ReadLine`
Upgraded from version 1.03 to 1.04.

`Term::UI`
Upgraded from version 0.18 to 0.20.

`Test::Harness`
Upgraded from version 2.64 to 3.17.

Note that one side-effect of the 2.x to 3.x upgrade is that the experimental

`Test::Harness::Straps` module (and its supporting `Assert`, `Iterator`, `Point` and `Results` modules) have been removed. If you still need this, then they are available in the (unmaintained) `Test-Harness-Straps` distribution on CPAN.

`Test::Simple`
Upgraded from version 0.72 to 0.92.

`Text::ParseWords`
Upgraded from version 3.26 to 3.27.

`Text::Tabs`
Upgraded from version 2007.1117 to 2009.0305.

`Text::Wrap`
Upgraded from version 2006.1117 to 2009.0305.

`Thread::Queue`
Upgraded from version 2.00 to 2.11.

`Thread::Semaphore`
Upgraded from version 2.01 to 2.09.

`threads`
Upgraded from version 1.67 to 1.72.

`threads::shared`
Upgraded from version 1.14 to 1.29.

`Tie::RefHash`
Upgraded from version 1.37 to 1.38.

`Tie::StdHandle`
This has documentation changes, and has been assigned a version for the first time: version 4.2.

`Time::HiRes`
Upgraded from version 1.9711 to 1.9719.

`Time::Local`
Upgraded from version 1.18 to 1.1901.

`Time::Piece`
Upgraded from version 1.12 to 1.15.

`Unicode::Normalize`
Upgraded from version 1.02 to 1.03.

`Unicode::UCD`
Upgraded from version 0.25 to 0.27.

`charinfo()` now works on Unified CJK code points added to later versions of Unicode.

`casefold()` has new fields returned to provide both a simpler interface and previously missing information. The old fields are retained for backwards compatibility. Information about Turkic-specific code points is now returned.

The documentation has been corrected and expanded.

`UNIVERSAL`
Upgraded from version 1.04 to 1.05.

`Win32`
Upgraded from version 0.34 to 0.39.

`Win32API::File`
Upgraded from version 0.1001_01 to 0.1101.

`XSLoader`
Upgraded from version 0.08 to 0.10.

## Utility Changes

*h2ph*
: Now looks in `include-fixed` too, which is a recent addition to gcc's search path.

*h2xs*
: No longer incorrectly treats enum values like macros (Daniel Burr).

  Now handles C++ style constants (`//`) properly in enums. (A patch from Rainer Weikusat was used; Daniel Burr also proposed a similar fix).

*perl5db.pl*
: `LVALUE` subroutines now work under the debugger.

  The debugger now correctly handles proxy constant subroutines, and subroutine stubs.

*perlthanks*
: Perl 5.10.1 adds a new utility *perlthanks*, which is a variant of *perlbug*, but for sending non-bug-reports to the authors and maintainers of Perl. Getting nothing but bug reports can become a bit demoralising: we'll see if this changes things.

## New Documentation

perlhaiku
: This contains instructions on how to build perl for the Haiku platform.

perlmroapi
: This describes the new interface for pluggable Method Resolution Orders.

perlperf
: This document, by Richard Foley, provides an introduction to the use of performance and optimization techniques which can be used with particular reference to perl programs.

perlrepository
: This describes how to access the perl source using the *git* version control system.

perlthanks
: This describes the new *perlthanks* utility.

## Changes to Existing Documentation

The various large `Changes*` files (which listed every change made to perl over the last 18 years) have been removed, and replaced by a small file, also called `Changes`, which just explains how that same information may be extracted from the git version control system.

The file *Porting/patching.pod* has been deleted, as it mainly described interacting with the old Perforce-based repository, which is now obsolete. Information still relevant has been moved to perlrepository.

perlapi, perlintern, perlmodlib and perltoc are now all generated at build time, rather than being shipped as part of the release.

## Performance Enhancements

- A new internal cache means that `isa()` will often be faster.
- Under `use locale`, the locale-relevant information is now cached on read-only values, such as the list returned by `keys %hash`. This makes operations such as `sort keys %hash` in the scope of `use locale` much faster.
- Empty `DESTROY` methods are no longer called.

## Installation and Configuration Improvements

### *ext/* reorganisation

The layout of directories in *ext* has been revised. Specifically, all extensions are now flat, and at the top level, with `/` in pathnames replaced by `-`, so that *ext/Data/Dumper/* is now *ext/Data-Dumper/*, etc. The names of the extensions as specified to *Configure*, and as reported by `%Config::Config` under the keys `dynamic_ext`, `known_extensions`, `nonxs_ext` and `static_ext` have not changed, and still use `/`. Hence this change will not have any affect once perl is installed. However, `Attribute::Handlers`, `Safe` and `mro` have now become extensions in their own right, so if you run *Configure* with options to specify an exact list of extensions to build, you will need to change it to account for this.

For 5.10.2, it is planned that many dual-life modules will have been moved from *lib* to *ext*; again this will have no effect on an installed perl, but will matter if you invoke *Configure* with a pre-canned list of

extensions to build.

### Configuration improvements

If `vendorlib` and `vendorarch` are the same, then they are only added to `@INC` once.

`$Config{usedevel}` and the C−level `PERL_USE_DEVEL` are now defined if perl is built with `−Dusedevel`.

*Configure* will enable use of `−fstack−protector`, to provide protection against stack-smashing attacks, if the compiler supports it.

*Configure* will now determine the correct prototypes for re-entrant functions, and for `gconvert`, if you are using a C++ compiler rather than a C compiler.

On Unix, if you build from a tree containing a git repository, the configuration process will note the commit hash you have checked out, for display in the output of `perl −v` and `perl −V`. Unpushed local commits are automatically added to the list of local patches displayed by `perl −V`.

### Compilation improvements

As part of the flattening of *ext*, all extensions on all platforms are built by *make_ext.pl*. This replaces the Unix-specific *ext/util/make_ext*, VMS-specific *make_ext.com* and Win32−specific *win32/buildext.pl*.

### Platform Specific Changes

AIX

Removed *libbsd* for AIX 5L and 6.1. Only **flock()** was used from *libbsd*.

Removed *libgdbm* for AIX 5L and 6.1. The *libgdbm* is delivered as an optional package with the AIX Toolbox. Unfortunately the 64 bit version is broken.

Hints changes mean that AIX 4.2 should work again.

Cygwin

On Cygwin we now strip the last number from the DLL. This has been the behaviour in the cygwin.com build for years. The hints files have been updated.

FreeBSD

The hints files now identify the correct threading libraries on FreeBSD 7 and later.

Irix We now work around a bizarre preprocessor bug in the Irix 6.5 compiler: `cc −E −` unfortunately goes into K&R mode, but `cc −E file.c` doesn't.

Haiku

Patches from the Haiku maintainers have been merged in. Perl should now build on Haiku.

MirOS BSD

Perl should now build on MirOS BSD.

NetBSD

Hints now supports versions 5.*.

Stratus VOS

Various changes from Stratus have been merged in.

Symbian

There is now support for Symbian S60 3.2 SDK and S60 5.0 SDK.

Win32

Improved message window handling means that `alarm` and `kill` messages will no longer be dropped under race conditions.

VMS

Reads from the in-memory temporary files of `PerlIO::scalar` used to fail if `$/` was set to a numeric reference (to indicate record-style reads). This is now fixed.

VMS now supports `getgrgid`.

Many improvements and cleanups have been made to the VMS file name handling and conversion code.

Enabling the `PERL_VMS_POSIX_EXIT` logical name now encodes a POSIX exit status in a VMS

condition value for better interaction with GNV's bash shell and other utilities that depend on POSIX exit values. See "$?" in perlvms for details.

## Selected Bug Fixes

- 5.10.0 inadvertently disabled an optimisation, which caused a measurable performance drop in list assignment, such as is often used to assign function parameters from `@_`. The optimisation has been re-instated, and the performance regression fixed.
- Fixed memory leak on `while (1) { map 1, 1 }` [RT #53038].
- Some potential coredumps in PerlIO fixed [RT #57322,54828].
- The debugger now works with lvalue subroutines.
- The debugger's `m` command was broken on modules that defined constants [RT #61222].
- `crypt()` and string complement could return tainted values for untainted arguments [RT #59998].
- The `-i.suffix` command-line switch now recreates the file using restricted permissions, before changing its mode to match the original file. This eliminates a potential race condition [RT #60904].
- On some Unix systems, the value in `$?` would not have the top bit set (`$? & 128`) even if the child core dumped.
- Under some circumstances, $^R could incorrectly become undefined [RT #57042].
- (XS) In various hash functions, passing a pre-computed hash to when the key is UTF-8 might result in an incorrect lookup.
- (XS) Including *XSUB.h* before *perl.h* gave a compile-time error [RT #57176].
- `$object->isa('Foo')` would report false if the package `Foo` didn't exist, even if the object's `@ISA` contained `Foo`.
- Various bugs in the new-to 5.10.0 mro code, triggered by manipulating `@ISA`, have been found and fixed.
- Bitwise operations on references could crash the interpreter, e.g. `$x=\$y; $x |= "foo"` [RT #54956].
- Patterns including alternation might be sensitive to the internal UTF-8 representation, e.g.

```
my $byte = chr(192);
my $utf8 = chr(192); utf8::upgrade($utf8);
$utf8 =~ /$byte|X}/i;       # failed in 5.10.0
```

- Within UTF8-encoded Perl source files (i.e. where `use utf8` is in effect), double-quoted literal strings could be corrupted where a `\xNN`, `\0NNN` or `\N{}` is followed by a literal character with ordinal value greater than 255 [RT #59908].
- `B::Deparse` failed to correctly deparse various constructs: `readpipe STRING` [RT #62428], `CORE::require(STRING)` [RT #62488], `sub foo(_)` [RT #62484].
- Using `setpgrp()` with no arguments could corrupt the perl stack.
- The block form of `eval` is now specifically trappable by `Safe` and `ops`. Previously it was erroneously treated like string `eval`.
- In 5.10.0, the two characters `[~` were sometimes parsed as the smart match operator (`~~`) [RT #63854].
- In 5.10.0, the `*` quantifier in patterns was sometimes treated as `{0,32767}` [RT #60034, #60464]. For example, this match would fail:

```
("ab" x 32768) =~ /^(ab)*$/
```

- `shmget` was limited to a 32 bit segment size on a 64 bit OS [RT #63924].
- Using `next` or `last` to exit a `given` block no longer produces a spurious warning like the following:

```
Exiting given via last at foo.pl line 123
```

- On Windows, `'.\foo'` and `'..\foo'` were treated differently than `'./foo'` and `'../foo'` by `do` and `require` [RT #63492].
- Assigning a format to a glob could corrupt the format; e.g.:

```
*bar=*foo{FORMAT}; # foo format now bad
```

- Attempting to coerce a typeglob to a string or number could cause an assertion failure. The correct error message is now generated, `Can't coerce GLOB to` *`$type`*.
- Under `use filetest 'access'`, `-x` was using the wrong access mode. This has been fixed [RT #49003].
- `length` on a tied scalar that returned a Unicode value would not be correct the first time. This has been fixed.
- Using an array `tie` inside in array `tie` could SEGV. This has been fixed. [RT #51636]
- A race condition inside `PerlIOStdio_close()` has been identified and fixed. This used to cause various threading issues, including SEGVs.
- In `unpack`, the use of `()` groups in scalar context was internally placing a list on the interpreter's stack, which manifested in various ways, including SEGVs. This is now fixed [RT #50256].
- Magic was called twice in `substr`, `\&$x`, `tie $x, $m` and `chop`. These have all been fixed.
- A 5.10.0 optimisation to clear the temporary stack within the implicit loop of `s///ge` has been reverted, as it turned out to be the cause of obscure bugs in seemingly unrelated parts of the interpreter [commit ef0d4e17921ee3de].
- The line numbers for warnings inside `elsif` are now correct.
- The `..` operator now works correctly with ranges whose ends are at or close to the values of the smallest and largest integers.
- `binmode STDIN, ':raw'` could lead to segmentation faults on some platforms. This has been fixed [RT #54828].
- An off-by-one error meant that `index $str, ...` was effectively being executed as `index "$str\0", ...`. This has been fixed [RT #53746].
- Various leaks associated with named captures in regexes have been fixed [RT #57024].
- A weak reference to a hash would leak. This was affecting `DBI` [RT #56908].
- Using (?|) in a regex could cause a segfault [RT #59734].
- Use of a UTF-8 `tr//` within a closure could cause a segfault [RT #61520].
- Calling `sv_chop()` or otherwise upgrading an SV could result in an unaligned 64-bit access on the SPARC architecture [RT #60574].
- In the 5.10.0 release, `inc_version_list` would incorrectly list `5.10.*` after `5.8.*`; this affected the `@INC` search order [RT #67628].
- In 5.10.0, `pack "a*", $tainted_value` returned a non-tainted value [RT #52552].
- In 5.10.0, `printf` and `sprintf` could produce the fatal error `panic: utf8_mg_pos_cache_update` when printing UTF-8 strings [RT #62666].
- In the 5.10.0 release, a dynamically created `AUTOLOAD` method might be missed (method cache issue) [RT #60220,60232].
- In the 5.10.0 release, a combination of `use feature` and `//ee` could cause a memory leak [RT #63110].
- `-C` on the shebang (`#!`) line is once more permitted if it is also specified on the command line. `-C` on the shebang line used to be a silent no-op *if* it was not also on the command line, so perl 5.10.0 disallowed it, which broke some scripts. Now perl checks whether it is also on the command line and only dies if it is not [RT #67880].

- In 5.10.0, certain types of re-entrant regular expression could crash, or cause the following assertion failure [RT #60508]:

```
Assertion rx->sublen >= (s - rx->subbeg) + i failed
```

## New or Changed Diagnostics

`panic: sv_chop %s`
: This new fatal error occurs when the C routine `Perl_sv_chop()` was passed a position that is not within the scalar's string buffer. This could be caused by buggy XS code, and at this point recovery is not possible.

`Can't locate package %s for the parents of %s`
: This warning has been removed. In general, it only got produced in conjunction with other warnings, and removing it allowed an ISA lookup optimisation to be added.

`v-string in use/require is non-portable`
: This warning has been removed.

`Deep recursion on subroutine ''%s''`
: It is now possible to change the depth threshold for this warning from the default of 100, by recompiling the *perl* binary, setting the C pre-processor macro `PERL_SUB_DEPTH_WARN` to the desired value.

## Changed Internals

- The J.R.R. Tolkien quotes at the head of C source file have been checked and proper citations added, thanks to a patch from Tom Christiansen.
- `vcroak()` now accepts a null first argument. In addition, a full audit was made of the ''not NULL'' compiler annotations, and those for several other internal functions were corrected.
- New macros `dSAVEDERRNO`, `dSAVE_ERRNO`, `SAVE_ERRNO`, `RESTORE_ERRNO` have been added to formalise the temporary saving of the `errno` variable.
- The function `Perl_sv_insert_flags` has been added to augment `Perl_sv_insert`.
- The function `Perl_newSV_type(type)` has been added, equivalent to `Perl_newSV()` followed by `Perl_sv_upgrade(type)`.
- The function `Perl_newSVpvn_flags()` has been added, equivalent to `Perl_newSVpvn()` and then performing the action relevant to the flag.

  Two flag bits are currently supported.

  `SVf_UTF8`
  : This will call `SvUTF8_on()` for you. (Note that this does not convert an sequence of ISO 8859-1 characters to UTF-8). A wrapper, `newSVpvn_utf8()` is available for this.

  `SVs_TEMP`
  : Call `sv_2mortal()` on the new SV.

  There is also a wrapper that takes constant strings, `newSVpvs_flags()`.
- The function `Perl_croak_xs_usage` has been added as a wrapper to `Perl_croak`.
- The functions `PerlIO_find_layer` and `PerlIO_list_alloc` are now exported.
- `PL_na` has been exterminated from the core code, replaced by local STRLEN temporaries, or `*_nolen()` calls. Either approach is faster than `PL_na`, which is a pointer deference into the interpreter structure under ithreads, and a global variable otherwise.
- `Perl_mg_free()` used to leave freed memory accessible via **SvMAGIC()** on the scalar. It now updates the linked list to remove each piece of magic as it is freed.
- Under ithreads, the regex in `PL_reg_curpm` is now reference counted. This eliminates a lot of hackish workarounds to cope with it not being reference counted.
- `Perl_mg_magical()` would sometimes incorrectly turn on `SvRMAGICAL()`. This has been fixed.
- The *public* IV and NV flags are now not set if the string value has trailing ''garbage''. This behaviour is consistent with not setting the public IV or NV flags if the value is out of range for the type.

- SV allocation tracing has been added to the diagnostics enabled by `-Dm`. The tracing can alternatively output via the `PERL_MEM_LOG` mechanism, if that was enabled when the *perl* binary was compiled.
- Uses of `Nullav`, `Nullcv`, `Nullhv`, `Nullop`, `Nullsv` etc have been replaced by `NULL` in the core code, and non-dual-life modules, as `NULL` is clearer to those unfamiliar with the core code.
- A macro `MUTABLE_PTR(p)` has been added, which on (non-pedantic) gcc will not cast away `const`, returning a `void *`. Macros `MUTABLE_SV(av)`, `MUTABLE_SV(cv)` etc build on this, casting to `AV *` etc without casting away `const`. This allows proper compile-time auditing of `const` correctness in the core, and helped picked up some errors (now fixed).
- Macros `mPUSHs()` and `mXPUSHs()` have been added, for pushing SVs on the stack and mortalizing them.
- Use of the private structure `mro_meta` has changed slightly. Nothing outside the core should be accessing this directly anyway.
- A new tool, `Porting/expand-macro.pl` has been added, that allows you to view how a C preprocessor macro would be expanded when compiled. This is handy when trying to decode the macro hell that is the perl guts.

## New Tests

Many modules updated from CPAN incorporate new tests.

Several tests that have the potential to hang forever if they fail now incorporate a ‘‘watchdog’’ functionality that will kill them after a timeout, which helps ensure that `make test` and `make test_harness` run to completion automatically. (Jerry Hedden).

Some core-specific tests have been added:

t/comp/retainedlines.t
: Check that the debugger can retain source lines from `eval`.

t/io/perlio_fail.t
: Check that bad layers fail.

t/io/perlio_leaks.t
: Check that PerlIO layers are not leaking.

t/io/perlio_open.t
: Check that certain special forms of open work.

t/io/perlio.t
: General PerlIO tests.

t/io/pvbm.t
: Check that there is no unexpected interaction between the internal types `PVBM` and `PVGV`.

t/mro/package_aliases.t
: Check that mro works properly in the presence of aliased packages.

t/op/dbm.t
: Tests for `dbmopen` and `dbmclose`.

t/op/index_thr.t
: Tests for the interaction of `index` and threads.

t/op/pat_thr.t
: Tests for the interaction of esoteric patterns and threads.

t/op/qr_gc.t
: Test that `qr` doesn’t leak.

t/op/reg_email_thr.t
: Tests for the interaction of regex recursion and threads.

t/op/regexp_qr_embed_thr.t
: Tests for the interaction of patterns with embedded `qr//` and threads.

t/op/regexp_unicode_prop.t
: Tests for Unicode properties in regular expressions.

t/op/regexp_unicode_prop_thr.t
: Tests for the interaction of Unicode properties and threads.

t/op/reg_nc_tie.t
: Test the tied methods of `Tie::Hash::NamedCapture`.

t/op/reg_posixcc.t
: Check that POSIX character classes behave consistently.

t/op/re.t
: Check that exportable `re` functions in *universal.c* work.

t/op/setpgrpstack.t
: Check that `setpgrp` works.

t/op/substr_thr.t
: Tests for the interaction of `substr` and threads.

t/op/upgrade.t
: Check that upgrading and assigning scalars works.

t/uni/lex_utf8.t
: Check that Unicode in the lexer works.

t/uni/tie.t
: Check that Unicode and `tie` work.

## Known Problems

This is a list of some significant unfixed bugs, which are regressions from either 5.10.0 or 5.8.x.

- `List::Util::first` misbehaves in the presence of a lexical `$_` (typically introduced by `my $_` or implicitly by `given`). The variable which gets set for each iteration is the package variable `$_`, not the lexical `$_` [RT #67694].

  A similar issue may occur in other modules that provide functions which take a block as their first argument, like

  ```
  foo { ... $_ ...} list
  ```

- The `charnames` pragma may generate a run-time error when a regex is interpolated [RT #56444]:

  ```
  use charnames ':full';
  my $r1 = qr/\N{THAI CHARACTER SARA I}/;
  "foo" =~ $r1;    # okay
  "foo" =~ /$r1+/; # runtime error
  ```

  A workaround is to generate the character outside of the regex:

  ```
  my $a = "\N{THAI CHARACTER SARA I}";
  my $r1 = qr/$a/;
  ```

- Some regexes may run much more slowly when run in a child thread compared with the thread the pattern was compiled into [RT #55600].

## Deprecations

The following items are now deprecated.

- `Switch` is buggy and should be avoided. From perl 5.11.0 onwards, it is intended that any use of the core version of this module will emit a warning, and that the module will eventually be removed from the core (probably in perl 5.14.0). See ‘‘Switch statements’’ in perlsyn for its replacement.
- `suidperl` will be removed in 5.12.0. This provides a mechanism to emulate setuid permission bits on systems that don’t support it properly.

## Acknowledgements

Some of the work in this release was funded by a TPF grant.

Nicholas Clark officially retired from maintenance pumpking duty at the end of 2008; however in reality he has put much effort in since then to help get 5.10.1 into a fit state to be released, including writing a considerable chunk of this perldelta.

Steffen Mueller and David Golden in particular helped getting CPAN modules polished and synchronised with their in-core equivalents.

Craig Berry was tireless in getting maint to run under VMS, no matter how many times we broke it for him.

The other core committers contributed most of the changes, and applied most of the patches sent in by the hundreds of contributors listed in *AUTHORS*.

(Sorry to all the people I haven't mentioned by name).

Finally, thanks to Larry Wall, without whom none of this would be necessary.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5120delta − what is new for perl v5.12.0

## DESCRIPTION

This document describes differences between the 5.10.0 release and the 5.12.0 release.

Many of the bug fixes in 5.12.0 are already included in the 5.10.1 maintenance release.

You can see the list of those changes in the 5.10.1 release notes (perl5101delta).

## Core Enhancements

### New `package NAME VERSION` syntax

This new syntax allows a module author to set the `$VERSION` of a namespace when the namespace is declared with 'package'. It eliminates the need for `our $VERSION = ...` and similar constructs. E.g.

```
package Foo::Bar 1.23;
# $Foo::Bar::VERSION == 1.23
```

There are several advantages to this:

- `$VERSION` is parsed in exactly the same way as `use NAME VERSION`
- `$VERSION` is set at compile time
- `$VERSION` is a version object that provides proper overloading of comparison operators so comparing `$VERSION` to decimal (1.23) or dotted-decimal (v1.2.3) version numbers works correctly.
- Eliminates `$VERSION = ...` and `eval $VERSION` clutter
- As it requires VERSION to be a numeric literal or v−string literal, it can be statically parsed by toolchain modules without `eval` the way MM−>parse_version does for `$VERSION = ...`

It does not break old code with only `package NAME`, but code that uses `package NAME VERSION` will need to be restricted to perl 5.12.0 or newer This is analogous to the change to `open` from two-args to three-args. Users requiring the latest Perl will benefit, and perhaps after several years, it will become a standard practice.

However, `package NAME VERSION` requires a new, 'strict' version number format. See ''Version number formats'' for details.

### The `...` operator

A new operator, `...`, nicknamed the Yada Yada operator, has been added. It is intended to mark placeholder code that is not yet implemented. See ''Yada Yada Operator'' in perlop.

### Implicit strictures

Using the `use VERSION` syntax with a version number greater or equal to 5.11.0 will lexically enable strictures just like `use strict` would do (in addition to enabling features.) The following:

```
use 5.12.0;
```

means:

```
use strict;
use feature ':5.12';
```

### Unicode improvements

Perl 5.12 comes with Unicode 5.2, the latest version available to us at the time of release. This version of Unicode was released in October 2009. See <http://www.unicode.org/versions/Unicode5.2.0> for further details about what's changed in this version of the standard. See perlunicode for instructions on installing and using other versions of Unicode.

Additionally, Perl's developers have significantly improved Perl's Unicode implementation. For full details, see ''Unicode overhaul'' below.

### Y2038 compliance

Perl's core time-related functions are now Y2038 compliant. (It may not mean much to you, but your kids will love it!)

### qr overloading

It is now possible to overload the `qr//` operator, that is, conversion to regexp, like it was already possible to overload conversion to boolean, string or number of objects. It is invoked when an object appears on the right hand side of the `=~` operator or when it is interpolated into a regexp. See overload.

### Pluggable keywords

Extension modules can now cleanly hook into the Perl parser to define new kinds of keyword-headed expression and compound statement. The syntax following the keyword is defined entirely by the extension. This allows a completely non-Perl sublanguage to be parsed inline, with the correct ops cleanly generated.

See ''PL_keyword_plugin'' in perlapi for the mechanism. The Perl core source distribution also includes a new module XS::APItest::KeywordRPN, which implements reverse Polish notation arithmetic via pluggable keywords. This module is mainly used for test purposes, and is not normally installed, but also serves as an example of how to use the new mechanism.

Perl's developers consider this feature to be experimental. We may remove it or change it in a backwards-incompatible way in Perl 5.14.

### APIs for more internals

The lowest layers of the lexer and parts of the pad system now have C APIs available to XS extensions. These are necessary to support proper use of pluggable keywords, but have other uses too. The new APIs are experimental, and only cover a small proportion of what would be necessary to take full advantage of the core's facilities in these areas. It is intended that the Perl 5.13 development cycle will see the addition of a full range of clean, supported interfaces.

Perl's developers consider this feature to be experimental. We may remove it or change it in a backwards-incompatible way in Perl 5.14.

### Overridable function lookup

Where an extension module hooks the creation of rv2cv ops to modify the subroutine lookup process, this now works correctly for bareword subroutine calls. This means that prototypes on subroutines referenced this way will be processed correctly. (Previously bareword subroutine names were initially looked up, for parsing purposes, by an unhookable mechanism, so extensions could only properly influence subroutine names that appeared with an & sigil.)

### A proper interface for pluggable Method Resolution Orders

As of Perl 5.12.0 there is a new interface for plugging and using method resolution orders other than the default linear depth first search. The C3 method resolution order added in 5.10.0 has been re-implemented as a plugin, without changing its Perl-space interface. See perlmroapi for more information.

### `\N` experimental regex escape

Perl now supports `\N`, a new regex escape which you can think of as the inverse of `\n`. It will match any character that is not a newline, independently from the presence or absence of the single line match modifier `/s`. It is not usable within a character class. `\N{3}` means to match 3 non-newlines; `\N{5,}` means to match at least 5. `\N{NAME}` still means the character or sequence named `NAME`, but `NAME` no longer can be things like `3`, or `5,`.

This will break a custom charnames translator which allows numbers for character names, as `\N{3}` will now mean to match 3 non-newline characters, and not the character whose name is `3`. (No name defined by the Unicode standard is a number, so only custom translators might be affected.)

Perl's developers are somewhat concerned about possible user confusion with the existing `\N{...}` construct which matches characters by their Unicode name. Consequently, this feature is experimental. We may remove it or change it in a backwards-incompatible way in Perl 5.14.

### DTrace support

Perl now has some support for DTrace. See ''DTrace support'' in *INSTALL*.

### Support for `configure_requires` in CPAN module metadata

Both `CPAN` and `CPANPLUS` now support the `configure_requires` keyword in the *META.yml* metadata file included in most recent CPAN distributions. This allows distribution authors to specify configuration prerequisites that must be installed before running *Makefile.PL* or *Build.PL*.

See the documentation for `ExtUtils::MakeMaker` or `Module::Build` for more on how to

specify `configure_requires` when creating a distribution for CPAN.

`each`, `keys`, `values` **are now more flexible**
The `each`, `keys`, `values` function can now operate on arrays.

`when` **as a statement modifier**
`when` is now allowed to be used as a statement modifier.

`$,` **flexibility**
The variable `$,` may now be tied.

**// in when clauses**
// now behaves like || in when clauses

**Enabling warnings from your shell environment**
You can now set `-W` from the `PERL5OPT` environment variable

`delete local`
`delete local` now allows you to locally delete a hash entry.

**New support for Abstract namespace sockets**
Abstract namespace sockets are Linux-specific socket type that live in AF_UNIX family, slightly abusing it to be able to use arbitrary character arrays as addresses: They start with nul byte and are not terminated by nul byte, but with the length passed to the **socket()** system call.

**32−bit limit on substr arguments removed**
The 32−bit limit on `substr` arguments has now been removed. The full range of the system's signed and unsigned integers is now available for the `pos` and `len` arguments.

## Potentially Incompatible Changes

**Deprecations warn by default**
Over the years, Perl's developers have deprecated a number of language features for a variety of reasons. Perl now defaults to issuing a warning if a deprecated language feature is used. Many of the deprecations Perl now warns you about have been deprecated for many years. You can find a list of what was deprecated in a given release of Perl in the `perl5xxdelta.pod` file for that release.

To disable this feature in a given lexical scope, you should use `no warnings 'deprecated';` For information about which language features are deprecated and explanations of various deprecation warnings, please see perldiag. See ''Deprecations'' below for the list of features and modules Perl's developers have deprecated as part of this release.

**Version number formats**
Acceptable version number formats have been formalized into ''strict'' and ''lax'' rules. `package NAME VERSION` takes a strict version number. `UNIVERSAL::VERSION` and the version object constructors take lax version numbers. Providing an invalid version will result in a fatal error. The version argument in `use NAME VERSION` is first parsed as a numeric literal or v−string and then passed to `UNIVERSAL::VERSION` (and must then pass the ''lax'' format test).

These formats are documented fully in the version module. To a first approximation, a ''strict'' version number is a positive decimal number (integer or decimal-fraction) without exponentiation or else a dotted-decimal v−string with a leading 'v' character and at least three components. A ''lax'' version number allows v−strings with fewer than three components or without a leading 'v'. Under ''lax'' rules, both decimal and dotted-decimal versions may have a trailing ''alpha'' component separated by an underscore character after a fractional or dotted-decimal component.

The version module adds `version::is_strict` and `version::is_lax` functions to check a scalar against these rules.

`@INC` **reorganization**
In `@INC`, `ARCHLIB` and `PRIVLIB` now occur after the current version's `site_perl` and `vendor_perl`. Modules installed into `site_perl` and `vendor_perl` will now be loaded in preference to those installed in `ARCHLIB` and `PRIVLIB`.

**REGEXPs are now first class**
Internally, Perl now treats compiled regular expressions (such as those created with `qr//`) as first class entities. Perl modules which serialize, deserialize or otherwise have deep interaction with Perl's internal data structures need to be updated for this change. Most affected CPAN modules have already been updated as of this writing.

### Switch statement changes

The `given`/`when` switch statement handles complex statements better than Perl 5.10.0 did (These enhancements are also available in 5.10.1 and subsequent 5.10 releases.) There are two new cases where `when` now interprets its argument as a boolean, instead of an expression to be used in a smart match:

flip-flop operators

The `..` and `...` flip-flop operators are now evaluated in boolean context, following their usual semantics; see ‘‘Range Operators’’ in perlop.

Note that, as in perl 5.10.0, `when (1..10)` will not work to test whether a given value is an integer between 1 and 10; you should use `when ([1..10])` instead (note the array reference).

However, contrary to 5.10.0, evaluating the flip-flop operators in boolean context ensures it can now be useful in a `when()`, notably for implementing bistable conditions, like in:

```
when (/^=begin/ .. /^=end/) {
  # do something
}
```

defined-or operator

A compound expression involving the defined-or operator, as in `when (expr1 // expr2)`, will be treated as boolean if the first expression is boolean. (This just extends the existing rule that applies to the regular or operator, as in `when (expr1 || expr2)`.)

### Smart match changes

Since Perl 5.10.0, Perl’s developers have made a number of changes to the smart match operator. These, of course, also alter the behaviour of the switch statements where smart matching is implicitly used. These changes were also made for the 5.10.1 release, and will remain in subsequent 5.10 releases.

*Changes to type-based dispatch*

The smart match operator `~~` is no longer commutative. The behaviour of a smart match now depends primarily on the type of its right hand argument. Moreover, its semantics have been adjusted for greater consistency or usefulness in several cases. While the general backwards compatibility is maintained, several changes must be noted:

- Code references with an empty prototype are no longer treated specially. They are passed an argument like the other code references (even if they choose to ignore it).
- `%hash ~~ sub {}` and `@array ~~ sub {}` now test that the subroutine returns a true value for each key of the hash (or element of the array), instead of passing the whole hash or array as a reference to the subroutine.
- Due to the commutativity breakage, code references are no longer treated specially when appearing on the left of the `~~` operator, but like any vulgar scalar.
- `undef ~~ %hash` is always false (since `undef` can’t be a key in a hash). No implicit conversion to `""` is done (as was the case in perl 5.10.0).
- `$scalar ~~ @array` now always distributes the smart match across the elements of the array. It’s true if one element in `@array` verifies `$scalar ~~ $element`. This is a generalization of the old behaviour that tested whether the array contained the scalar.

The full dispatch table for the smart match operator is given in ‘‘Smart matching in detail’’ in perlsyn.

*Smart match and overloading*

According to the rule of dispatch based on the rightmost argument type, when an object overloading `~~` appears on the right side of the operator, the overload routine will always be called (with a 3rd argument set to a true value, see overload.) However, when the object will appear on the left, the overload routine will be called only when the rightmost argument is a simple scalar. This way, distributivity of smart match across arrays is not broken, as well as the other behaviours with complex types (coderefs, hashes, regexes). Thus, writers of overloading routines for smart match mostly need to worry only with comparing against a scalar, and possibly with stringification overloading; the other common cases will be automatically handled consistently.

`~~` will now refuse to work on objects that do not overload it (in order to avoid relying on the object’s

underlying structure). (However, if the object overloads the stringification or the numification operators, and if overload fallback is active, it will be used instead, as usual.)

### Other potentially incompatible changes

- The definitions of a number of Unicode properties have changed to match those of the current Unicode standard. These are listed above under ‘‘Unicode overhaul’’. This change may break code that expects the old definitions.
- The boolkeys op has moved to the group of hash ops. This breaks binary compatibility.
- Filehandles are now always blessed into `IO::File`.

  The previous behaviour was to bless Filehandles into FileHandle (an empty proxy class) if it was loaded into memory and otherwise to bless them into `IO::Handle`.
- The semantics of `use feature :5.10*` have changed slightly. See ‘‘Modules and Pragmata’’ for more information.
- Perl’s developers now use git, rather than Perforce. This should be a purely internal change only relevant to people actively working on the core. However, you may see minor difference in perl as a consequence of the change. For example in some of details of the output of `perl -V`. See perlrepository for more information.
- As part of the `Test::Harness` 2.x to 3.x upgrade, the experimental `Test::Harness::Straps` module has been removed. See ‘‘Modules and Pragmata’’ for more details.
- As part of the `ExtUtils::MakeMaker` upgrade, the `ExtUtils::MakeMaker::bytes` and `ExtUtils::MakeMaker::vmsish` modules have been removed from this distribution.
- `Module::CoreList` no longer contains the `%:patchlevel` hash.
- `length undef` now returns undef.
- Unsupported private C API functions are now declared ‘‘static’’ to prevent leakage to Perl’s public API.
- To support the bootstrapping process, *miniperl* no longer builds with UTF-8 support in the regexp engine.

  This allows a build to complete with PERL_UNICODE set and a UTF-8 locale. Without this there’s a bootstrapping problem, as miniperl can’t load the UTF-8 components of the regexp engine, because they’re not yet built.
- *miniperl*’s `@INC` is now restricted to just `-I...`, the split of `$ENV{PERL5LIB}`, and `"."`
- A space or a newline is now required after a `"#line XXX"` directive.
- Tied filehandles now have an additional method EOF which provides the EOF type.
- To better match all other flow control statements, `foreach` may no longer be used as an attribute.
- Perl’s command-line switch ‘‘-P’’, which was deprecated in version 5.10.0, has now been removed. The CPAN module `Filter::cpp` can be used as an alternative.

## Deprecations

From time to time, Perl’s developers find it necessary to deprecate features or modules we’ve previously shipped as part of the core distribution. We are well aware of the pain and frustration that a backwards-incompatible change to Perl can cause for developers building or maintaining software in Perl. You can be sure that when we deprecate a functionality or syntax, it isn’t a choice we make lightly. Sometimes, we choose to deprecate functionality or syntax because it was found to be poorly designed or implemented. Sometimes, this is because they’re holding back other features or causing performance problems. Sometimes, the reasons are more complex. Wherever possible, we try to keep deprecated functionality available to developers in its previous form for at least one major release. So long as a deprecated feature isn’t actively disrupting our ability to maintain and extend Perl, we’ll try to leave it in place as long as possible.

The following items are now deprecated:

suidperl
: `suidperl` is no longer part of Perl. It used to provide a mechanism to emulate setuid permission bits on systems that don't support it properly.

Use of `:=` to mean an empty attribute list
: An accident of Perl's parser meant that these constructions were all equivalent:

```
my $pi := 4;
my $pi : = 4;
my $pi :  = 4;
```

with the `:` being treated as the start of an attribute list, which ends before the `=`. As whitespace is not significant here, all are parsed as an empty attribute list, hence all the above are equivalent to, and better written as

```
my $pi = 4;
```

because no attribute processing is done for an empty list.

As is, this meant that `:=` cannot be used as a new token, without silently changing the meaning of existing code. Hence that particular form is now deprecated, and will become a syntax error. If it is absolutely necessary to have empty attribute lists (for example, because of a code generator) then avoid the warning by adding a space before the `=`.

`UNIVERSAL->import()`
: The method `UNIVERSAL->import()` is now deprecated. Attempting to pass import arguments to a `use UNIVERSAL` statement will result in a deprecation warning.

Use of "goto" to jump into a construct
: Using `goto` to jump from an outer scope into an inner scope is now deprecated. This rare use case was causing problems in the implementation of scopes.

Custom character names in \N{name} that don't look like names
: In `\N{`*`name`*`}`, *name* can be just about anything. The standard Unicode names have a very limited domain, but a custom name translator could create names that are, for example, made up entirely of punctuation symbols. It is now deprecated to make names that don't begin with an alphabetic character, and aren't alphanumeric or contain other than a very few other characters, namely spaces, dashes, parentheses and colons. Because of the added meaning of `\N` (See `"\N" experimental regex escape`), names that look like curly brace −enclosed quantifiers won't work. For example, `\N{3,4}` now means to match 3 to 4 non-newlines; before a custom name `3,4` could have been created.

Deprecated Modules
: The following modules will be removed from the core distribution in a future release, and should be installed from CPAN instead. Distributions on CPAN which require these should add them to their prerequisites. The core versions of these modules warnings will issue a deprecation warning.

If you ship a packaged version of Perl, either alone or as part of a larger system, then you should carefully consider the repercussions of core module deprecations. You may want to consider shipping your default build of Perl with packages for some or all deprecated modules which install into `vendor` or `site` perl library directories. This will inhibit the deprecation warnings.

Alternatively, you may want to consider patching *lib/deprecate.pm* to provide deprecation warnings specific to your packaging system or distribution of Perl, consistent with how your packaging system or distribution manages a staged transition from a release where the installation of a single package provides the given functionality, to a later release where the system administrator needs to know to install multiple packages to get that same functionality.

You can silence these deprecation warnings by installing the modules in question from CPAN. To install the latest version of all of them, just install `Task::Deprecations::5_12`.

Class::ISA
Pod::Plainer
Shell

Switch
: Switch is buggy and should be avoided. You may find Perl's new `given`/`when` feature a suitable replacement. See ''Switch statements'' in perlsyn for more information.

Assignment to $[
Use of the attribute :locked on subroutines
Use of ''locked'' with the attributes pragma
Use of ''unique'' with the attributes pragma
Perl_pmflag
: `Perl_pmflag` is no longer part of Perl's public API. Calling it now generates a deprecation warning, and it will be removed in a future release. Although listed as part of the API, it was never documented, and only ever used in *toke.c*, and prior to 5.10, *regcomp.c*. In core, it has been replaced by a static function.

Numerous Perl 4−era libraries
: *termcap.pl*, *tainted.pl*, *stat.pl*, *shellwords.pl*, *pwd.pl*, *open3.pl*, *open2.pl*, *newgetopt.pl*, *look.pl*, *find.pl*, *finddepth.pl*, *importenv.pl*, *hostname.pl*, *getopts.pl*, *getopt.pl*, *getcwd.pl*, *flush.pl*, *fastcwd.pl*, *exceptions.pl*, *ctime.pl*, *complete.pl*, *cacheout.pl*, *bigrat.pl*, *bigint.pl*, *bigfloat.pl*, *assert.pl*, *abbrev.pl*, *dotsh.pl*, and *timelocal.pl* are all now deprecated. Earlier, Perl's developers intended to remove these libraries from Perl's core for the 5.14.0 release.

  During final testing before the release of 5.12.0, several developers discovered current production code using these ancient libraries, some inside the Perl core itself. Accordingly, the pumpking granted them a stay of execution. They will begin to warn about their deprecation in the 5.14.0 release and will be removed in the 5.16.0 release.

## Unicode overhaul

Perl's developers have made a concerted effort to update Perl to be in sync with the latest Unicode standard. Changes for this include:

Perl can now handle every Unicode character property. New documentation, perluniprops, lists all available non-Unihan character properties. By default, perl does not expose Unihan, deprecated or Unicode-internal properties. See below for more details on these; there is also a section in the pod listing them, and explaining why they are not exposed.

Perl now fully supports the Unicode compound-style of using `=` and `:` in writing regular expressions: `\p{property=value}` and `\p{property:value}` (both of which mean the same thing).

Perl now fully supports the Unicode loose matching rules for text between the braces in `\p{...}` constructs. In addition, Perl allows underscores between digits of numbers.

Perl now accepts all the Unicode-defined synonyms for properties and property values.

`qr/\X/`, which matches a Unicode logical character, has been expanded to work better with various Asian languages. It now is defined as an *extended grapheme cluster*. (See <http://www.unicode.org/reports/tr29/>). Anything matched previously and that made sense will continue to be accepted. Additionally:

- `\X` will not break apart a `CR LF` sequence.
- `\X` will now match a sequence which includes the `ZWJ` and `ZWNJ` characters.
- `\X` will now always match at least one character, including an initial mark. Marks generally come after a base character, but it is possible in Unicode to have them in isolation, and `\X` will now handle that case, for example at the beginning of a line, or after a `ZWSP`. And this is the part where `\X` doesn't match the things that it used to that don't make sense. Formerly, for example, you could have the nonsensical case of an accented LF.
- `\X` will now match a (Korean) Hangul syllable sequence, and the Thai and Lao exception cases.

Otherwise, this change should be transparent for the non-affected languages.

`\p{...}` matches using the Canonical_Combining_Class property were completely broken in previous releases of Perl. They should now work correctly.

Before Perl 5.12, the Unicode `Decomposition_Type=Compat` property and a Perl extension had the same name, which led to neither matching all the correct values (with more than 100 mistakes in one, and several thousand in the other). The Perl extension has now been renamed to be

`Decomposition_Type=Noncanonical` (short: `dt=noncanon`). It has the same meaning as was previously intended, namely the union of all the non-canonical Decomposition types, with Unicode `Compat` being just one of those.

`\p{Decomposition_Type=Canonical}` now includes the Hangul syllables.

`\p{Uppercase}` and `\p{Lowercase}` now work as the Unicode standard says they should. This means they each match a few more characters than they used to.

`\p{Cntrl}` now matches the same characters as `\p{Control}`. This means it no longer will match Private Use (gc=co), Surrogates (gc=cs), nor Format (gc=cf) code points. The Format code points represent the biggest possible problem. All but 36 of them are either officially deprecated or strongly discouraged from being used. Of those 36, likely the most widely used are the soft hyphen (U+00AD), and BOM, ZWSP, ZWNJ, WJ, and similar characters, plus bidirectional controls.

`\p{Alpha}` now matches the same characters as `\p{Alphabetic}`. Before 5.12, Perl's definition included a number of things that aren't really alpha (all marks) while omitting many that were. The definitions of `\p{Alnum}` and `\p{Word}` depend on Alpha's definition and have changed accordingly.

`\p{Word}` no longer incorrectly matches non-word characters such as fractions.

`\p{Print}` no longer matches the line control characters: Tab, LF, CR, FF, VT, and NEL. This brings it in line with standards and the documentation.

`\p{XDigit}` now matches the same characters as `\p{Hex_Digit}`. This means that in addition to the characters it currently matches, `[A-Fa-f0-9]`, it will also match the 22 fullwidth equivalents, for example U+FF10: FULLWIDTH DIGIT ZERO.

The Numeric type property has been extended to include the Unihan characters.

There is a new Perl extension, the 'Present_In', or simply 'In', property. This is an extension of the Unicode Age property, but `\p{In=5.0}` matches any code point whose usage has been determined *as of* Unicode version 5.0. The `\p{Age=5.0}` only matches code points added in *precisely* version 5.0.

A number of properties now have the correct values for unassigned code points. The affected properties are Bidi_Class, East_Asian_Width, Joining_Type, Decomposition_Type, Hangul_Syllable_Type, Numeric_Type, and Line_Break.

The Default_Ignorable_Code_Point, ID_Continue, and ID_Start properties are now up to date with current Unicode definitions.

Earlier versions of Perl erroneously exposed certain properties that are supposed to be Unicode internal-only. Use of these in regular expressions will now generate, if enabled, a deprecation warning message. The properties are: Other_Alphabetic, Other_Default_Ignorable_Code_Point, Other_Grapheme_Extend, Other_ID_Continue, Other_ID_Start, Other_Lowercase, Other_Math, and Other_Uppercase.

It is now possible to change which Unicode properties Perl understands on a per-installation basis. As mentioned above, certain properties are turned off by default. These include all the Unihan properties (which should be accessible via the CPAN module Unicode::Unihan) and any deprecated or Unicode internal-only property that Perl has never exposed.

The generated files in the `lib/unicore/To` directory are now more clearly marked as being stable, directly usable by applications. New hash entries in them give the format of the normal entries, which allows for easier machine parsing. Perl can generate files in this directory for any property, though most are suppressed. You can find instructions for changing which are written in perluniprops.

## Modules and Pragmata

### New Modules and Pragmata

#### `autodie`

`autodie` is a new lexically-scoped alternative for the `Fatal` module. The bundled version is 2.06_01. Note that in this release, using a string eval when `autodie` is in effect can cause the autodie behaviour to leak into the surrounding scope. See "BUGS" in autodie for more details.

Version 2.06_01 has been added to the Perl core.

`Compress::Raw::Bzip2`
Version 2.024 has been added to the Perl core.

`overloading`
`overloading` allows you to lexically disable or enable overloading for some or all operations.

Version 0.001 has been added to the Perl core.

`parent`
`parent` establishes an ISA relationship with base classes at compile time. It provides the key feature of `base` without further unwanted behaviors.

Version 0.223 has been added to the Perl core.

`Parse::CPAN::Meta`
Version 1.40 has been added to the Perl core.

`VMS::DCLsym`
Version 1.03 has been added to the Perl core.

`VMS::Stdio`
Version 2.4 has been added to the Perl core.

`XS::APItest::KeywordRPN`
Version 0.003 has been added to the Perl core.

### Updated Pragmata

`base`
Upgraded from version 2.13 to 2.15.

`bignum`
Upgraded from version 0.22 to 0.23.

`charnames`
`charnames` now contains the Unicode *NameAliases.txt* database file. This has the effect of adding some extra `\N` character names that formerly wouldn't have been recognised; for example, `"\N{LATIN CAPITAL LETTER GHA}"`.

Upgraded from version 1.06 to 1.07.

`constant`
Upgraded from version 1.13 to 1.20.

`diagnostics`
`diagnostics` now supports %.0f formatting internally.

`diagnostics` no longer suppresses `Use of uninitialized value in range (or flip)` warnings. [perl #71204]

Upgraded from version 1.17 to 1.19.

`feature`
In `feature`, the meaning of the `:5.10` and `:5.10.X` feature bundles has changed slightly. The last component, if any (i.e. `X`) is simply ignored. This is predicated on the assumption that new features will not, in general, be added to maintenance releases. So `:5.10` and `:5.10.X` have identical effect. This is a change to the behaviour documented for 5.10.0.

`feature` now includes the `unicode_strings` feature:

```
use feature "unicode_strings";
```

This pragma turns on Unicode semantics for the case-changing operations (`uc`, `lc`, `ucfirst`, `lcfirst`) on strings that don't have the internal UTF-8 flag set, but that contain single-byte characters between 128 and 255.

Upgraded from version 1.11 to 1.16.

`less`
`less` now includes the `stash_name` method to allow subclasses of `less` to pick where in %^H to store their stash.

Upgraded from version 0.02 to 0.03.

`lib`
Upgraded from version 0.5565 to 0.62.

`mro`
`mro` is now implemented as an XS extension. The documented interface has not changed. Code relying on the implementation detail that some `mro::` methods happened to be available at all times gets to ‘‘keep both pieces’’.

Upgraded from version 1.00 to 1.02.

`overload`
`overload` now allow overloading of ’qr’.

Upgraded from version 1.06 to 1.10.

`threads`
Upgraded from version 1.67 to 1.75.

`threads::shared`
Upgraded from version 1.14 to 1.32.

`version`
`version` now has support for ‘‘Version number formats’’ as described earlier in this document and in its own documentation.

Upgraded from version 0.74 to 0.82.

`warnings`
`warnings` has a new `warnings::fatal_enabled()` function. It also includes a new `illegalproto` warning category. See also ‘‘New or Changed Diagnostics’’ for this change.

Upgraded from version 1.06 to 1.09.

### Updated Modules

`Archive::Extract`
Upgraded from version 0.24 to 0.38.

`Archive::Tar`
Upgraded from version 1.38 to 1.54.

`Attribute::Handlers`
Upgraded from version 0.79 to 0.87.

`AutoLoader`
Upgraded from version 5.63 to 5.70.

`B::Concise`
Upgraded from version 0.74 to 0.78.

`B::Debug`
Upgraded from version 1.05 to 1.12.

`B::Deparse`
Upgraded from version 0.83 to 0.96.

`B::Lint`
Upgraded from version 1.09 to 1.11_01.

`CGI`
Upgraded from version 3.29 to 3.48.

`Class::ISA`
Upgraded from version 0.33 to 0.36.

NOTE: `Class::ISA` is deprecated and may be removed from a future version of Perl.

`Compress::Raw::Zlib`
Upgraded from version 2.008 to 2.024.

`CPAN`
: Upgraded from version 1.9205 to 1.94_56.

`CPANPLUS`
: Upgraded from version 0.84 to 0.90.

`CPANPLUS::Dist::Build`
: Upgraded from version 0.06_02 to 0.46.

`Data::Dumper`
: Upgraded from version 2.121_14 to 2.125.

`DB_File`
: Upgraded from version 1.816_1 to 1.820.

`Devel::PPPort`
: Upgraded from version 3.13 to 3.19.

`Digest`
: Upgraded from version 1.15 to 1.16.

`Digest::MD5`
: Upgraded from version 2.36_01 to 2.39.

`Digest::SHA`
: Upgraded from version 5.45 to 5.47.

`Encode`
: Upgraded from version 2.23 to 2.39.

`Exporter`
: Upgraded from version 5.62 to 5.64_01.

`ExtUtils::CBuilder`
: Upgraded from version 0.21 to 0.27.

`ExtUtils::Command`
: Upgraded from version 1.13 to 1.16.

`ExtUtils::Constant`
: Upgraded from version 0.2 to 0.22.

`ExtUtils::Install`
: Upgraded from version 1.44 to 1.55.

`ExtUtils::MakeMaker`
: Upgraded from version 6.42 to 6.56.

`ExtUtils::Manifest`
: Upgraded from version 1.51_01 to 1.57.

`ExtUtils::ParseXS`
: Upgraded from version 2.18_02 to 2.21.

`File::Fetch`
: Upgraded from version 0.14 to 0.24.

`File::Path`
: Upgraded from version 2.04 to 2.08_01.

`File::Temp`
: Upgraded from version 0.18 to 0.22.

`Filter::Simple`
: Upgraded from version 0.82 to 0.84.

`Filter::Util::Call`
: Upgraded from version 1.07 to 1.08.

`Getopt::Long`
: Upgraded from version 2.37 to 2.38.

`IO` Upgraded from version 1.23_01 to 1.25_02.

`IO::Zlib`
Upgraded from version 1.07 to 1.10.

`IPC::Cmd`
Upgraded from version 0.40_1 to 0.54.

`IPC::SysV`
Upgraded from version 1.05 to 2.01.

`Locale::Maketext`
Upgraded from version 1.12 to 1.14.

`Locale::Maketext::Simple`
Upgraded from version 0.18 to 0.21.

`Log::Message`
Upgraded from version 0.01 to 0.02.

`Log::Message::Simple`
Upgraded from version 0.04 to 0.06.

`Math::BigInt`
Upgraded from version 1.88 to 1.89_01.

`Math::BigInt::FastCalc`
Upgraded from version 0.16 to 0.19.

`Math::BigRat`
Upgraded from version 0.21 to 0.24.

`Math::Complex`
Upgraded from version 1.37 to 1.56.

`Memoize`
Upgraded from version 1.01_02 to 1.01_03.

`MIME::Base64`
Upgraded from version 3.07_01 to 3.08.

`Module::Build`
Upgraded from version 0.2808_01 to 0.3603.

`Module::CoreList`
Upgraded from version 2.12 to 2.29.

`Module::Load`
Upgraded from version 0.12 to 0.16.

`Module::Load::Conditional`
Upgraded from version 0.22 to 0.34.

`Module::Loaded`
Upgraded from version 0.01 to 0.06.

`Module::Pluggable`
Upgraded from version 3.6 to 3.9.

`Net::Ping`
Upgraded from version 2.33 to 2.36.

`NEXT`
Upgraded from version 0.60_01 to 0.64.

`Object::Accessor`
Upgraded from version 0.32 to 0.36.

`Package::Constants`
Upgraded from version 0.01 to 0.02.

`PerlIO`
Upgraded from version 1.04 to 1.06.

`Pod::Parser`
Upgraded from version 1.35 to 1.37.

`Pod::Perldoc`
Upgraded from version 3.14_02 to 3.15_02.

`Pod::Plainer`
Upgraded from version 0.01 to 1.02.

NOTE: `Pod::Plainer` is deprecated and may be removed from a future version of Perl.

`Pod::Simple`
Upgraded from version 3.05 to 3.13.

`Safe`
Upgraded from version 2.12 to 2.22.

`SelfLoader`
Upgraded from version 1.11 to 1.17.

`Storable`
Upgraded from version 2.18 to 2.22.

`Switch`
Upgraded from version 2.13 to 2.16.

NOTE: `Switch` is deprecated and may be removed from a future version of Perl.

`Sys::Syslog`
Upgraded from version 0.22 to 0.27.

`Term::ANSIColor`
Upgraded from version 1.12 to 2.02.

`Term::UI`
Upgraded from version 0.18 to 0.20.

`Test`
Upgraded from version 1.25 to 1.25_02.

`Test::Harness`
Upgraded from version 2.64 to 3.17.

`Test::Simple`
Upgraded from version 0.72 to 0.94.

`Text::Balanced`
Upgraded from version 2.0.0 to 2.02.

`Text::ParseWords`
Upgraded from version 3.26 to 3.27.

`Text::Soundex`
Upgraded from version 3.03 to 3.03_01.

`Thread::Queue`
Upgraded from version 2.00 to 2.11.

`Thread::Semaphore`
Upgraded from version 2.01 to 2.09.

`Tie::RefHash`
Upgraded from version 1.37 to 1.38.

`Time::HiRes`
Upgraded from version 1.9711 to 1.9719.

`Time::Local`
Upgraded from version 1.18 to 1.1901_01.

`Time::Piece`
: Upgraded from version 1.12 to 1.15.

`Unicode::Collate`
: Upgraded from version 0.52 to 0.52_01.

`Unicode::Normalize`
: Upgraded from version 1.02 to 1.03.

`Win32`
: Upgraded from version 0.34 to 0.39.

`Win32API::File`
: Upgraded from version 0.1001_01 to 0.1101.

`XSLoader`
: Upgraded from version 0.08 to 0.10.

### Removed Modules and Pragmata

`attrs`
: Removed from the Perl core. Prior version was 1.02.

`CPAN::API::HOWTO`
: Removed from the Perl core. Prior version was 'undef'.

`CPAN::DeferedCode`
: Removed from the Perl core. Prior version was 5.50.

`CPANPLUS::inc`
: Removed from the Perl core. Prior version was 'undef'.

`DCLsym`
: Removed from the Perl core. Prior version was 1.03.

`ExtUtils::MakeMaker::bytes`
: Removed from the Perl core. Prior version was 6.42.

`ExtUtils::MakeMaker::vmsish`
: Removed from the Perl core. Prior version was 6.42.

`Stdio`
: Removed from the Perl core. Prior version was 2.3.

`Test::Harness::Assert`
: Removed from the Perl core. Prior version was 0.02.

`Test::Harness::Iterator`
: Removed from the Perl core. Prior version was 0.02.

`Test::Harness::Point`
: Removed from the Perl core. Prior version was 0.01.

`Test::Harness::Results`
: Removed from the Perl core. Prior version was 0.01.

`Test::Harness::Straps`
: Removed from the Perl core. Prior version was 0.26_01.

`Test::Harness::Util`
: Removed from the Perl core. Prior version was 0.01.

`XSSymSet`
: Removed from the Perl core. Prior version was 1.1.

### Deprecated Modules and Pragmata

See ''Deprecated Modules'' above.

## Documentation

### New Documentation

- perlhaiku contains instructions on how to build perl for the Haiku platform.

- perlmroapi describes the new interface for pluggable Method Resolution Orders.
- perlperf, by Richard Foley, provides an introduction to the use of performance and optimization techniques which can be used with particular reference to perl programs.
- perlrepository describes how to access the perl source using the *git* version control system.
- perlpolicy extends the ‘‘Social contract about contributed modules’’ into the beginnings of a document on Perl porting policies.

### Changes to Existing Documentation

- The various large *Changes** files (which listed every change made to perl over the last 18 years) have been removed, and replaced by a small file, also called *Changes*, which just explains how that same information may be extracted from the git version control system.
- *Porting/patching.pod* has been deleted, as it mainly described interacting with the old Perforce-based repository, which is now obsolete. Information still relevant has been moved to perlrepository.
- The syntax `unless (EXPR) BLOCK else BLOCK` is now documented as valid, as is the syntax `unless (EXPR) BLOCK elsif (EXPR) BLOCK ... else BLOCK`, although actually using the latter may not be the best idea for the readability of your source code.
- Documented −X overloading.
- Documented that `when()` treats specially most of the filetest operators
- Documented `when` as a syntax modifier.
- Eliminated ‘‘Old Perl threads tutorial’’, which described 5005 threads.

  *pod/perlthrtut.pod* is the same material reworked for ithreads.
- Correct previous documentation: v−strings are not deprecated

  With version objects, we need them to use MODULE VERSION syntax. This patch removes the deprecation notice.
- Security contact information is now part of perlsec.
- A significant fraction of the core documentation has been updated to clarify the behavior of Perl’s Unicode handling.

  Much of the remaining core documentation has been reviewed and edited for clarity, consistent use of language, and to fix the spelling of Tom Christiansen’s name.
- The Pod specification (perlpodspec) has been updated to bring the specification in line with modern usage already supported by most Pod systems. A parameter string may now follow the format name in a ‘‘begin/end’’ region. Links to URIs with a text description are now allowed. The usage of `L<"section">` has been marked as deprecated.
- if.pm has been documented in ‘‘use’’ in perlfunc as a means to get conditional loading of modules despite the implicit BEGIN block around `use`.
- The documentation for `$1` in perlvar.pod has been clarified.
- `\N{U+`*`code point`*`}` is now documented.

## Selected Performance Enhancements

- A new internal cache means that `isa()` will often be faster.
- The implementation of `C3` Method Resolution Order has been optimised − linearisation for classes with single inheritance is 40% faster. Performance for multiple inheritance is unchanged.
- Under `use locale`, the locale-relevant information is now cached on read-only values, such as the list returned by `keys %hash`. This makes operations such as `sort keys %hash` in the scope of `use locale` much faster.
- Empty `DESTROY` methods are no longer called.
- `Perl_sv_utf8_upgrade()` is now faster.

- `keys` on empty hash is now faster.
- `if (%foo)` has been optimized to be faster than `if (keys %foo)`.
- The string repetition operator (`$str x $num`) is now several times faster when `$str` has length one or `$num` is large.
- Reversing an array to itself (as in `@a = reverse @a`) in void context now happens in-place and is several orders of magnitude faster than it used to be. It will also preserve non-existent elements whenever possible, i.e. for non magical arrays or tied arrays with `EXISTS` and `DELETE` methods.

## Installation and Configuration Improvements

- perlapi, perlintern, perlmodlib and perltoc are now all generated at build time, rather than being shipped as part of the release.
- If `vendorlib` and `vendorarch` are the same, then they are only added to `@INC` once.
- `$Config{usedevel}` and the C−level `PERL_USE_DEVEL` are now defined if perl is built with `-Dusedevel`.
- *Configure* will enable use of `-fstack-protector`, to provide protection against stack-smashing attacks, if the compiler supports it.
- *Configure* will now determine the correct prototypes for re-entrant functions and for `gconvert` if you are using a C++ compiler rather than a C compiler.
- On Unix, if you build from a tree containing a git repository, the configuration process will note the commit hash you have checked out, for display in the output of `perl -v` and `perl -V`. Unpushed local commits are automatically added to the list of local patches displayed by `perl -V`.
- Perl now supports SystemTap's `dtrace` compatibility layer and an issue with linking `miniperl` has been fixed in the process.
- perldoc now uses `less -R` instead of `less` for improved behaviour in the face of `groff`'s new usage of ANSI escape codes.
- `perl -V` now reports use of the compile-time options `USE_PERL_ATOF` and `USE_ATTRIBUTES_FOR_PERLIO`.
- As part of the flattening of *ext*, all extensions on all platforms are built by *make_ext.pl*. This replaces the Unix-specific *ext/util/make_ext*, VMS-specific *make_ext.com* and Win32−specific *win32/buildext.pl*.

## Internal Changes

Each release of Perl sees numerous internal changes which shouldn't affect day to day usage but may still be notable for developers working with Perl's source code.

- The J.R.R. Tolkien quotes at the head of C source file have been checked and proper citations added, thanks to a patch from Tom Christiansen.
- The internal structure of the dual-life modules traditionally found in the *lib/* and *ext/* directories in the perl source has changed significantly. Where possible, dual-lifed modules have been extracted from *lib/* and *ext/*.

  Dual-lifed modules maintained by Perl's developers as part of the Perl core now live in *dist/*. Dual-lifed modules maintained primarily on CPAN now live in *cpan/*. When reporting a bug in a module located under *cpan/*, please send your bug report directly to the module's bug tracker or author, rather than Perl's bug tracker.
- `\N{...}` now compiles better, always forces UTF−8 internal representation

  Perl's developers have fixed several problems with the recognition of `\N{...}` constructs. As part of this, perl will store any scalar or regex containing `\N{`*`name`*`}` or `\N{U+`*`code point`*`}` in its definition in UTF−8 format. (This was true previously for all occurrences of `\N{`*`name`*`}` that did not use a custom translator, but now it's always true.)

- Perl_magic_setmglob now knows about globs, fixing RT #71254.
- SVt_RV no longer exists. RVs are now stored in IVs.
- Perl_vcroak() now accepts a null first argument. In addition, a full audit was made of the "not NULL" compiler annotations, and those for several other internal functions were corrected.
- New macros dSAVEDERRNO, dSAVE_ERRNO, SAVE_ERRNO, RESTORE_ERRNO have been added to formalise the temporary saving of the errno variable.
- The function Perl_sv_insert_flags has been added to augment Perl_sv_insert.
- The function Perl_newSV_type(type) has been added, equivalent to Perl_newSV() followed by Perl_sv_upgrade(type).
- The function Perl_newSVpvn_flags() has been added, equivalent to Perl_newSVpvn() and then performing the action relevant to the flag.

  Two flag bits are currently supported.

  - SVf_UTF8 will call SvUTF8_on() for you. (Note that this does not convert a sequence of ISO 8859-1 characters to UTF-8). A wrapper, newSVpvn_utf8() is available for this.
  - SVs_TEMP now calls Perl_sv_2mortal() on the new SV.

  There is also a wrapper that takes constant strings, newSVpvs_flags().
- The function Perl_croak_xs_usage has been added as a wrapper to Perl_croak.
- Perl now exports the functions PerlIO_find_layer and PerlIO_list_alloc.
- PL_na has been exterminated from the core code, replaced by local STRLEN temporaries, or *_nolen() calls. Either approach is faster than PL_na, which is a pointer dereference into the interpreter structure under ithreads, and a global variable otherwise.
- Perl_mg_free() used to leave freed memory accessible via SvMAGIC() on the scalar. It now updates the linked list to remove each piece of magic as it is freed.
- Under ithreads, the regex in PL_reg_curpm is now reference counted. This eliminates a lot of hackish workarounds to cope with it not being reference counted.
- Perl_mg_magical() would sometimes incorrectly turn on SvRMAGICAL(). This has been fixed.
- The *public* IV and NV flags are now not set if the string value has trailing "garbage". This behaviour is consistent with not setting the public IV or NV flags if the value is out of range for the type.
- Uses of Nullav, Nullcv, Nullhv, Nullop, Nullsv etc have been replaced by NULL in the core code, and non-dual-life modules, as NULL is clearer to those unfamiliar with the core code.
- A macro MUTABLE_PTR(p) has been added, which on (non-pedantic) gcc will not cast away const, returning a void *. Macros MUTABLE_SV(av), MUTABLE_SV(cv) etc build on this, casting to AV * etc without casting away const. This allows proper compile-time auditing of const correctness in the core, and helped picked up some errors (now fixed).
- Macros mPUSHs() and mXPUSHs() have been added, for pushing SVs on the stack and mortalizing them.
- Use of the private structure mro_meta has changed slightly. Nothing outside the core should be accessing this directly anyway.
- A new tool, *Porting/expand-macro.pl* has been added, that allows you to view how a C preprocessor macro would be expanded when compiled. This is handy when trying to decode the macro hell that is the perl guts.

## Testing

### Testing improvements

#### Parallel tests

The core distribution can now run its regression tests in parallel on Unix-like platforms. Instead of running make test, set TEST_JOBS in your environment to the number of tests to run in parallel, and run make test_harness. On a Bourne-like shell, this can be done as

```
TEST_JOBS=3 make test_harness  # Run 3 tests in parallel
```

An environment variable is used, rather than parallel make itself, because TAP::Harness needs to be able to schedule individual non-conflicting test scripts itself, and there is no standard interface to `make` utilities to interact with their job schedulers.

Note that currently some test scripts may fail when run in parallel (most notably `ext/IO/t/io_dir.t`). If necessary run just the failing scripts again sequentially and see if the failures go away.

Test harness flexibility
: It's now possible to override `PERL5OPT` and friends in *t/TEST*

Test watchdog
: Several tests that have the potential to hang forever if they fail now incorporate a ''watchdog'' functionality that will kill them after a timeout, which helps ensure that `make test` and `make test_harness` run to completion automatically.

### New Tests

Perl's developers have added a number of new tests to the core. In addition to the items listed below, many modules updated from CPAN incorporate new tests.

- Significant cleanups to core tests to ensure that language and interpreter features are not used before they're tested.
- `make test_porting` now runs a number of important pre-commit checks which might be of use to anyone working on the Perl core.
- *t/porting/podcheck.t* automatically checks the well-formedness of POD found in all .pl, .pm and .pod files in the *MANIFEST*, other than in dual-lifed modules which are primarily maintained outside the Perl core.
- *t/porting/manifest.t* now tests that all files listed in MANIFEST are present.
- *t/op/while_readdir.t* tests that a bare readdir in while loop sets `$_`.
- *t/comp/retainedlines.t* checks that the debugger can retain source lines from `eval`.
- *t/io/perlio_fail.t* checks that bad layers fail.
- *t/io/perlio_leaks.t* checks that PerlIO layers are not leaking.
- *t/io/perlio_open.t* checks that certain special forms of open work.
- *t/io/perlio.t* includes general PerlIO tests.
- *t/io/pvbm.t* checks that there is no unexpected interaction between the internal types `PVBM` and `PVGV`.
- *t/mro/package_aliases.t* checks that mro works properly in the presence of aliased packages.
- *t/op/dbm.t* tests `dbmopen` and `dbmclose`.
- *t/op/index_thr.t* tests the interaction of `index` and threads.
- *t/op/pat_thr.t* tests the interaction of esoteric patterns and threads.
- *t/op/qr_gc.t* tests that `qr` doesn't leak.
- *t/op/reg_email_thr.t* tests the interaction of regex recursion and threads.
- *t/op/regexp_qr_embed_thr.t* tests the interaction of patterns with embedded `qr//` and threads.
- *t/op/regexp_unicode_prop.t* tests Unicode properties in regular expressions.
- *t/op/regexp_unicode_prop_thr.t* tests the interaction of Unicode properties and threads.
- *t/op/reg_nc_tie.t* tests the tied methods of `Tie::Hash::NamedCapture`.
- *t/op/reg_posixcc.t* checks that POSIX character classes behave consistently.
- *t/op/re.t* checks that exportable `re` functions in *universal.c* work.
- *t/op/setpgrpstack.t* checks that `setpgrp` works.

- *t/op/substr_thr.t* tests the interaction of `substr` and threads.
- *t/op/upgrade.t* checks that upgrading and assigning scalars works.
- *t/uni/lex_utf8.t* checks that Unicode in the lexer works.
- *t/uni/tie.t* checks that Unicode and `tie` work.
- *t/comp/final_line_num.t* tests whether line numbers are correct at EOF
- *t/comp/form_scope.t* tests format scoping.
- *t/comp/line_debug.t* tests whether `@{"_<$file"}` works.
- *t/op/filetest_t.t* tests if −t file test works.
- *t/op/qr.t* tests `qr`.
- *t/op/utf8cache.t* tests malfunctions of the utf8 cache.
- *t/re/uniprops.t* test unicodes `\p{}` regex constructs.
- *t/op/filehandle.t* tests some suitably portable filetest operators to check that they work as expected, particularly in the light of some internal changes made in how filehandles are blessed.
- *t/op/time_loop.t* tests that unix times greater than `2**63`, which can now be handed to `gmtime` and `localtime`, do not cause an internal overflow or an excessively long loop.

## New or Changed Diagnostics

### New Diagnostics

- SV allocation tracing has been added to the diagnostics enabled by `-Dm`. The tracing can alternatively output via the `PERL_MEM_LOG` mechanism, if that was enabled when the *perl* binary was compiled.
- Smartmatch resolution tracing has been added as a new diagnostic. Use `-DM` to enable it.
- A new debugging flag `-DB` now dumps subroutine definitions, leaving `-Dx` for its original purpose of dumping syntax trees.
- Perl 5.12 provides a number of new diagnostic messages to help you write better code. See perldiag for details of these new messages.
  - `Bad plugin affecting keyword '%s'`
  - `gmtime(%.0f) too large`
  - `Lexing code attempted to stuff non-Latin-1 character into Latin-1 input`
  - `Lexing code internal error (%s)`
  - `localtime(%.0f) too large`
  - `Overloaded dereference did not return a reference`
  - `Overloaded qr did not return a REGEXP`
  - `Perl_pmflag() is deprecated, and will be removed from the XS API`
  - `lvalue attribute ignored after the subroutine has been defined`

    This new warning is issued when one attempts to mark a subroutine as lvalue after it has been defined.
  - Perl now warns you if `++` or `--` are unable to change the value because it's beyond the limit of representation.

    This uses a new warnings category: ''imprecision''.
  - `lc`, `uc`, `lcfirst`, and `ucfirst` warn when passed undef.
  - `Show constant in "Useless use of a constant in void context"`
  - `Prototype after '%s'`

- `panic: sv_chop %s`

  This new fatal error occurs when the C routine `Perl_sv_chop()` was passed a position that is not within the scalar's string buffer. This could be caused by buggy XS code, and at this point recovery is not possible.

- The fatal error `Malformed UTF-8 returned by \N` is now produced if the `charnames` handler returns malformed UTF-8.

- If an unresolved named character or sequence was encountered when compiling a regex pattern then the fatal error `\N{NAME} must be resolved by the lexer` is now produced. This can happen, for example, when using a single-quotish context like `$re = '\N{SPACE}'; /$re/;`. See perldiag for more examples of how the lexer can get bypassed.

- `Invalid hexadecimal number in \N{U+...}` is a new fatal error triggered when the character constant represented by `...` is not a valid hexadecimal number.

- The new meaning of `\N` as `[^\n]` is not valid in a bracketed character class, just like `.` in a character class loses its special meaning, and will cause the fatal error `\N in a character class must be a named character: \N{...}`.

- The rules on what is legal for the `...` in `\N{...}` have been tightened up so that unless the `...` begins with an alphabetic character and continues with a combination of alphanumerics, dashes, spaces, parentheses or colons then the warning `Deprecated character(s) in \N{...} starting at '%s'` is now issued.

- The warning `Using just the first characters returned by \N{}` will be issued if the `charnames` handler returns a sequence of characters which exceeds the limit of the number of characters that can be used. The message will indicate which characters were used and which were discarded.

### Changed Diagnostics

A number of existing diagnostic messages have been improved or corrected:

- A new warning category `illegalproto` allows finer-grained control of warnings around function prototypes.

  The two warnings:

  ```
  Illegal character in prototype for %s : %s
  Prototype after '%c' for %s : %s
  ```

  have been moved from the `syntax` top-level warnings category into a new first-level category, `illegalproto`. These two warnings are currently the only ones emitted during parsing of an invalid/illegal prototype, so one can now use

  ```
  no warnings 'illegalproto';
  ```

  to suppress only those, but not other syntax-related warnings. Warnings where prototypes are changed, ignored, or not met are still in the `prototype` category as before.

- `Deep recursion on subroutine "%s"`

  It is now possible to change the depth threshold for this warning from the default of 100, by recompiling the *perl* binary, setting the C pre-processor macro `PERL_SUB_DEPTH_WARN` to the desired value.

- `Illegal character in prototype` warning is now more precise when reporting illegal characters after _

- mro merging error messages are now very similar to those produced by Algorithm::C3.

- Amelioration of the error message ‘‘Unrecognized character `%s` in column `%d`’’

  Changes the error message to ‘‘Unrecognized character `%s`; marked by <-- HERE after `%s`<-- HERE near column `%d`’’. This should make it a little simpler to spot and correct the suspicious character.

- Perl now explicitly points to `$.` when it causes an uninitialized warning for ranges in scalar context.
- `split` now warns when called in void context.
- `printf`−style functions called with too few arguments will now issue the warning `"Missing argument in %s"` [perl #71000]
- Perl now properly returns a syntax error instead of segfaulting if `each`, `keys`, or `values` is used without an argument.
- `tell()` now fails properly if called without an argument and when no previous file was read.

  `tell()` now returns `−1`, and sets errno to `EBADF`, thus restoring the 5.8.x behaviour.
- `overload` no longer implicitly unsets fallback on repeated 'use overload' lines.
- **POSIX::strftime()** can now handle Unicode characters in the format string.
- The `syntax` category was removed from 5 warnings that should only be in `deprecated`.
- Three fatal `pack`/`unpack` error messages have been normalized to `panic: %s`
- `Unicode character is illegal` has been rephrased to be more accurate

  It now reads `Unicode non−character is illegal in interchange` and the perldiag documentation has been expanded a bit.
- Currently, all but the first of the several characters that the `charnames` handler may return are discarded when used in a regular expression pattern bracketed character class. If this happens then the warning `Using just the first character returned by \N{} in character class` will be issued.
- The warning `Missing right brace on \N{} or unescaped left brace after \N. Assuming the latter` will be issued if Perl encounters a `\N{` but doesn't find a matching `}`. In this case Perl doesn't know if it was mistakenly omitted, or if ''match non-newline'' followed by "match a `{`" was desired. It assumes the latter because that is actually a valid interpretation as written, unlike the other case. If you meant the former, you need to add the matching right brace. If you did mean the latter, you can silence this warning by writing instead `\N\{`.
- `gmtime` and `localtime` called with numbers smaller than they can reliably handle will now issue the warnings `gmtime(%.0f) too small` and `localtime(%.0f) too small`.

The following diagnostic messages have been removed:

- `Runaway format`
- `Can't locate package %s for the parents of %s`

  In general this warning it only got produced in conjunction with other warnings, and removing it allowed an ISA lookup optimisation to be added.
- `v−string in use/require is non−portable`

## Utility Changes

- *h2ph* now looks in `include−fixed` too, which is a recent addition to gcc's search path.
- *h2xs* no longer incorrectly treats enum values like macros. It also now handles C++ style comments (`//`) properly in enums.
- *perl5db.pl* now supports `LVALUE` subroutines. Additionally, the debugger now correctly handles proxy constant subroutines, and subroutine stubs.
- *perlbug* now uses `%Module::CoreList::bug_tracker` to print out upstream bug tracker URLs. If a user identifies a particular module as the topic of their bug report and we're able to divine the URL for its upstream bug tracker, perlbug now provide a message to the user explaining that the core copies the CPAN version directly, and provide the URL for reporting the bug directly to the upstream author.

  *perlbug* no longer reports ''Message sent'' when it hasn't actually sent the message

- *perlthanks* is a new utility for sending non-bug-reports to the authors and maintainers of Perl. Getting nothing but bug reports can become a bit demoralising. If Perl 5.12 works well for you, please try out *perlthanks*. It will make the developers smile.
- Perl's developers have fixed bugs in *a2p* having to do with the `match()` operator in list context. Additionally, *a2p* no longer generates code that uses the `$[` variable.

## Selected Bug Fixes

- U+0FFFF is now a legal character in regular expressions.
- pp_qr now always returns a new regexp SV. Resolves RT #69852.

  Instead of returning a(nother) reference to the (pre-compiled) regexp in the optree, use **reg_temp_copy()** to create a copy of it, and return a reference to that. This resolves issues about Regexp::DESTROY not being called in a timely fashion (the original bug tracked by RT #69852), as well as bugs related to blessing regexps, and of assigning to regexps, as described in correspondence added to the ticket.

  It transpires that we also need to undo the **SvPVX()** sharing when ithreads cloning a Regexp SV, because mother_re is set to NULL, instead of a cloned copy of the mother_re. This change might fix bugs with regexps and threads in certain other situations, but as yet neither tests nor bug reports have indicated any problems, so it might not actually be an edge case that it's possible to reach.
- Several compilation errors and segfaults when perl was built with `-Dmad` were fixed.
- Fixes for lexer API changes in 5.11.2 which broke NYTProf's savesrc option.
- `-t` should only return TRUE for file handles connected to a TTY

  The Microsoft C version of `isatty()` returns TRUE for all character mode devices, including the */dev/null*−style ''nul'' device and printers like ''lpt1''.
- Fixed a regression caused by commit fafafbaf which caused a panic during parameter passing [perl #70171]
- On systems which in-place edits without backup files, −i'*' now works as the documentation says it does [perl #70802]
- Saving and restoring magic flags no longer loses readonly flag.
- The malformed syntax `grep EXPR LIST` (note the missing comma) no longer causes abrupt and total failure.
- Regular expressions compiled with `qr{}` literals properly set `$'` when matching again.
- Using named subroutines with `sort` should no longer lead to bus errors [perl #71076]
- Numerous bugfixes catch small issues caused by the recently-added Lexer API.
- Smart match against `@_` sometimes gave false negatives. [perl #71078]
- `$@` may now be assigned a read-only value (without error or busting the stack).
- `sort` called recursively from within an active comparison subroutine no longer causes a bus error if run multiple times. [perl #71076]
- Tie::Hash::NamedCapture::* will not abort if passed bad input (RT #71828)
- `@_` and `$_` no longer leak under threads (RT #34342 and #41138, also #70602, #70974)
- `-I` on shebang line now adds directories in front of `@INC` as documented, and as does `-I` when specified on the command-line.
- `kill` is now fatal when called on non-numeric process identifiers. Previously, an `undef` process identifier would be interpreted as a request to kill process 0, which would terminate the current process group on POSIX systems. Since process identifiers are always integers, killing a non-numeric process is now fatal.
- 5.10.0 inadvertently disabled an optimisation, which caused a measurable performance drop in list assignment, such as is often used to assign function parameters from `@_`. The optimisation has been re-instated, and the performance regression fixed. (This fix is also present in 5.10.1)

- Fixed memory leak on `while (1) { map 1, 1 }` [RT #53038].
- Some potential coredumps in PerlIO fixed [RT #57322,54828].
- The debugger now works with lvalue subroutines.
- The debugger's `m` command was broken on modules that defined constants [RT #61222].
- `crypt` and string complement could return tainted values for untainted arguments [RT #59998].
- The `-i`*suffix* command-line switch now recreates the file using restricted permissions, before changing its mode to match the original file. This eliminates a potential race condition [RT #60904].
- On some Unix systems, the value in `$?` would not have the top bit set (`$? & 128`) even if the child core dumped.
- Under some circumstances, `$^R` could incorrectly become undefined [RT #57042].
- In the XS API, various hash functions, when passed a pre-computed hash where the key is UTF-8, might result in an incorrect lookup.
- XS code including *XSUB.h* before *perl.h* gave a compile-time error [RT #57176].
- `$object->isa('Foo')` would report false if the package `Foo` didn't exist, even if the object's `@ISA` contained `Foo`.
- Various bugs in the new-to 5.10.0 mro code, triggered by manipulating `@ISA`, have been found and fixed.
- Bitwise operations on references could crash the interpreter, e.g. `$x=\$y; $x |= "foo"` [RT #54956].
- Patterns including alternation might be sensitive to the internal UTF-8 representation, e.g.

```
my $byte = chr(192);
my $utf8 = chr(192); utf8::upgrade($utf8);
$utf8 =~ /$byte|X}/i;       # failed in 5.10.0
```

- Within UTF8-encoded Perl source files (i.e. where `use utf8` is in effect), double-quoted literal strings could be corrupted where a `\xNN`, `\0NNN` or `\N{}` is followed by a literal character with ordinal value greater than 255 [RT #59908].
- `B::Deparse` failed to correctly deparse various constructs: `readpipe STRING` [RT #62428], `CORE::require(STRING)` [RT #62488], `sub foo(_)` [RT #62484].
- Using `setpgrp` with no arguments could corrupt the perl stack.
- The block form of `eval` is now specifically trappable by `Safe` and `ops`. Previously it was erroneously treated like string `eval`.
- In 5.10.0, the two characters `[~` were sometimes parsed as the smart match operator (`~~`) [RT #63854].
- In 5.10.0, the `*` quantifier in patterns was sometimes treated as `{0,32767}` [RT #60034, #60464]. For example, this match would fail:

```
("ab" x 32768) =~ /^(ab)*$/
```

- `shmget` was limited to a 32 bit segment size on a 64 bit OS [RT #63924].
- Using `next` or `last` to exit a `given` block no longer produces a spurious warning like the following:

```
Exiting given via last at foo.pl line 123
```

- Assigning a format to a glob could corrupt the format; e.g.:

```
 *bar=*foo{FORMAT}; # foo format now bad
```

- Attempting to coerce a typeglob to a string or number could cause an assertion failure. The correct error message is now generated, `Can't coerce GLOB to` *`$type`*.

- Under `use filetest 'access'`, `-x` was using the wrong access mode. This has been fixed [RT #49003].
- `length` on a tied scalar that returned a Unicode value would not be correct the first time. This has been fixed.
- Using an array `tie` inside in array `tie` could SEGV. This has been fixed. [RT #51636]
- A race condition inside `PerlIOStdio_close()` has been identified and fixed. This used to cause various threading issues, including SEGVs.
- In `unpack`, the use of `()` groups in scalar context was internally placing a list on the interpreter's stack, which manifested in various ways, including SEGVs. This is now fixed [RT #50256].
- Magic was called twice in `substr`, `\&$x`, `tie $x, $m` and `chop`. These have all been fixed.
- A 5.10.0 optimisation to clear the temporary stack within the implicit loop of `s///ge` has been reverted, as it turned out to be the cause of obscure bugs in seemingly unrelated parts of the interpreter [commit ef0d4e17921ee3de].
- The line numbers for warnings inside `elsif` are now correct.
- The `..` operator now works correctly with ranges whose ends are at or close to the values of the smallest and largest integers.
- `binmode STDIN, ':raw'` could lead to segmentation faults on some platforms. This has been fixed [RT #54828].
- An off-by-one error meant that `index $str, ...` was effectively being executed as `index "$str\0", ...`. This has been fixed [RT #53746].
- Various leaks associated with named captures in regexes have been fixed [RT #57024].
- A weak reference to a hash would leak. This was affecting `DBI` [RT #56908].
- Using (?|) in a regex could cause a segfault [RT #59734].
- Use of a UTF-8 `tr//` within a closure could cause a segfault [RT #61520].
- Calling `Perl_sv_chop()` or otherwise upgrading an SV could result in an unaligned 64-bit access on the SPARC architecture [RT #60574].
- In the 5.10.0 release, `inc_version_list` would incorrectly list `5.10.*` after `5.8.*`; this affected the `@INC` search order [RT #67628].
- In 5.10.0, `pack "a*", $tainted_value` returned a non-tainted value [RT #52552].
- In 5.10.0, `printf` and `sprintf` could produce the fatal error `panic: utf8_mg_pos_cache_update` when printing UTF-8 strings [RT #62666].
- In the 5.10.0 release, a dynamically created `AUTOLOAD` method might be missed (method cache issue) [RT #60220,60232].
- In the 5.10.0 release, a combination of `use feature` and `//ee` could cause a memory leak [RT #63110].
- `-C` on the shebang (`#!`) line is once more permitted if it is also specified on the command line. `-C` on the shebang line used to be a silent no-op *if* it was not also on the command line, so perl 5.10.0 disallowed it, which broke some scripts. Now perl checks whether it is also on the command line and only dies if it is not [RT #67880].
- In 5.10.0, certain types of re-entrant regular expression could crash, or cause the following assertion failure [RT #60508]:

```
Assertion rx->sublen >= (s - rx->subbeg) + i failed
```

- Perl now includes previously missing files from the Unicode Character Database.
- Perl now honors `TMPDIR` when opening an anonymous temporary file.

## Platform Specific Changes

Perl is incredibly portable. In general, if a platform has a C compiler, someone has ported Perl to it (or will soon). We're happy to announce that Perl 5.12 includes support for several new platforms. At the same time, it's time to bid farewell to some (very) old friends.

### New Platforms

Haiku

Perl's developers have merged patches from Haiku's maintainers. Perl should now build on Haiku.

MirOS BSD

Perl should now build on MirOS BSD.

### Discontinued Platforms

Domain/OS

MiNT

Tenon MachTen

### Updated Platforms

AIX

- Removed *libbsd* for AIX 5L and 6.1. Only `flock()` was used from *libbsd*.
- Removed *libgdbm* for AIX 5L and 6.1 if *libgdbm* < 1.8.3−5 is installed. The *libgdbm* is delivered as an optional package with the AIX Toolbox. Unfortunately the versions below 1.8.3−5 are broken.
- Hints changes mean that AIX 4.2 should work again.

Cygwin

- Perl now supports IPv6 on Cygwin 1.7 and newer.
- On Cygwin we now strip the last number from the DLL. This has been the behaviour in the cygwin.com build for years. The hints files have been updated.

Darwin (Mac OS X)

- Skip testing the be_BY.CP1131 locale on Darwin 10 (Mac OS X 10.6), as it's still buggy.
- Correct infelicities in the regexp used to identify buggy locales on Darwin 8 and 9 (Mac OS X 10.4 and 10.5, respectively).

DragonFly BSD

- Fix thread library selection [perl #69686]

FreeBSD

- The hints files now identify the correct threading libraries on FreeBSD 7 and later.

Irix

- We now work around a bizarre preprocessor bug in the Irix 6.5 compiler: `cc -E -` unfortunately goes into K&R mode, but `cc -E file.c` doesn't.

NetBSD

- Hints now supports versions 5.*.

OpenVMS

- `-UDEBUGGING` is now the default on VMS.

  Like it has been everywhere else for ages and ages. Also make command-line selection of −UDEBUGGING and −DDEBUGGING work in configure.com; before the only way to turn it off was by saying no in answer to the interactive question.
- The default pipe buffer size on VMS has been updated to 8192 on 64−bit systems.
- Reads from the in-memory temporary files of `PerlIO::scalar` used to fail if `$/` was set to a numeric reference (to indicate record-style reads). This is now fixed.
- VMS now supports `getgrgid`.
- Many improvements and cleanups have been made to the VMS file name handling and conversion code.
- Enabling the `PERL_VMS_POSIX_EXIT` logical name now encodes a POSIX exit status in a VMS condition value for better interaction with GNV's bash shell and other utilities that depend on POSIX exit values. See "$?" in perlvms for details.
- `File::Copy` now detects Unix compatibility mode on VMS.

Stratus VOS

- Various changes from Stratus have been merged in.

Symbian

- There is now support for Symbian S60 3.2 SDK and S60 5.0 SDK.

Windows

- Perl 5.12 supports Windows 2000 and later. The supporting code for legacy versions of Windows is still included, but will be removed during the next development cycle.
- Initial support for building Perl with MinGW−w64 is now available.
- *perl.exe* now includes a manifest resource to specify the `trustInfo` settings for Windows Vista and later. Without this setting Windows would treat *perl.exe* as a legacy application and apply various heuristics like redirecting access to protected file system areas (like the ‘‘Program Files’’ folder) to the users ‘‘VirtualStore’’ instead of generating a proper ‘‘permission denied’’ error.

  The manifest resource also requests the Microsoft Common-Controls version 6.0 (themed controls introduced in Windows XP). Check out the Win32::VisualStyles module on CPAN to switch back to old style unthemed controls for legacy applications.
- The `−t` filetest operator now only returns true if the filehandle is connected to a console window. In previous versions of Perl it would return true for all character mode devices, including *NUL* and *LPT1*.
- The `−p` filetest operator now works correctly, and the Fcntl::S_IFIFO constant is defined when Perl is compiled with Microsoft Visual C. In previous Perl versions `−p` always returned a false value, and the Fcntl::S_IFIFO constant was not defined.

  This bug is specific to Microsoft Visual C and never affected Perl binaries built with MinGW.
- The socket error codes are now more widely supported: The POSIX module will define the symbolic names, like POSIX::EWOULDBLOCK, and stringification of socket error codes in $! works as well now;

  ```
  C:\>perl −MPOSIX −E "$!=POSIX::EWOULDBLOCK; say $!"
  A non−blocking socket operation could not be completed immediately.
  ```
- **flock()** will now set sensible error codes in $!. Previous Perl versions copied the value of $^E into $!, which caused much confusion.
- **select()** now supports all empty `fd_set`s more correctly.
- `'.\foo'` and `'..\foo'` were treated differently than `'./foo'` and `'../foo'` by `do` and `require` [RT #63492].
- Improved message window handling means that `alarm` and `kill` messages will no longer be dropped under race conditions.
- Various bits of Perl’s build infrastructure are no longer converted to win32 line endings at release time. If this hurts you, please report the problem with the perlbug program included with perl.

## Known Problems

This is a list of some significant unfixed bugs, which are regressions from either 5.10.x or 5.8.x.

- Some CPANPLUS tests may fail if there is a functioning file *../../cpanp−run−perl* outside your build directory. The failure shouldn’t imply there’s a problem with the actual functional software. The bug is already fixed in [RT #74188] and is scheduled for inclusion in perl−v5.12.1.
- `List::Util::first` misbehaves in the presence of a lexical `$_` (typically introduced by `my $_` or implicitly by `given`). The variable which gets set for each iteration is the package variable `$_`, not the lexical `$_` [RT #67694].

  A similar issue may occur in other modules that provide functions which take a block as their first argument, like

  ```
  foo { ... $_ ...} list
  ```

- Some regexes may run much more slowly when run in a child thread compared with the thread the pattern was compiled into [RT #55600].
- Things like `"\N{LATIN SMALL LIGATURE FF}" =~ /\N{LATIN SMALL LETTER F}+/` will appear to hang as they get into a very long running loop [RT #72998].
- Several porters have reported mysterious crashes when Perl's entire test suite is run after a build on certain Windows 2000 systems. When run by hand, the individual tests reportedly work fine.

## Errata

- This one is actually a change introduced in 5.10.0, but it was missed from that release's perldelta, so it is mentioned here instead.

  A bugfix related to the handling of the `/m` modifier and `qr` resulted in a change of behaviour between 5.8.x and 5.10.0:

  ```
  # matches in 5.8.x, doesn't match in 5.10.0
  $re = qr/^bar/; "foo\nbar" =~ /$re/m;
  ```

## Acknowledgements

Perl 5.12.0 represents approximately two years of development since Perl 5.10.0 and contains over 750,000 lines of changes across over 3,000 files from over 200 authors and committers.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.12.0:

Aaron Crane, Abe Timmerman, Abhijit Menon-Sen, Abigail, Adam Russell, Adriano Ferreira, ÆEvar Arnfjð Bjarmason, Alan Grover, Alexandr Ciornii, Alex Davies, Alex Vandiver, Andreas Koenig, Andrew Rodland, andrew@sundale.net, Andy Armstrong, Andy Dougherty, Jose AUGUSTE-ETIENNE, Benjamin Smith, Ben Morrow, bharanee rathna, Bo Borgerson, Bo Lindbergh, Brad Gilbert, Bram, Brendan O'Dea, brian d foy, Charles Bailey, Chip Salzenberg, Chris 'BinGOs' Williams, Christoph Lamprecht, Chris Williams, chromatic, Claes Jakobsson, Craig A. Berry, Dan Dascalescu, Daniel Frederick Crisman, Daniel M. Quinlan, Dan Jacobson, Dan Kogai, Dave Mitchell, Dave Rolsky, David Cantrell, David Dick, David Golden, David Mitchell, David M. Syzdek, David Nicol, David Wheeler, Dennis Kaarsemaker, Dintelmann, Peter, Dominic Dunlop, Dr.Ruud, Duke Leto, Enrico Sorcinelli, Eric Brine, Father Chrysostomos, Florian Ragwitz, Frank Wiegand, Gabor Szabo, Gene Sullivan, Geoffrey T. Dairiki, George Greer, Gerard Goossen, Gisle Aas, Goro Fuji, Graham Barr, Green, Paul, Hans Dieter Pearcey, Harmen, H. Merijn Brand, Hugo van der Sanden, Ian Goodacre, Igor Sutton, Ingo Weinhold, James Bence, James Mastros, Jan Dubois, Jari Aalto, Jarkko Hietaniemi, Jay Hannah, Jerry Hedden, Jesse Vincent, Jim Cromie, Jody Belka, John E. Malmberg, John Malmberg, John Peacock, John Peacock via RT, John P. Linderman, John Wright, Josh ben Jore, Jos I. Boumans, Karl Williamson, Kenichi Ishigaki, Ken Williams, Kevin Brintnall, Kevin Ryde, Kurt Starsinic, Leon Brocard, Lubomir Rintel, Luke Ross, Marcel Grünauer, Marcus Holland-Moritz, Mark Jason Dominus, Marko Asplund, Martin Hasch, Mashrab Kuvatov, Matt Kraai, Matt S Trout, Max Maischein, Michael Breen, Michael Cartmell, Michael G Schwern, Michael Witten, Mike Giroux, Milosz Tanski, Moritz Lenz, Nicholas Clark, Nick Cleaton, Niko Tyni, Offer Kaye, Osvaldo Villalon, Paul Fenwick, Paul Gaborit, Paul Green, Paul Johnson, Paul Marquess, Philip Hazel, Philippe Bruhat, Rafael Garcia-Suarez, Rainer Tammer, Rajesh Mandalemula, Reini Urban, Renée Bäcker, Ricardo Signes, Ricardo SIGNES, Richard Foley, Rich Rauenzahn, Rick Delaney, Risto Kankkunen, Robert May, Roberto C. Sanchez, Robin Barker, SADAHIRO Tomoyuki, Salvador Ortiz Garcia, Sam Vilain, Scott Lanning, Sébastien Aperghis-Tramoni, Sérgio Durigan Júnior, Shlomi Fish, Simon 'corecode' Schubert, Sisyphus, Slaven Rezic, Smylers, Steffen Müller, Steffen Ullrich, Stepan Kasal, Steve Hay, Steven Schubiger, Steve Peters, Tels, The Doctor, Tim Bunce, Tim Jenness, Todd Rinaldo, Tom Christiansen, Tom Hukins, Tom Wyant, Tony Cook, Torsten Schoenfeld, Tye McQueen, Vadim Konovalov, Vincent Pit, Hio YAMASHINA, Yasuhiro Matsumoto, Yitzchak Scott-Thoennes, Yuval Kogman, Yves Orton, Zefram, Zsban Ambrus

This is woefully incomplete as it's automatically generated from version control history. In particular, it doesn't include the names of the (very much appreciated) contributors who reported issues in previous versions of Perl that helped make Perl 5.12.0 better. For a more complete list of all of Perl's historical contributors, please see the `AUTHORS` file in the Perl 5.12.0 distribution.

Our ''retired'' pumpkings Nicholas Clark and Rafael Garcia-Suarez deserve special thanks for their brilliant and substantive ongoing contributions. Nicholas personally authored over 30% of the patches

since 5.10.0. Rafael comes in second in patch authorship with 11%, but is first by a long shot in committing patches authored by others, pushing 44% of the commits since 5.10.0 in this category, often after providing considerable coaching to the patch authors. These statistics in no way comprise all of their contributions, but express in shorthand that we couldn't have done it without them.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at <http://rt.perl.org/perlbug/>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analyzed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

<http://dev.perl.org/perl5/errata.html> for a list of issues found after this release, as well as a list of CPAN modules known to be incompatible with this release.

## NAME

perl5121delta − what is new for perl v5.12.1

## DESCRIPTION

This document describes differences between the 5.12.0 release and the 5.12.1 release.

If you are upgrading from an earlier release such as 5.10.1, first read perl5120delta, which describes differences between 5.10.0 and 5.12.0.

## Incompatible Changes

There are no changes intentionally incompatible with 5.12.0. If any incompatibilities with 5.12.0 exist, they are bugs. Please report them.

## Core Enhancements

Other than the bug fixes listed below, there should be no user-visible changes to the core language in this release.

## Modules and Pragmata

### Pragmata Changes

- We fixed exporting of `is_strict` and `is_lax` from version.

  These were being exported with a wrapper that treated them as method calls, which caused them to fail. They are just functions, are documented as such, and should never be subclassed, so this patch just exports them directly as functions without the wrapper.

### Updated Modules

- We upgraded CGI to version 3.49 to incorporate fixes for regressions introduced in the release we shipped with Perl 5.12.0.
- We upgraded Pod::Simple to version 3.14 to get an improvement to \C\<\< \>\> parsing.
- We made a small fix to the CPANPLUS test suite to fix an occasional spurious test failure.
- We upgraded Safe to version 2.27 to wrap coderefs returned by `reval()` and `rdo()`.

## Changes to Existing Documentation

- We added the new maintenance release policy to perlpolicy
- We've clarified the multiple-angle-bracket construct in the spec for POD in perlpodspec
- We added a missing explanation for a warning about `:=` to perldiag
- We removed a false claim in perlunitut that all text strings are Unicode strings in Perl.
- We updated the GitHub mirror link in perlrepository to mirrors/perl, not github/perl
- We fixed a minor error in perl5114delta.
- We replaced a mention of the now-obsolete Switch with *given*/*when*.
- We improved documentation about *`$sitelibexp`/sitecustomize.pl* in perlrun.
- We corrected perlmodlib which had unintentionally omitted a number of modules.
- We updated the documentation for 'require' in perlfunc relating to putting Perl code in `@INC`.
- We reinstated some erroneously-removed documentation about quotemeta in perlfunc.
- We fixed an *a2p* example in perlutil.
- We filled in a blank in perlport with the release date of Perl 5.12.
- We fixed broken links in a number of perldelta files.
- The documentation for Carp incorrectly stated that the `$Carp::Verbose` variable makes cluck generate stack backtraces.
- We fixed a number of typos in Pod::Functions
- We improved documentation of case-changing functions in perlfunc
- We corrected perlgpl to contain the correct version of the GNU General Public License.

## Testing

### Testing Improvements

- *t/op/sselect.t* is now less prone to clock jitter during timing checks on Windows.

  **sleep()** time on Win32 may be rounded down to multiple of the clock tick interval.

- *lib/blib.t* and *lib/locale.t*: Fixes for test failures on Darwin/PPC

- *perl5db.t*: Fix for test failures when `Term::ReadLine::Gnu` is installed.

## Installation and Configuration Improvements

### Configuration improvements

- We updated *INSTALL* with notes about how to deal with broken *dbm.h* on OpenSUSE (and possibly other platforms)

## Bug Fixes

- A bug in how we process filetest operations could cause a segfault. Filetests don't always expect an op on the stack, so we now use TOPs only if we're sure that we're not stat'ing the _ filehandle. This is indicated by OPf_KIDS (as checked in ck_ftst).

  See also: <https://github.com/Perl/perl5/issues/10335>

- When deparsing a nextstate op that has both a change of package (relative to the previous nextstate) and a label, the package declaration is now emitted first, because it is syntactically impermissible for a label to prefix a package declaration.

- XSUB.h now correctly redefines fgets under PERL_IMPLICIT_SYS

  See also: <http://rt.cpan.org/Public/Bug/Display.html?id=55049>

- utf8::is_utf8 now respects GMAGIC (e.g. `$1`)

- XS code using `fputc()` or `fputs()`: on Windows could cause an error due to their arguments being swapped.

  See also: <https://github.com/Perl/perl5/issues/10156>

- We fixed a small bug in **lex_stuff_pvn()** that caused spurious syntax errors in an obscure situation. It happened when stuffing was performed on the last line of a file and the line ended with a statement that lacked a terminating semicolon.

  See also: <https://github.com/Perl/perl5/issues/10273>

- We fixed a bug that could cause \N{} constructs followed by a single . to be parsed incorrectly.

  See also: <https://github.com/Perl/perl5/issues/10367>

- We fixed a bug that caused when(scalar) without an argument not to be treated as a syntax error.

  See also: <https://github.com/Perl/perl5/issues/10287>

- We fixed a regression in the handling of labels immediately before string evals that was introduced in Perl 5.12.0.

  See also: <https://github.com/Perl/perl5/issues/10301>

- We fixed a regression in case-insensitive matching of folded characters in regular expressions introduced in Perl 5.10.1.

  See also: <https://github.com/Perl/perl5/issues/10193>

## Platform Specific Notes

### HP-UX

- Perl now allows −Duse64bitint without promoting to use64bitall on HP-UX

### AIX

- Perl now builds on AIX 4.2

  The changes required work around AIX 4.2s' lack of support for IPv6, and limited support for POSIX `sigaction()`.

### FreeBSD 7

- FreeBSD 7 no longer contains */usr/bin/objformat*. At build time, Perl now skips the *objformat* check for versions 7 and higher and assumes ELF.

#### VMS

- It's now possible to build extensions on older (pre 7.3−2) VMS systems.

  DCL symbol length was limited to 1K up until about seven years or so ago, but there was no particularly deep reason to prevent those older systems from configuring and building Perl.

- We fixed the previously-broken `-Uuseperlio` build on VMS.

  We were checking a variable that doesn't exist in the non-default case of disabling perlio. Now we only look at it when it exists.

- We fixed the −Uuseperlio command-line option in configure.com.

  Formerly it only worked if you went through all the questions interactively and explicitly answered no.

## Known Problems

- `List::Util::first` misbehaves in the presence of a lexical `$_` (typically introduced by `my $_` or implicitly by `given`). The variable which gets set for each iteration is the package variable `$_`, not the lexical `$_`.

  A similar issue may occur in other modules that provide functions which take a block as their first argument, like

  ```
  foo { ... $_ ...} list
  ```

  See also: <https://github.com/Perl/perl5/issues/9798>

- `Module::Load::Conditional` and `version` have an unfortunate interaction which can cause `CPANPLUS` to crash when it encounters an unparseable version string. Upgrading to `CPANPLUS` 0.9004 or `Module::Load::Conditional` 0.38 from CPAN will resolve this issue.

## Acknowledgements

Perl 5.12.1 represents approximately four weeks of development since Perl 5.12.0 and contains approximately 4,000 lines of changes across 142 files from 28 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.12.1:

Ævar Arnfjörð Bjarmason, Chris Williams, chromatic, Craig A. Berry, David Golden, Father Chrysostomos, Florian Ragwitz, Frank Wiegand, Gene Sullivan, Goro Fuji, H.Merijn Brand, James E Keenan, Jan Dubois, Jesse Vincent, Josh ben Jore, Karl Williamson, Leon Brocard, Michael Schwern, Nga Tang Chan, Nicholas Clark, Niko Tyni, Philippe Bruhat, Rafael Garcia-Suarez, Ricardo Signes, Steffen Mueller, Todd Rinaldo, Vincent Pit and Zefram.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5122delta − what is new for perl v5.12.2

## DESCRIPTION

This document describes differences between the 5.12.1 release and the 5.12.2 release.

If you are upgrading from an earlier major version, such as 5.10.1, first read perl5120delta, which describes differences between 5.10.0 and 5.12.0, as well as perl5121delta, which describes earlier changes in the 5.12 stable release series.

## Incompatible Changes

There are no changes intentionally incompatible with 5.12.1. If any exist, they are bugs and reports are welcome.

## Core Enhancements

Other than the bug fixes listed below, there should be no user-visible changes to the core language in this release.

## Modules and Pragmata

### New Modules and Pragmata

This release does not introduce any new modules or pragmata.

### Pragmata Changes

In the previous release, `no VERSION;` statements triggered a bug which could cause feature bundles to be loaded and strict mode to be enabled unintentionally.

### Updated Modules

`Carp`

Upgraded from version 1.16 to 1.17.

Carp now detects incomplete **caller()** overrides and avoids using bogus `@DB::args`. To provide backtraces, Carp relies on particular behaviour of the caller built-in. Carp now detects if other code has overridden this with an incomplete implementation, and modifies its backtrace accordingly. Previously incomplete overrides would cause incorrect values in backtraces (best case), or obscure fatal errors (worst case)

This fixes certain cases of `Bizarre copy of ARRAY` caused by modules overriding `caller()` incorrectly.

`CPANPLUS`

A patch to *cpanp-run-perl* has been backported from CPANPLUS `0.9004`. This resolves RT #55964 <http://rt.cpan.org/Public/Bug/Display.html?id=55964> and RT #57106 <http://rt.cpan.org/Public/Bug/Display.html?id=57106>, both of which related to failures to install distributions that use `Module::Install::DSL`.

`File::Glob`

A regression which caused a failure to find `CORE::GLOBAL::glob` after loading `File::Glob` to crash has been fixed. Now, it correctly falls back to external globbing via `pp_glob`.

`File::Copy`

`File::Copy::copy(FILE, DIR)` is now documented.

`File::Spec`

Upgraded from version 3.31 to 3.31_01.

Several portability fixes were made in `File::Spec::VMS`: a colon is now recognized as a delimiter in native filespecs; caret-escaped delimiters are recognized for better handling of extended filespecs; `catpath()` returns an empty directory rather than the current directory if the input directory name is empty; `abs2rel()` properly handles Unix-style input.

## Utility Changes

- *perlbug* now always gives the reporter a chance to change the email address it guesses for them.
- *perlbug* should no longer warn about uninitialized values when using the `-d` and `-v` options.

## Changes to Existing Documentation

- The existing policy on backward-compatibility and deprecation has been added to perlpolicy, along with definitions of terms like *deprecation*.
- ‘‘srand’’ in perlfunc’s usage has been clarified.
- The entry for ‘‘die’’ in perlfunc was reorganized to emphasize its role in the exception mechanism.
- Perl’s INSTALL file has been clarified to explicitly state that Perl requires a C89 compliant ANSI C Compiler.
- IO::Socket’s `getsockopt()` and `setsockopt()` have been documented.
- ***alarm()***’s inability to interrupt blocking IO on Windows has been documented.
- Math::TrulyRandom hasn’t been updated since 1996 and has been removed as a recommended solution for random number generation.
- perlrun has been updated to clarify the behaviour of octal flags to *perl*.
- To ease user confusion, `$#` and `$*`, two special variables that were removed in earlier versions of Perl have been documented.
- The version of perlfaq shipped with the Perl core has been updated from the official FAQ version, which is now maintained in the `briandfoy/perlfaq` branch of the Perl repository at <git://perl5.git.perl.org/perl.git>.

## Installation and Configuration Improvements

### Configuration improvements

- The `d_u32align` configuration probe on ARM has been fixed.

### Compilation improvements

- An "`incompatible operand types`" error in ternary expressions when building with `clang` has been fixed.
- Perl now skips setuid `File::Copy` tests on partitions it detects to be mounted as `nosuid`.

## Selected Bug Fixes

- A possible segfault in the `T_PRTOBJ` default typemap has been fixed.
- A possible memory leak when using **caller()** to set `@DB::args` has been fixed.
- Several memory leaks when loading XS modules were fixed.
- `unpack()` now handles scalar context correctly for `%32H` and `%32u`, fixing a potential crash. `split()` would crash because the third item on the stack wasn’t the regular expression it expected. `unpack("%2H", ...)` would return both the unpacked result and the checksum on the stack, as would `unpack("%2u", ...)`. [GH #10257] <https://github.com/Perl/perl5/issues/10257>
- Perl now avoids using memory after calling `free()` in *pp_require* when there are CODEREFs in `@INC`.
- A bug that could cause "`Unknown error`“ messages when ’’`call_sv(code, G_EVAL)`" is called from an XS destructor has been fixed.
- The implementation of the `open $fh, '>' \$buffer` feature now supports get/set magic and thus tied buffers correctly.
- The `pp_getc`, `pp_tell`, and `pp_eof` opcodes now make room on the stack for their return values in cases where no argument was passed in.
- When matching unicode strings under some conditions inappropriate backtracking would result in a `Malformed UTF-8 character (fatal)` error. This should no longer occur. See [GH #10434] <https://github.com/Perl/perl5/issues/10434>

## Platform Specific Notes

### AIX

- *README.aix* has been updated with information about the XL C/C++ V11 compiler suite.

### Windows

- When building Perl with the mingw64 x64 cross-compiler `incpath`, `libpth`, `ldflags`, `lddlflags` and `ldflags_nolargefiles` values in *Config.pm* and *Config_heavy.pl* were not previously being set correctly because, with that compiler, the include and lib directories are not immediately below `$(CCHOME)`.

### VMS

- *git_version.h* is now installed on VMS. This was an oversight in v5.12.0 which caused some extensions to fail to build.
- Several memory leaks in **stat()** have been fixed.
- A memory leak in `Perl_rename()` due to a double allocation has been fixed.
- A memory leak in `vms_fid_to_name()` (used by `realpath()` and `realname()`) has been fixed.

## Acknowledgements

Perl 5.12.2 represents approximately three months of development since Perl 5.12.1 and contains approximately 2,000 lines of changes across 100 files from 36 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.12.2:

Abigail, ÆEvar Arnfjörð Bjarmason, Ben Morrow, brian d foy, Brian Phillips, Chas. Owens, Chris 'BinGOs' Williams, Chris Williams, Craig A. Berry, Curtis Jewell, Dan Dascalescu, David Golden, David Mitchell, Father Chrysostomos, Florian Ragwitz, George Greer, H.Merijn Brand, Jan Dubois, Jesse Vincent, Jim Cromie, Karl Williamson, Lars DXXXXXX XXX, Leon Brocard, Maik Hentsche, Matt S Trout, Nicholas Clark, Rafael Garcia-Suarez, Rainer Tammer, Ricardo Signes, Salvador Ortiz Garcia, Sisyphus, Slaven Rezic, Steffen Mueller, Tony Cook, Vincent Pit and Yves Orton.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5123delta − what is new for perl v5.12.3

## DESCRIPTION

This document describes differences between the 5.12.2 release and the 5.12.3 release.

If you are upgrading from an earlier release such as 5.12.1, first read perl5122delta, which describes differences between 5.12.1 and 5.12.2. The major changes made in 5.12.0 are described in perl5120delta.

## Incompatible Changes

```
There are no changes intentionally incompatible with 5.12.2. If any
exist, they are bugs and reports are welcome.
```

## Core Enhancements

### `keys`, `values` work on arrays

You can now use the `keys`, `values`, `each` builtin functions on arrays (previously you could only use them on hashes). See perlfunc for details. This is actually a change introduced in perl 5.12.0, but it was missed from that release's perldelta.

## Bug Fixes

"no VERSION" will now correctly deparse with B::Deparse, as will certain constant expressions.

Module::Build should be more reliably pass its tests under cygwin.

Lvalue subroutines are again able to return copy-on-write scalars. This had been broken since version 5.10.0.

## Platform Specific Notes

Solaris

A separate DTrace is now build for miniperl, which means that perl can be compiled with −Dusedtrace on Solaris again.

VMS

A number of regressions on VMS have been fixed. In addition to minor cleanup of questionable expressions in *vms.c*, file permissions should no longer be garbled by the PerlIO layer, and spurious record boundaries should no longer be introduced by the PerlIO layer during output.

For more details and discussion on the latter, see:

```
http://www.nntp.perl.org/group/perl.vmsperl/2010/11/msg15419.html
```

VOS

A few very small changes were made to the build process on VOS to better support the platform. Longer−than−32−character filenames are now supported on OpenVOS, and build properly without IPv6 support.

## Acknowledgements

Perl 5.12.3 represents approximately four months of development since Perl 5.12.2 and contains approximately 2500 lines of changes across 54 files from 16 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.12.3:

Craig A. Berry, David Golden, David Leadbeater, Father Chrysostomos, Florian Ragwitz, Jesse Vincent, Karl Williamson, Nick Johnston, Nicolas Kaiser, Paul Green, Rafael Garcia-Suarez, Rainer Tammer, Ricardo Signes, Steffen Mueller, Zsbán Ambrus, ÆEvar Arnfjörð Bjarmason

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed

subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5124delta − what is new for perl v5.12.4

## DESCRIPTION

This document describes differences between the 5.12.3 release and the 5.12.4 release.

If you are upgrading from an earlier release such as 5.12.2, first read perl5123delta, which describes differences between 5.12.2 and 5.12.3. The major changes made in 5.12.0 are described in perl5120delta.

### Incompatible Changes

There are no changes intentionally incompatible with 5.12.3. If any exist, they are bugs and reports are welcome.

### Selected Bug Fixes

When strict ''refs'' mode is off, `%{...}` in rvalue context returns `undef` if its argument is undefined. An optimisation introduced in Perl 5.12.0 to make `keys %{...}` faster when used as a boolean did not take this into account, causing `keys %{+undef}` (and `keys %$foo` when `$foo` is undefined) to be an error, which it should be so in strict mode only [perl #81750].

`lc`, `uc`, `lcfirst`, and `ucfirst` no longer return untainted strings when the argument is tainted. This has been broken since perl 5.8.9 [perl #87336].

Fixed a case where it was possible that a freed buffer may have been read from when parsing a here document.

### Modules and Pragmata

Module::CoreList has been upgraded from version 2.43 to 2.50.

### Testing

The *cpan/CGI/t/http.t* test script has been fixed to work when the environment has HTTPS_* environment variables, such as HTTPS_PROXY.

### Documentation

Updated the documentation for **rand()** in perlfunc to note that it is not cryptographically secure.

### Platform Specific Notes

Linux

Support Ubuntu 11.04's new multi-arch library layout.

### Acknowledgements

Perl 5.12.4 represents approximately 5 months of development since Perl 5.12.3 and contains approximately 200 lines of changes across 11 files from 8 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.12.4:

Andy Dougherty, David Golden, David Leadbeater, Father Chrysostomos, Florian Ragwitz, Jesse Vincent, Leon Brocard, Zsbán Ambrus.

### Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5125delta − what is new for perl v5.12.5

## DESCRIPTION

This document describes differences between the 5.12.4 release and the 5.12.5 release.

If you are upgrading from an earlier release such as 5.12.3, first read perl5124delta, which describes differences between 5.12.3 and 5.12.4.

## Security

### `Encode` decode_xs n−byte heap-overflow (CVE−2011−2939)

A bug in `Encode` could, on certain inputs, cause the heap to overflow. This problem has been corrected. Bug reported by Robert Zacek.

### `File::Glob::bsd_glob()` memory error with GLOB_ALTDIRFUNC (CVE−2011−2728).

Calling `File::Glob::bsd_glob` with the unsupported flag GLOB_ALTDIRFUNC would cause an access violation / segfault. A Perl program that accepts a flags value from an external source could expose itself to denial of service or arbitrary code execution attacks. There are no known exploits in the wild. The problem has been corrected by explicitly disabling all unsupported flags and setting unused function pointers to null. Bug reported by Clément Lecigne.

### Heap buffer overrun in 'x' string repeat operator (CVE−2012−5195)

Poorly written perl code that allows an attacker to specify the count to perl's 'x' string repeat operator can already cause a memory exhaustion denial-of-service attack. A flaw in versions of perl before 5.15.5 can escalate that into a heap buffer overrun; coupled with versions of glibc before 2.16, it possibly allows the execution of arbitrary code.

This problem has been fixed.

## Incompatible Changes

There are no changes intentionally incompatible with 5.12.4. If any exist, they are bugs and reports are welcome.

## Modules and Pragmata

### Updated Modules

*B::Concise*

B::Concise no longer produces mangled output with the **−tree** option [perl #80632].

*charnames*

A regression introduced in Perl 5.8.8 has been fixed, that caused `charnames::viacode(0)` to return `undef` instead of the string "NULL" [perl #72624].

*Encode has been upgraded from version 2.39 to version 2.39_01.*

See "Security".

*File::Glob has been upgraded from version 1.07 to version 1.07_01.*

See "Security".

*Unicode::UCD*

The documentation for the `upper` function now actually says "upper", not "lower".

*Module::CoreList*

Module::CoreList has been updated to version 2.50_02 to add data for this release.

## Changes to Existing Documentation

### perlebcdic

The perlebcdic document contains a helpful table to use in `tr///` to convert between EBCDIC and Latin1/ASCII. Unfortunately, the table was the inverse of the one it describes. This has been corrected.

### perlunicode

The section on User-Defined Case Mappings had some bad markup and unclear sentences, making parts of it unreadable. This has been rectified.

### perluniprops

This document has been corrected to take non-ASCII platforms into account.

## Installation and Configuration Improvements

### Platform Specific Changes

Mac OS X

There have been configuration and test fixes to make Perl build cleanly on Lion and Mountain Lion.

NetBSD

The NetBSD hints file was corrected to be compatible with NetBSD 6.*

## Selected Bug Fixes

- `chop` now correctly handles characters above "\x{7fffffff}" [perl #73246].
- `($<,$>) = (...)` stopped working properly in 5.12.0. It is supposed to make a single `setreuid()` call, rather than calling `setruid()` and `seteuid()` separately. Consequently it did not work properly. This has been fixed [perl #75212].
- Fixed a regression of **kill()** when a match variable is used for the process ID to kill [perl #75812].
- `UNIVERSAL::VERSION` no longer leaks memory. It started leaking in Perl 5.10.0.
- The C−level `my_strftime` functions no longer leaks memory. This fixes a memory leak in `POSIX::strftime` [perl #73520].
- `caller` no longer leaks memory when called from the DB package if `@DB::args` was assigned to after the first call to `caller`. Carp was triggering this bug [perl #97010].
- Passing to `index` an offset beyond the end of the string when the string is encoded internally in UTF8 no longer causes panics [perl #75898].
- Syntax errors in `(?{...})` blocks in regular expressions no longer cause panic messages [perl #2353].
- Perl 5.10.0 introduced some faulty logic that made "U*" in the middle of a pack template equivalent to "U0" if the input string was empty. This has been fixed [perl #90160].

## Errata

### split() and @_

**split()** no longer modifies `@_` when called in scalar or void context. In void context it now produces a "Useless use of split" warning. This is actually a change introduced in perl 5.12.0, but it was missed from that release's perl5120delta.

## Acknowledgements

Perl 5.12.5 represents approximately 17 months of development since Perl 5.12.4 and contains approximately 1,900 lines of changes across 64 files from 18 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.12.5:

Andy Dougherty, Chris 'BinGOs' Williams, Craig A. Berry, David Mitchell, Dominic Hargreaves, Father Chrysostomos, Florian Ragwitz, George Greer, Goro Fuji, Jesse Vincent, Karl Williamson, Leon Brocard, Nicholas Clark, Rafael Garcia-Suarez, Reini Urban, Ricardo Signes, Steve Hay, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release.

Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5140delta − what is new for perl v5.14.0

## DESCRIPTION

This document describes differences between the 5.12.0 release and the 5.14.0 release.

If you are upgrading from an earlier release such as 5.10.0, first read perl5120delta, which describes differences between 5.10.0 and 5.12.0.

Some of the bug fixes in this release have been backported to subsequent releases of 5.12.x. Those are indicated with the 5.12.x version in parentheses.

## Notice

As described in perlpolicy, the release of Perl 5.14.0 marks the official end of support for Perl 5.10. Users of Perl 5.10 or earlier should consider upgrading to a more recent release of Perl.

## Core Enhancements

### Unicode

*Unicode Version 6.0 is now supported (mostly)*

Perl comes with the Unicode 6.0 data base updated with Corrigendum #8 <http://www.unicode.org/versions/corrigendum8.html>, with one exception noted below. See <http://unicode.org/versions/Unicode6.0.0/> for details on the new release. Perl does not support any Unicode provisional properties, including the new ones for this release.

Unicode 6.0 has chosen to use the name `BELL` for the character at U+1F514, which is a symbol that looks like a bell, and is used in Japanese cell phones. This conflicts with the long-standing Perl usage of having `BELL` mean the ASCII `BEL` character, U+0007. In Perl 5.14, `\N{BELL}` continues to mean U+0007, but its use generates a deprecation warning message unless such warnings are turned off. The new name for U+0007 in Perl is `ALERT`, which corresponds nicely with the existing shorthand sequence for it, `"\a"`. `\N{BEL}` means U+0007, with no warning given. The character at U+1F514 has no name in 5.14, but can be referred to by `\N{U+1F514}`. In Perl 5.16, `\N{BELL}` will refer to U+1F514; all code that uses `\N{BELL}` should be converted to use `\N{ALERT}`, `\N{BEL}`, or `"\a"` before upgrading.

*Full functionality for `use feature 'unicode_strings'`*

This release provides full functionality for `use feature 'unicode_strings'`. Under its scope, all string operations executed and regular expressions compiled (even if executed outside its scope) have Unicode semantics. See ‘‘the ’unicode_strings’ feature’’ in feature. However, see ‘‘Inverted bracketed character classes and multi-character folds’’, below.

This feature avoids most forms of the ‘‘Unicode Bug’’ (see ‘‘The ’’Unicode Bug"" in perlunicode for details). If there is any possibility that your code will process Unicode strings, you are *strongly* encouraged to use this subpragma to avoid nasty surprises.

*`\N{NAME}` and `charnames` enhancements*

- `\N{NAME}` and `charnames::vianame` now know about the abbreviated character names listed by Unicode, such as NBSP, SHY, LRO, ZWJ, etc.; all customary abbreviations for the C0 and C1 control characters (such as ACK, BEL, CAN, etc.); and a few new variants of some C1 full names that are in common usage.
- Unicode has several *named character sequences*, in which particular sequences of code points are given names. `\N{NAME}` now recognizes these.
- `\N{NAME}`, `charnames::vianame`, and `charnames::viacode` now know about every character in Unicode. In earlier releases of Perl, they didn’t know about the Hangul syllables nor several CJK (Chinese/Japanese/Korean) characters.
- It is now possible to override Perl’s abbreviations with your own custom aliases.
- You can now create a custom alias of the ordinal of a character, known by `\N{NAME}`, `charnames::vianame()`, and `charnames::viacode()`. Previously, aliases had to be to official Unicode character names. This made it impossible to create an alias for unnamed code points, such as those reserved for private use.

- The new function **charnames::string_vianame()** is a run-time version of `\N{NAME}`, returning the string of characters whose Unicode name is its parameter. It can handle Unicode named character sequences, whereas the pre-existing **charnames::vianame()** cannot, as the latter returns a single code point.

See charnames for details on all these changes.

*New warnings categories for problematic (non−)Unicode code points.*

Three new warnings subcategories of ‘‘utf8’’ have been added. These allow you to turn off some ‘‘utf8’’ warnings, while allowing other warnings to remain on. The three categories are: `surrogate` when UTF−16 surrogates are encountered; `nonchar` when Unicode non-character code points are encountered; and `non_unicode` when code points above the legal Unicode maximum of 0x10FFFF are encountered.

*Any unsigned value can be encoded as a character*

With this release, Perl is adopting a model that any unsigned value can be treated as a code point and encoded internally (as utf8) without warnings, not just the code points that are legal in Unicode. However, unless utf8 or the corresponding sub-category (see previous item) of lexical warnings have been explicitly turned off, outputting or executing a Unicode-defined operation such as upper-casing on such a code point generates a warning. Attempting to input these using strict rules (such as with the `:encoding(UTF−8)` layer) will continue to fail. Prior to this release, handling was inconsistent and in places, incorrect.

Unicode non-characters, some of which previously were erroneously considered illegal in places by Perl, contrary to the Unicode Standard, are now always legal internally. Inputting or outputting them works the same as with the non-legal Unicode code points, because the Unicode Standard says they are (only) illegal for ‘‘open interchange’’.

*Unicode database files not installed*

The Unicode database files are no longer installed with Perl. This doesn’t affect any functionality in Perl and saves significant disk space. If you need these files, you can download them from <http://www.unicode.org/Public/zipped/6.0.0/>.

### Regular Expressions

*`(?^...)` construct signifies default modifiers*

An ASCII caret `"^"` immediately following a `"(?"` in a regular expression now means that the subexpression does not inherit surrounding modifiers such as `/i`, but reverts to the Perl defaults. Any modifiers following the caret override the defaults.

Stringification of regular expressions now uses this notation. For example, `qr/hlagh/i` would previously be stringified as `(?i−xsm:hlagh)`, but now it’s stringified as `(?^i:hlagh)`.

The main purpose of this change is to allow tests that rely on the stringification *not* to have to change whenever new modifiers are added. See ‘‘Extended Patterns’’ in perlre.

This change is likely to break code that compares stringified regular expressions with fixed strings containing `?−xism`.

*`/d`, `/l`, `/u`, and `/a` modifiers*

Four new regular expression modifiers have been added. These are mutually exclusive: one only can be turned on at a time.

- The `/l` modifier says to compile the regular expression as if it were in the scope of `use locale`, even if it is not.
- The `/u` modifier says to compile the regular expression as if it were in the scope of a `use feature 'unicode_strings'` pragma.
- The `/d` (default) modifier is used to override any `use locale` and `use feature 'unicode_strings'` pragmas in effect at the time of compiling the regular expression.
- The `/a` regular expression modifier restricts `\s`, `\d` and `\w` and the POSIX (`[[:posix:]]`) character classes to the ASCII range. Their complements and `\b` and `\B` are correspondingly affected. Otherwise, `/a` behaves like the `/u` modifier, in that case-insensitive matching uses Unicode semantics.

If the `/a` modifier is repeated, then additionally in case-insensitive matching, no ASCII character can match a non-ASCII character. For example,

```
"k"     =~ /\N{KELVIN SIGN}/ai
"\xDF" =~ /ss/ai
```

match but

```
"k"    =~ /\N{KELVIN SIGN}/aai
"\xDF" =~ /ss/aai
```

do not match.

See ‘‘Modifiers’’ in perlre for more detail.

*Non-destructive substitution*

The substitution (`s///`) and transliteration (`y///`) operators now support an `/r` option that copies the input variable, carries out the substitution on the copy, and returns the result. The original remains unmodified.

```
my $old = "cat";
my $new = $old =~ s/cat/dog/r;
# $old is "cat" and $new is "dog"
```

This is particularly useful with `map`. See perlop for more examples.

*Re-entrant regular expression engine*

It is now safe to use regular expressions within `(?{...})` and `(??{...})` code blocks inside regular expressions.

These blocks are still experimental, however, and still have problems with lexical (`my`) variables and abnormal exiting.

*`use re '/flags'`*

The `re` pragma now has the ability to turn on regular expression flags till the end of the lexical scope:

```
use re "/x";
"foo" =~ / (.+) /;  # /x implied
```

See ‘‘’/flags’ mode’’ in re for details.

*\o{...} for octals*

There is a new octal escape sequence, `"\o"`, in doublequote-like contexts. This construct allows large octal ordinals beyond the current max of 0777 to be represented. It also allows you to specify a character in octal which can safely be concatenated with other regex snippets and which won’t be confused with being a backreference to a regex capture group. See ‘‘Capture groups’’ in perlre.

*Add `\p{Titlecase}` as a synonym for `\p{Title}`*

This synonym is added for symmetry with the Unicode property names `\p{Uppercase}` and `\p{Lowercase}`.

*Regular expression debugging output improvement*

Regular expression debugging output (turned on by `use re 'debug'`) now uses hexadecimal when escaping non-ASCII characters, instead of octal.

*Return value of `delete $+{...}`*

Custom regular expression engines can now determine the return value of `delete` on an entry of `%+` or `%-`.

## Syntactical Enhancements

*Array and hash container functions accept references*

**Warning:** This feature is considered experimental, as the exact behaviour may change in a future version of Perl.

All builtin functions that operate directly on array or hash containers now also accept unblessed hard references to arrays or hashes:

```
|----------------------------+----------------------------|
| Traditional syntax         | Terse syntax               |
|----------------------------+----------------------------|
| push @$arrayref, @stuff    | push $arrayref, @stuff     |
| unshift @$arrayref, @stuff | unshift $arrayref, @stuff  |
| pop @$arrayref             | pop $arrayref              |
| shift @$arrayref           | shift $arrayref            |
| splice @$arrayref, 0, 2    | splice $arrayref, 0, 2     |
| keys %$hashref             | keys $hashref              |
| keys @$arrayref            | keys $arrayref             |
| values %$hashref           | values $hashref            |
| values @$arrayref          | values $arrayref           |
| ($k,$v) = each %$hashref   | ($k,$v) = each $hashref    |
| ($k,$v) = each @$arrayref  | ($k,$v) = each $arrayref   |
|----------------------------+----------------------------|
```

This allows these builtin functions to act on long dereferencing chains or on the return value of subroutines without needing to wrap them in `@{}` or `%{}`:

```
push @{$obj->tags}, $new_tag;  # old way
push $obj->tags,    $new_tag;  # new way

for ( keys %{$hoh->{genres}{artists}} ) {...} # old way
for ( keys $hoh->{genres}{artists}    ) {...} # new way
```

*Single term prototype*

The `+` prototype is a special alternative to `$` that acts like `\[@%]` when given a literal array or hash variable, but will otherwise force scalar context on the argument. See ''Prototypes'' in perlsub.

*`package` block syntax*

A package declaration can now contain a code block, in which case the declaration is in scope inside that block only. So `package Foo { ... }` is precisely equivalent to `{ package Foo; ... }`. It also works with a version number in the declaration, as in `package Foo 1.2 { ... }`, which is its most attractive feature. See perlfunc.

*Statement labels can appear in more places*

Statement labels can now occur before any type of statement or declaration, such as `package`.

*Stacked labels*

Multiple statement labels can now appear before a single statement.

*Uppercase X/B allowed in hexadecimal/binary literals*

Literals may now use either upper case `0X...` or `0B...` prefixes, in addition to the already supported `0x...` and `0b...` syntax [perl #76296].

C, Ruby, Python, and PHP already support this syntax, and it makes Perl more internally consistent: a round-trip with `eval sprintf "%#X", 0x10` now returns `16`, just like `eval sprintf "%#x", 0x10`.

*Overridable tie functions*

`tie`, `tied` and `untie` can now be overridden [perl #75902].

### Exception Handling

To make them more reliable and consistent, several changes have been made to how `die`, `warn`, and `$@` behave.

- When an exception is thrown inside an `eval`, the exception is no longer at risk of being clobbered by destructor code running during unwinding. Previously, the exception was written into `$@` early in the throwing process, and would be overwritten if `eval` was used internally in the destructor for an object that had to be freed while exiting from the outer `eval`. Now the exception is written into `$@` last thing before exiting the outer `eval`, so the code running immediately thereafter can rely on the value in `$@` correctly corresponding to that `eval`. (`$@` is still also set before exiting the `eval`, for the sake of destructors that rely on this.)

Likewise, a `local $@` inside an `eval` no longer clobbers any exception thrown in its scope. Previously, the restoration of `$@` upon unwinding would overwrite any exception being thrown. Now the exception gets to the `eval` anyway. So `local $@` is safe before a `die`.

Exceptions thrown from object destructors no longer modify the `$@` of the surrounding context. (If the surrounding context was exception unwinding, this used to be another way to clobber the exception being thrown.) Previously such an exception was sometimes emitted as a warning, and then either was string-appended to the surrounding `$@` or completely replaced the surrounding `$@`, depending on whether that exception and the surrounding `$@` were strings or objects. Now, an exception in this situation is always emitted as a warning, leaving the surrounding `$@` untouched. In addition to object destructors, this also affects any function call run by XS code using the `G_KEEPERR` flag.

- Warnings for `warn` can now be objects in the same way as exceptions for `die`. If an object-based warning gets the default handling of writing to standard error, it is stringified as before with the filename and line number appended. But a `$SIG{__WARN__}` handler now receives an object-based warning as an object, where previously it was passed the result of stringifying the object.

## Other Enhancements

*Assignment to `$0` sets the legacy process name with **prctl()** on Linux*

On Linux the legacy process name is now set with **prctl** (2), in addition to altering the POSIX name via `argv[0]`, as Perl has done since version 4.000. Now system utilities that read the legacy process name such as *ps*, *top*, and *killall* recognize the name you set when assigning to `$0`. The string you supply is truncated at 16 bytes; this limitation is imposed by Linux.

***srand()** now returns the seed*

This allows programs that need to have repeatable results not to have to come up with their own seed-generating mechanism. Instead, they can use **srand()** and stash the return value for future use. One example is a test program with too many combinations to test comprehensively in the time available for each run. It can test a random subset each time and, should there be a failure, log the seed used for that run so this can later be used to produce the same results.

*printf-like functions understand post−1980 size modifiers*

Perl's printf and sprintf operators, and Perl's internal printf replacement function, now understand the C90 size modifiers "hh" (`char`), "z" (`size_t`), and "t" (`ptrdiff_t`). Also, when compiled with a C99 compiler, Perl now understands the size modifier "j" (`intmax_t`) (but this is not portable).

So, for example, on any modern machine, `sprintf("%hhd", 257)` returns "1".

*New global variable `${^GLOBAL_PHASE}`*

A new global variable, `${^GLOBAL_PHASE}`, has been added to allow introspection of the current phase of the Perl interpreter. It's explained in detail in "${^GLOBAL_PHASE}" in perlvar and in "BEGIN, UNITCHECK, CHECK, INIT and END" in perlmod.

*`-d:-foo` calls `Devel::foo::unimport`*

The syntax **−d:foo** was extended in 5.6.1 to make **−d:foo=bar** equivalent to **−MDevel::foo=bar**, which expands internally to `use Devel::foo 'bar'`. Perl now allows prefixing the module name with **−**, with the same semantics as **−M**; that is:

`-d:-foo`
: Equivalent to **−M−Devel::foo**: expands to `no Devel::foo` and calls `Devel::foo->unimport()` if that method exists.

`-d:-foo=bar`
: Equivalent to **−M−Devel::foo=bar**: expands to `no Devel::foo 'bar'`, and calls `Devel::foo->unimport("bar")` if that method exists.

This is particularly useful for suppressing the default actions of a `Devel::*` module's `import` method whilst still loading it for debugging.

*Filehandle method calls load IO::File on demand*

When a method call on a filehandle would die because the method cannot be resolved and IO::File has not been loaded, Perl now loads IO::File via `require` and attempts method resolution again:

```
  open my $fh, ">", $file;
  $fh->binmode(":raw");     # loads IO::File and succeeds
```

This also works for globs like `STDOUT`, `STDERR`, and `STDIN`:

```
  STDOUT->autoflush(1);
```

Because this on-demand load happens only if method resolution fails, the legacy approach of manually loading an IO::File parent class for partial method support still works as expected:

```
  use IO::Handle;
  open my $fh, ">", $file;
  $fh->autoflush(1);        # IO::File not loaded
```

*Improved IPv6 support*

The `Socket` module provides new affordances for IPv6, including implementations of the `Socket::getaddrinfo()` and `Socket::getnameinfo()` functions, along with related constants and a handful of new functions. See Socket.

*DTrace probes now include package name*

The `DTrace` probes now include an additional argument, `arg3`, which contains the package the subroutine being entered or left was compiled in.

For example, using the following DTrace script:

```
  perl$target:::sub-entry
  {
      printf("%s::%s\n", copyinstr(arg0), copyinstr(arg3));
  }
```

and then running:

```
  $ perl -e 'sub test { }; test'
```

`DTrace` will print:

```
  main::test
```

### New C APIs

See ‘‘Internal Changes’’.

## Security

### User-defined regular expression properties

‘‘User-Defined Character Properties’’ in perlunicode documented that you can create custom properties by defining subroutines whose names begin with ‘‘In’’ or ‘‘Is’’. However, Perl did not actually enforce that naming restriction, so `\p{foo::bar}` could call **foo::bar()** if it existed. The documented convention is now enforced.

Also, Perl no longer allows tainted regular expressions to invoke a user-defined property. It simply dies instead [perl #82616].

## Incompatible Changes

Perl 5.14.0 is not binary-compatible with any previous stable release.

In addition to the sections that follow, see ‘‘C API Changes’’.

### Regular Expressions and String Escapes

*Inverted bracketed character classes and multi-character folds*

Some characters match a sequence of two or three characters in `/i` regular expression matching under Unicode rules. One example is `LATIN SMALL LETTER SHARP S` which matches the sequence `ss`.

```
 'ss' =~ /\A[\N{LATIN SMALL LETTER SHARP S}]\z/i  # Matches
```

This, however, can lead to very counter-intuitive results, especially when inverted. Because of this, Perl 5.14 does not use multi-character `/i` matching in inverted character classes.

```
 'ss' =~ /\A[^\N{LATIN SMALL LETTER SHARP S}]+\z/i  # ???
```

This should match any sequences of characters that aren’t the `SHARP S` nor what `SHARP S` matches under `/i`. `"s"` isn’t `SHARP S`, but Unicode says that `"ss"` is what `SHARP S` matches under `/i`. So

which one ‘‘wins’’? Do you fail the match because the string has `ss` or accept it because it has an `s` followed by another `s`?

Earlier releases of Perl did allow this multi-character matching, but due to bugs, it mostly did not work.

*\400−\777*

In certain circumstances, `\400−\777` in regexes have behaved differently than they behave in all other doublequote-like contexts. Since 5.10.1, Perl has issued a deprecation warning when this happens. Now, these literals behave the same in all doublequote-like contexts, namely to be equivalent to `\x{100}−\x{1FF}`, with no deprecation warning.

Use of `\400−\777` in the command-line option **−0** retain their conventional meaning. They slurp whole input files; previously, this was documented only for **−0777**.

Because of various ambiguities, you should use the new `\o{...}` construct to represent characters in octal instead.

*Most `\p{}` properties are now immune to case-insensitive matching*

For most Unicode properties, it doesn’t make sense to have them match differently under `/i` case-insensitive matching. Doing so can lead to unexpected results and potential security holes. For example

```
m/\p{ASCII_Hex_Digit}+/i
```

could previously match non-ASCII characters because of the Unicode matching rules (although there were several bugs with this). Now matching under `/i` gives the same results as non−`/i` matching except for those few properties where people have come to expect differences, namely the ones where casing is an integral part of their meaning, such as `m/\p{Uppercase}/i` and `m/\p{Lowercase}/i`, both of which match the same code points as matched by `m/\p{Cased}/i`. Details are in ‘‘Unicode Properties’’ in perlrecharclass.

User-defined property handlers that need to match differently under `/i` must be changed to read the new boolean parameter passed to them, which is non-zero if case-insensitive matching is in effect and 0 otherwise. See ‘‘User-Defined Character Properties’’ in perlunicode.

*\p{} implies Unicode semantics*

Specifying a Unicode property in the pattern indicates that the pattern is meant for matching according to Unicode rules, the way `\N{`*`NAME`*`}` does.

*Regular expressions retain their localeness when interpolated*

Regular expressions compiled under `use locale` now retain this when interpolated into a new regular expression compiled outside a `use locale`, and vice-versa.

Previously, one regular expression interpolated into another inherited the localeness of the surrounding regex, losing whatever state it originally had. This is considered a bug fix, but may trip up code that has come to rely on the incorrect behaviour.

*Stringification of regexes has changed*

Default regular expression modifiers are now notated using `(?^...)`. Code relying on the old stringification will fail. This is so that when new modifiers are added, such code won’t have to keep changing each time this happens, because the stringification will automatically incorporate the new modifiers.

Code that needs to work properly with both old− and new-style regexes can avoid the whole issue by using (for perls since 5.9.5; see re):

```
use re qw(regexp_pattern);
my ($pat, $mods) = regexp_pattern($re_ref);
```

If the actual stringification is important or older Perls need to be supported, you can use something like the following:

```
    # Accept both old and new-style stringification
    my $modifiers = (qr/foobar/ =~ /\Q(?^/) ? "^" : "-xism";
```

And then use `$modifiers` instead of `-xism`.

*Run-time code blocks in regular expressions inherit pragmata*

Code blocks in regular expressions (`(?{...})` and `(??{...})`) previously did not inherit pragmata (strict, warnings, etc.) if the regular expression was compiled at run time as happens in cases like these two:

```
use re "eval";
$foo =~ $bar; # when $bar contains (?{...})
$foo =~ /$bar(?{ $finished = 1 })/;
```

This bug has now been fixed, but code that relied on the buggy behaviour may need to be fixed to account for the correct behaviour.

### Stashes and Package Variables

*Localised tied hashes and arrays are no longed tied*

In the following:

```
tie @a, ...;
{
        local @a;
        # here, @a is a now a new, untied array
}
# here, @a refers again to the old, tied array
```

Earlier versions of Perl incorrectly tied the new local array. This has now been fixed. This fix could however potentially cause a change in behaviour of some code.

*Stashes are now always defined*

`defined %Foo::` now always returns true, even when no symbols have yet been defined in that package.

This is a side-effect of removing a special-case kludge in the tokeniser, added for 5.10.0, to hide side-effects of changes to the internal storage of hashes. The fix drastically reduces hashes' memory overhead.

Calling defined on a stash has been deprecated since 5.6.0, warned on lexicals since 5.6.0, and warned for stashes and other package variables since 5.12.0. `defined %hash` has always exposed an implementation detail: emptying a hash by deleting all entries from it does not make `defined %hash` false. Hence `defined %hash` is not valid code to determine whether an arbitrary hash is empty. Instead, use the behaviour of an empty `%hash` always returning false in scalar context.

*Clearing stashes*

Stash list assignment `%foo:: = ()` used to make the stash temporarily anonymous while it was being emptied. Consequently, any of its subroutines referenced elsewhere would become anonymous, showing up as ''(unknown)'' in `caller`. They now retain their package names such that `caller` returns the original sub name if there is still a reference to its typeglob and ''foo::__ANON__'' otherwise [perl #79208].

*Dereferencing typeglobs*

If you assign a typeglob to a scalar variable:

```
$glob = *foo;
```

the glob that is copied to `$glob` is marked with a special flag indicating that the glob is just a copy. This allows subsequent assignments to `$glob` to overwrite the glob. The original glob, however, is immutable.

Some Perl operators did not distinguish between these two types of globs. This would result in strange behaviour in edge cases: `untie $scalar` would not untie the scalar if the last thing assigned to it was a glob (because it treated it as `untie *$scalar`, which unties a handle). Assignment to a glob slot (such as `*$glob = \@some_array`) would simply assign `\@some_array` to `$glob`.

To fix this, the `*{}` operator (including its `*foo` and `*$foo` forms) has been modified to make a new immutable glob if its operand is a glob copy. This allows operators that make a distinction between globs and scalars to be modified to treat only immutable globs as globs. (`tie`, `tied` and `untie` have been left as they are for compatibility's sake, but will warn. See ''Deprecations''.)

This causes an incompatible change in code that assigns a glob to the return value of `*{}` when that operator was passed a glob copy. Take the following code, for instance:

```
$glob = *foo;
*$glob = *bar;
```

The `*$glob` on the second line returns a new immutable glob. That new glob is made an alias to `*bar`. Then it is discarded. So the second assignment has no effect.

See <https://github.com/Perl/perl5/issues/10625> for more detail.

*Magic variables outside the main package*

In previous versions of Perl, magic variables like `$!`, `%SIG`, etc. would ''leak'' into other packages. So `%foo::SIG` could be used to access signals, `${"foo::!"}` (with strict mode off) to access C's `errno`, etc.

This was a bug, or an ''unintentional'' feature, which caused various ill effects, such as signal handlers being wiped when modules were loaded, etc.

This has been fixed (or the feature has been removed, depending on how you see it).

*local($_) strips all magic from $_*

**local()** on scalar variables gives them a new value but keeps all their magic intact. This has proven problematic for the default scalar variable `$_`, where perlsub recommends that any subroutine that assigns to `$_` should first localize it. This would throw an exception if `$_` is aliased to a read-only variable, and could in general have various unintentional side-effects.

Therefore, as an exception to the general rule, local($_) will not only assign a new value to `$_`, but also remove all existing magic from it as well.

*Parsing of package and variable names*

Parsing the names of packages and package variables has changed: multiple adjacent pairs of colons, as in `foo:::bar`, are now all treated as package separators.

Regardless of this change, the exact parsing of package separators has never been guaranteed and is subject to change in future Perl versions.

### Changes to Syntax or to Perl Operators

*`given` return values*

`given` blocks now return the last evaluated expression, or an empty list if the block was exited by `break`. Thus you can now write:

```
my $type = do {
 given ($num) {
  break     when undef;
  "integer" when /^[+-]?[0-9]+$/;
  "float"   when /^[+-]?[0-9]+(?:\.[0-9]+)?$/;
  "unknown";
 }
};
```

See ''Return value'' in perlsyn for details.

*Change in parsing of certain prototypes*

Functions declared with the following prototypes now behave correctly as unary functions:

```
*
\$ \% \@ \* \&
\[...]
;$ ;*
;\$ ;\% etc.
;\[...]
```

Due to this bug fix [perl #75904], functions using the `(*)`, `(;$)` and `(;*)` prototypes are parsed with higher precedence than before. So in the following example:

```
sub foo(;$);
foo $a < $b;
```

the second line is now parsed correctly as `foo($a) < $b`, rather than `foo($a < $b)`. This happens when one of these operators is used in an unparenthesised argument:

```
< > <= >= lt gt le ge
== != <=> eq ne cmp ~~
&
| ^
&&
|| //
.. ...
?:
= += -= *= etc.
, =>
```

*Smart-matching against array slices*

Previously, the following code resulted in a successful match:

```
my @a = qw(a y0 z);
my @b = qw(a x0 z);
@a[0 .. $#b] ~~ @b;
```

This odd behaviour has now been fixed [perl #77468].

*Negation treats strings differently from before*

The unary negation operator, `-`, now treats strings that look like numbers as numbers [perl #57706].

*Negative zero*

Negative zero (-0.0), when converted to a string, now becomes "0" on all platforms. It used to become "-0" on some, but "0" on others.

If you still need to determine whether a zero is negative, use `sprintf("%g", $zero) =~ /^-/` or the Data::Float module on CPAN.

*`:=` is now a syntax error*

Previously `my $pi := 4` was exactly equivalent to `my $pi : = 4`, with the `:` being treated as the start of an attribute list, ending before the `=`. The use of `:=` to mean `: =` was deprecated in 5.12.0, and is now a syntax error. This allows future use of `:=` as a new token.

Outside the core's tests for it, we find no Perl 5 code on CPAN using this construction, so we believe that this change will have little impact on real-world codebases.

If it is absolutely necessary to have empty attribute lists (for example, because of a code generator), simply avoid the error by adding a space before the `=`.

*Change in the parsing of identifiers*

Characters outside the Unicode "XIDStart" set are no longer allowed at the beginning of an identifier. This means that certain accents and marks that normally follow an alphabetic character may no longer be the first character of an identifier.

### Threads and Processes

*Directory handles not copied to threads*

On systems other than Windows that do not have a `fchdir` function, newly-created threads no longer inherit directory handles from their parent threads. Such programs would usually have crashed anyway [perl #75154].

*`close` on shared pipes*

To avoid deadlocks, the `close` function no longer waits for the child process to exit if the underlying file descriptor is still in use by another thread. It returns true in such cases.

***fork()** emulation will not wait for signalled children*

On Windows parent processes would not terminate until all forked children had terminated first.

However, `kill("KILL", ...)` is inherently unstable on pseudo-processes, and `kill("TERM", ...)` might not get delivered if the child is blocked in a system call.

To avoid the deadlock and still provide a safe mechanism to terminate the hosting process, Perl now no longer waits for children that have been sent a SIGTERM signal. It is up to the parent process to **waitpid()** for these children if child-cleanup processing must be allowed to finish. However, it is also then the responsibility of the parent to avoid the deadlock by making sure the child process can't be blocked on I/O.

See perlfork for more information about the **fork()** emulation on Windows.

### Configuration

*Naming fixes in Policy_sh.SH may invalidate Policy.sh*

Several long-standing typos and naming confusions in *Policy_sh.SH* have been fixed, standardizing on the variable names used in *config.sh*.

This will change the behaviour of *Policy.sh* if you happen to have been accidentally relying on its incorrect behaviour.

*Perl source code is read in text mode on Windows*

Perl scripts used to be read in binary mode on Windows for the benefit of the ByteLoader module (which is no longer part of core Perl). This had the side-effect of breaking various operations on the `DATA` filehandle, including **seek()**/**tell()**, and even simply reading from `DATA` after filehandles have been flushed by a call to **system()**, backticks, **fork()** etc.

The default build options for Windows have been changed to read Perl source code on Windows in text mode now. ByteLoader will (hopefully) be updated on CPAN to automatically handle this situation [perl #28106].

## Deprecations

See also ''Deprecated C APIs''.

### Omitting a space between a regular expression and subsequent word

Omitting the space between a regular expression operator or its modifiers and the following word is deprecated. For example, `m/foo/sand $bar` is for now still parsed as `m/foo/s and $bar`, but will now issue a warning.

`\c`*X*

The backslash-c construct was designed as a way of specifying non-printable characters, but there were no restrictions (on ASCII platforms) on what the character following the `c` could be. Now, a deprecation warning is raised if that character isn't an ASCII character. Also, a deprecation warning is raised for `"\c{"` (which is the same as simply saying `";"`).

`''\b{''` **and** `''\B{''`

In regular expressions, a literal `"{"` immediately following a `"\b"` (not in a bracketed character class) or a `"\B{"` is now deprecated to allow for its future use by Perl itself.

### Perl 4−era .pl libraries

Perl bundles a handful of library files that predate Perl 5. This bundling is now deprecated for most of these files, which are now available from CPAN. The affected files now warn when run, if they were installed as part of the core.

This is a mandatory warning, not obeying **−X** or lexical warning bits. The warning is modelled on that supplied by *deprecate.pm* for deprecated-in-core *.pm* libraries. It points to the specific CPAN distribution that contains the *.pl* libraries. The CPAN versions, of course, do not generate the warning.

### List assignment to `$[`

Assignment to `$[` was deprecated and started to give warnings in Perl version 5.12.0. This version of Perl (5.14) now also emits a warning when assigning to `$[` in list context. This fixes an oversight in 5.12.0.

### Use of qw(...) as parentheses

Historically the parser fooled itself into thinking that `qw(...)` literals were always enclosed in parentheses, and as a result you could sometimes omit parentheses around them:

```
for $x qw(a b c) { ... }
```

The parser no longer lies to itself in this way. Wrap the list literal in parentheses like this:

```
for $x (qw(a b c)) { ... }
```

This is being deprecated because the parentheses in `for $i (1,2,3) { ... }` are not part of expression syntax. They are part of the statement syntax, with the `for` statement wanting literal parentheses. The synthetic parentheses that a `qw` expression acquired were only intended to be treated as part of expression syntax.

Note that this does not change the behaviour of cases like:

```
use POSIX qw(setlocale localeconv);
our @EXPORT = qw(foo bar baz);
```

where parentheses were never required around the expression.

`\N{BELL}`

This is because Unicode is using that name for a different character. See ''Unicode Version 6.0 is now supported (mostly)'' for more explanation.

`?PATTERN?`

`?PATTERN?` (without the initial `m`) has been deprecated and now produces a warning. This is to allow future use of `?` in new operators. The match-once functionality is still available as `m?PATTERN?`.

### Tie functions on scalars holding typeglobs

Calling a tie function (`tie`, `tied`, `untie`) with a scalar argument acts on a filehandle if the scalar happens to hold a typeglob.

This is a long-standing bug that will be removed in Perl 5.16, as there is currently no way to tie the scalar itself when it holds a typeglob, and no way to untie a scalar that has had a typeglob assigned to it.

Now there is a deprecation warning whenever a tie function is used on a handle without an explicit `*`.

### User-defined case-mapping

This feature is being deprecated due to its many issues, as documented in ''User-Defined Case Mappings (for serious hackers only)'' in perlunicode. This feature will be removed in Perl 5.16. Instead use the CPAN module Unicode::Casing, which provides improved functionality.

### Deprecated modules

The following module will be removed from the core distribution in a future release, and should be installed from CPAN instead. Distributions on CPAN that require this should add it to their prerequisites. The core version of these module now issues a deprecation warning.

If you ship a packaged version of Perl, either alone or as part of a larger system, then you should carefully consider the repercussions of core module deprecations. You may want to consider shipping your default build of Perl with a package for the deprecated module that installs into `vendor` or `site` Perl library directories. This will inhibit the deprecation warnings.

Alternatively, you may want to consider patching *lib/deprecate.pm* to provide deprecation warnings specific to your packaging system or distribution of Perl, consistent with how your packaging system or distribution manages a staged transition from a release where the installation of a single package provides the given functionality, to a later release where the system administrator needs to know to install multiple packages to get that same functionality.

You can silence these deprecation warnings by installing the module in question from CPAN. To install the latest version of it by role rather than by name, just install `Task::Deprecations::5_14`.

Devel::DProf

We strongly recommend that you install and use Devel::NYTProf instead of Devel::DProf, as Devel::NYTProf offers significantly improved profiling and reporting.

## Performance Enhancements

### "Safe signals" optimisation

Signal dispatch has been moved from the runloop into control ops. This should give a few percent speed increase, and eliminates nearly all the speed penalty caused by the introduction of ''safe signals'' in 5.8.0. Signals should still be dispatched within the same statement as they were previously. If this does *not* happen, or if you find it possible to create uninterruptible loops, this is a bug, and reports are encouraged of how to recreate such issues.

### Optimisation of shift() and pop() calls without arguments

Two fewer OPs are used for **shift()** and **pop()** calls with no argument (with implicit `@_`). This change makes **shift()** 5% faster than `shift @_` on non-threaded perls, and 25% faster on threaded ones.

### Optimisation of regexp engine string comparison work

The `foldEQ_utf8` API function for case-insensitive comparison of strings (which is used heavily by the regexp engine) was substantially refactored and optimised — and its documentation much improved as a free bonus.

### Regular expression compilation speed-up

Compiling regular expressions has been made faster when upgrading the regex to utf8 is necessary but this isn't known when the compilation begins.

### String appending is 100 times faster

When doing a lot of string appending, perls built to use the system's `malloc` could end up allocating a lot more memory than needed in a inefficient way.

`sv_grow`, the function used to allocate more memory if necessary when appending to a string, has been taught to round up the memory it requests to a certain geometric progression, making it much faster on certain platforms and configurations. On Win32, it's now about 100 times faster.

### Eliminate `PL_*` accessor functions under ithreads

When `MULTIPLICITY` was first developed, and interpreter state moved into an interpreter struct, thread− and interpreter-local `PL_*` variables were defined as macros that called accessor functions (returning the address of the value) outside the Perl core. The intent was to allow members within the interpreter struct to change size without breaking binary compatibility, so that bug fixes could be merged to a maintenance branch that necessitated such a size change. This mechanism was redundant and penalised well-behaved code. It has been removed.

### Freeing weak references

When there are many weak references to an object, freeing that object can under some circumstances take O(*N***N*) time to free, where *N* is the number of references. The circumstances in which this can happen have been reduced [perl #75254]

### Lexical array and hash assignments

An earlier optimisation to speed up `my @array = ...` and `my %hash = ...` assignments caused a bug and was disabled in Perl 5.12.0.

Now we have found another way to speed up these assignments [perl #82110].

### `@_` uses less memory

Previously, `@_` was allocated for every subroutine at compile time with enough space for four entries. Now this allocation is done on demand when the subroutine is called [perl #72416].

### Size optimisations to SV and HV structures

`xhv_fill` has been eliminated from `struct xpvhv`, saving 1 IV per hash and on some systems will cause `struct xpvhv` to become cache-aligned. To avoid this memory saving causing a slowdown elsewhere, boolean use of `HvFILL` now calls `HvTOTALKEYS` instead (which is equivalent), so while the fill data when actually required are now calculated on demand, cases when this needs to be done should be rare.

The order of structure elements in SV bodies has changed. Effectively, the NV slot has swapped location with STASH and MAGIC. As all access to SV members is via macros, this should be completely transparent. This change allows the space saving for PVHVs documented above, and may reduce the memory allocation needed for PVIVs on some architectures.

`XPV`, `XPVIV`, and `XPVNV` now allocate only the parts of the `SV` body they actually use, saving some space.

Scalars containing regular expressions now allocate only the part of the `SV` body they actually use, saving some space.

### Memory consumption improvements to Exporter

The `@EXPORT_FAIL` AV is no longer created unless needed, hence neither is the typeglob backing it. This saves about 200 bytes for every package that uses Exporter but doesn't use this functionality.

### Memory savings for weak references

For weak references, the common case of just a single weak reference per referent has been optimised to reduce the storage required. In this case it saves the equivalent of one small Perl array per referent.

### %+ and %− use less memory

The bulk of the `Tie::Hash::NamedCapture` module used to be in the Perl core. It has now been moved to an XS module to reduce overhead for programs that do not use `%+` or `%−`.

### Multiple small improvements to threads

The internal structures of threading now make fewer API calls and fewer allocations, resulting in noticeably smaller object code. Additionally, many thread context checks have been deferred so they're done only as needed (although this is only possible for non-debugging builds).

### Adjacent pairs of nextstate opcodes are now optimized away

Previously, in code such as

```
use constant DEBUG => 0;

sub GAK {
    warn if DEBUG;
    print "stuff\n";
}
```

the ops for `warn if DEBUG` would be folded to a `null` op (`ex-const`), but the `nextstate` op would remain, resulting in a runtime op dispatch of `nextstate`, `nextstate`, etc.

The execution of a sequence of `nextstate` ops is indistinguishable from just the last `nextstate` op so the peephole optimizer now eliminates the first of a pair of `nextstate` ops except when the first carries a label, since labels must not be eliminated by the optimizer, and label usage isn't conclusively known at compile time.

## Modules and Pragmata

### New Modules and Pragmata

- CPAN::Meta::YAML 0.003 has been added as a dual-life module. It supports a subset of YAML sufficient for reading and writing *META.yml* and *MYMETA.yml* files included with CPAN distributions or generated by the module installation toolchain. It should not be used for any other general YAML parsing or generation task.
- CPAN::Meta version 2.110440 has been added as a dual-life module. It provides a standard library to read, interpret and write CPAN distribution metadata files (like *META.json* and *META.yml*) that describe a distribution, its contents, and the requirements for building it and installing it. The latest CPAN distribution metadata specification is included as CPAN::Meta::Spec and notes on changes in the specification over time are given in CPAN::Meta::History.
- HTTP::Tiny 0.012 has been added as a dual-life module. It is a very small, simple HTTP/1.1 client designed for simple GET requests and file mirroring. It has been added so that *CPAN.pm* and CPANPLUS can ''bootstrap'' HTTP access to CPAN using pure Perl without relying on external binaries like **curl** (1) or **wget** (1).
- JSON::PP 2.27105 has been added as a dual-life module to allow CPAN clients to read *META.json* files in CPAN distributions.
- Module::Metadata 1.000004 has been added as a dual-life module. It gathers package and POD information from Perl module files. It is a standalone module based on Module::Build::ModuleInfo for use by other module installation toolchain components. Module::Build::ModuleInfo has been deprecated in favor of this module instead.
- Perl::OSType 1.002 has been added as a dual-life module. It maps Perl operating system names (like ''dragonfly'' or ''MSWin32'') to more generic types with standardized names (like ''Unix'' or ''Windows''). It has been refactored out of Module::Build and ExtUtils::CBuilder and consolidates such mappings into a single location for easier maintenance.
- The following modules were added by the Unicode::Collate upgrade. See below for details.

  Unicode::Collate::CJK::Big5

  Unicode::Collate::CJK::GB2312

Unicode::Collate::CJK::JISX0208

Unicode::Collate::CJK::Korean

Unicode::Collate::CJK::Pinyin

Unicode::Collate::CJK::Stroke

- Version::Requirements version 0.101020 has been added as a dual-life module. It provides a standard library to model and manipulates module prerequisites and version constraints defined in CPAN::Meta::Spec.

### Updated Modules and Pragma

- attributes has been upgraded from version 0.12 to 0.14.
- Archive::Extract has been upgraded from version 0.38 to 0.48.

  Updates since 0.38 include: a safe print method that guards Archive::Extract from changes to `$\`; a fix to the tests when run in core Perl; support for TZ files; a modification for the lzma logic to favour IO::Uncompress::Unlzma; and a fix for an issue with NetBSD-current and its new **unzip** (1) executable.
- Archive::Tar has been upgraded from version 1.54 to 1.76.

  Important changes since 1.54 include the following:

  - Compatibility with busybox implementations of **tar** (1).
  - A fix so that **write()** and **create_archive()** close only filehandles they themselves opened.
  - A bug was fixed regarding the exit code of extract_archive.
  - The **ptar** (1) utility has a new option to allow safe creation of tarballs without world-writable files on Windows, allowing those archives to be uploaded to CPAN.
  - A new **ptargrep** (1) utility for using regular expressions against the contents of files in a tar archive.
  - pax extended headers are now skipped.
- Attribute::Handlers has been upgraded from version 0.87 to 0.89.
- autodie has been upgraded from version 2.06_01 to 2.1001.
- AutoLoader has been upgraded from version 5.70 to 5.71.
- The B module has been upgraded from version 1.23 to 1.29.

  It no longer crashes when taking apart a `y///` containing characters outside the octet range or compiled in a `use utf8` scope.

  The size of the shared object has been reduced by about 40%, with no reduction in functionality.
- B::Concise has been upgraded from version 0.78 to 0.83.

  B::Concise marks **rv2sv()**, **rv2av()**, and **rv2hv()** ops with the new `OPpDEREF` flag as ''DREFed''.

  It no longer produces mangled output with the **−tree** option [perl #80632].
- B::Debug has been upgraded from version 1.12 to 1.16.
- B::Deparse has been upgraded from version 0.96 to 1.03.

  The deparsing of a `nextstate` op has changed when it has both a change of package relative to the previous nextstate, or a change of `%^H` or other state and a label. The label was previously emitted first, but is now emitted last (5.12.1).

  The `no 5.13.2` or similar form is now correctly handled by B::Deparse (5.12.3).

  B::Deparse now properly handles the code that applies a conditional pattern match against implicit `$_` as it was fixed in [perl #20444].

  Deparsing of `our` followed by a variable with funny characters (as permitted under the `use utf8` pragma) has also been fixed [perl #33752].

- B::Lint has been upgraded from version 1.11_01 to 1.13.
- base has been upgraded from version 2.15 to 2.16.
- Benchmark has been upgraded from version 1.11 to 1.12.
- bignum has been upgraded from version 0.23 to 0.27.
- Carp has been upgraded from version 1.15 to 1.20.

  Carp now detects incomplete **caller()** overrides and avoids using bogus `@DB::args`. To provide backtraces, Carp relies on particular behaviour of the **caller()** builtin. Carp now detects if other code has overridden this with an incomplete implementation, and modifies its backtrace accordingly. Previously incomplete overrides would cause incorrect values in backtraces (best case), or obscure fatal errors (worst case).

  This fixes certain cases of ''Bizarre copy of ARRAY'' caused by modules overriding **caller()** incorrectly (5.12.2).

  It now also avoids using regular expressions that cause Perl to load its Unicode tables, so as to avoid the ''BEGIN not safe after errors'' error that ensue if there has been a syntax error [perl #82854].
- CGI has been upgraded from version 3.48 to 3.52.

  This provides the following security fixes: the MIME boundary in **multipart_init()** is now random and the handling of newlines embedded in header values has been improved.
- Compress::Raw::Bzip2 has been upgraded from version 2.024 to 2.033.

  It has been updated to use **bzip2** (1) 1.0.6.
- Compress::Raw::Zlib has been upgraded from version 2.024 to 2.033.
- constant has been upgraded from version 1.20 to 1.21.

  Unicode constants work once more. They have been broken since Perl 5.10.0 [CPAN RT #67525].
- CPAN has been upgraded from version 1.94_56 to 1.9600.

  Major highlights:

  - much less configuration dialog hassle
  - support for *META/MYMETA.json*
  - support for local::lib
  - support for HTTP::Tiny to reduce the dependency on FTP sites
  - automatic mirror selection
  - iron out all known bugs in configure_requires
  - support for distributions compressed with **bzip2** (1)
  - allow *Foo/Bar.pm* on the command line to mean `Foo::Bar`
- CPANPLUS has been upgraded from version 0.90 to 0.9103.

  A change to *cpanp-run-perl* resolves RT #55964 <http://rt.cpan.org/Public/Bug/Display.html?id=55964> and RT #57106 <http://rt.cpan.org/Public/Bug/Display.html?id=57106>, both of which related to failures to install distributions that use `Module::Install::DSL` (5.12.2).

  A dependency on Config was not recognised as a core module dependency. This has been fixed.

  CPANPLUS now includes support for *META.json* and *MYMETA.json*.
- CPANPLUS::Dist::Build has been upgraded from version 0.46 to 0.54.
- Data::Dumper has been upgraded from version 2.125 to 2.130_02.

  The indentation used to be off when `$Data::Dumper::Terse` was set. This has been fixed [perl #73604].

  This upgrade also fixes a crash when using custom sort functions that might cause the stack to

change [perl #74170].

Dumpxs no longer crashes with globs returned by `*$io_ref` [perl #72332].

- DB_File has been upgraded from version 1.820 to 1.821.
- DBM_Filter has been upgraded from version 0.03 to 0.04.
- Devel::DProf has been upgraded from version 20080331.00 to 20110228.00.

  Merely loading Devel::DProf now no longer triggers profiling to start. Both `use Devel::DProf` and `perl -d:DProf ...` behave as before and start the profiler.

  **NOTE**: Devel::DProf is deprecated and will be removed from a future version of Perl. We strongly recommend that you install and use Devel::NYTProf instead, as it offers significantly improved profiling and reporting.
- Devel::Peek has been upgraded from version 1.04 to 1.07.
- Devel::SelfStubber has been upgraded from version 1.03 to 1.05.
- diagnostics has been upgraded from version 1.19 to 1.22.

  It now renders pod links slightly better, and has been taught to find descriptions for messages that share their descriptions with other messages.
- Digest::MD5 has been upgraded from version 2.39 to 2.51.

  It is now safe to use this module in combination with threads.
- Digest::SHA has been upgraded from version 5.47 to 5.61.

  `shasum` now more closely mimics **sha1sum**(1)/**md5sum**(1).

  `addfile` accepts all POSIX filenames.

  New SHA-512/224 and SHA-512/256 transforms (ref. NIST Draft FIPS 180-4 [February 2011])
- DirHandle has been upgraded from version 1.03 to 1.04.
- Dumpvalue has been upgraded from version 1.13 to 1.16.
- DynaLoader has been upgraded from version 1.10 to 1.13.

  It fixes a buffer overflow when passed a very long file name.

  It no longer inherits from AutoLoader; hence it no longer produces weird error messages for unsuccessful method calls on classes that inherit from DynaLoader [perl #84358].
- Encode has been upgraded from version 2.39 to 2.42.

  Now, all 66 Unicode non-characters are treated the same way U+FFFF has always been treated: in cases when it was disallowed, all 66 are disallowed, and in cases where it warned, all 66 warn.
- Env has been upgraded from version 1.01 to 1.02.
- Errno has been upgraded from version 1.11 to 1.13.

  The implementation of Errno has been refactored to use about 55% less memory.

  On some platforms with unusual header files, like Win32 **gcc**(1) using `mingw64` headers, some constants that weren't actually error numbers have been exposed by Errno. This has been fixed [perl #77416].
- Exporter has been upgraded from version 5.64_01 to 5.64_03.

  Exporter no longer overrides `$SIG{__WARN__}` [perl #74472]
- ExtUtils::CBuilder has been upgraded from version 0.27 to 0.280203.
- ExtUtils::Command has been upgraded from version 1.16 to 1.17.
- ExtUtils::Constant has been upgraded from 0.22 to 0.23.

  The AUTOLOAD helper code generated by `ExtUtils::Constant::ProxySubs` can now **croak()** for missing constants, or generate a complete `AUTOLOAD` subroutine in XS, allowing simplification of many modules that use it (Fcntl, File::Glob, GDBM_File, I18N::Langinfo,

POSIX, Socket).

ExtUtils::Constant::ProxySubs can now optionally push the names of all constants onto the package's `@EXPORT_OK`.

- ExtUtils::Install has been upgraded from version 1.55 to 1.56.
- ExtUtils::MakeMaker has been upgraded from version 6.56 to 6.57_05.
- ExtUtils::Manifest has been upgraded from version 1.57 to 1.58.
- ExtUtils::ParseXS has been upgraded from version 2.21 to 2.2210.
- Fcntl has been upgraded from version 1.06 to 1.11.
- File::Basename has been upgraded from version 2.78 to 2.82.
- File::CheckTree has been upgraded from version 4.4 to 4.41.
- File::Copy has been upgraded from version 2.17 to 2.21.
- File::DosGlob has been upgraded from version 1.01 to 1.04.

  It allows patterns containing literal parentheses: they no longer need to be escaped. On Windows, it no longer adds an extra *./* to file names returned when the pattern is a relative glob with a drive specification, like *C:*.pl* [perl #71712].
- File::Fetch has been upgraded from version 0.24 to 0.32.

  HTTP::Lite is now supported for the ''http'' scheme.

  The **fetch**(1) utility is supported on FreeBSD, NetBSD, and Dragonfly BSD for the `http` and `ftp` schemes.
- File::Find has been upgraded from version 1.15 to 1.19.

  It improves handling of backslashes on Windows, so that paths like *C:\dir\/file* are no longer generated [perl #71710].
- File::Glob has been upgraded from version 1.07 to 1.12.
- File::Spec has been upgraded from version 3.31 to 3.33.

  Several portability fixes were made in File::Spec::VMS: a colon is now recognized as a delimiter in native filespecs; caret-escaped delimiters are recognized for better handling of extended filespecs; **catpath()** returns an empty directory rather than the current directory if the input directory name is empty; and **abs2rel()** properly handles Unix-style input (5.12.2).
- File::stat has been upgraded from 1.02 to 1.05.

  The `-x` and `-X` file test operators now work correctly when run by the superuser.
- Filter::Simple has been upgraded from version 0.84 to 0.86.
- GDBM_File has been upgraded from 1.10 to 1.14.

  This fixes a memory leak when DBM filters are used.
- Hash::Util has been upgraded from 0.07 to 0.11.

  Hash::Util no longer emits spurious ''uninitialized'' warnings when recursively locking hashes that have undefined values [perl #74280].
- Hash::Util::FieldHash has been upgraded from version 1.04 to 1.09.
- I18N::Collate has been upgraded from version 1.01 to 1.02.
- I18N::Langinfo has been upgraded from version 0.03 to 0.08.

  **langinfo()** now defaults to using `$_` if there is no argument given, just as the documentation has always claimed.
- I18N::LangTags has been upgraded from version 0.35 to 0.35_01.
- if has been upgraded from version 0.05 to 0.0601.

- IO has been upgraded from version 1.25_02 to 1.25_04.

  This version of IO includes a new IO::Select, which now allows IO::Handle objects (and objects in derived classes) to be removed from an IO::Select set even if the underlying file descriptor is closed or invalid.

- IPC::Cmd has been upgraded from version 0.54 to 0.70.

  Resolves an issue with splitting Win32 command lines. An argument consisting of the single character ‘‘0’’ used to be omitted (CPAN RT #62961).

- IPC::Open3 has been upgraded from 1.05 to 1.09.

  **open3()** now produces an error if the `exec` call fails, allowing this condition to be distinguished from a child process that exited with a non-zero status [perl #72016].

  The internal **xclose()** routine now knows how to handle file descriptors as documented, so duplicating `STDIN` in a child process using its file descriptor now works [perl #76474].

- IPC::SysV has been upgraded from version 2.01 to 2.03.

- lib has been upgraded from version 0.62 to 0.63.

- Locale::Maketext has been upgraded from version 1.14 to 1.19.

  Locale::Maketext now supports external caches.

  This upgrade also fixes an infinite loop in `Locale::Maketext::Guts::_compile()` when working with tainted values (CPAN RT #40727).

  `->maketext` calls now back up and restore `$@` so error messages are not suppressed (CPAN RT #34182).

- Log::Message has been upgraded from version 0.02 to 0.04.

- Log::Message::Simple has been upgraded from version 0.06 to 0.08.

- Math::BigInt has been upgraded from version 1.89_01 to 1.994.

  This fixes, among other things, incorrect results when computing binomial coefficients [perl #77640].

  It also prevents `sqrt($int)` from crashing under `use bigrat`. [perl #73534].

- Math::BigInt::FastCalc has been upgraded from version 0.19 to 0.28.

- Math::BigRat has been upgraded from version 0.24 to 0.26_02.

- Memoize has been upgraded from version 1.01_03 to 1.02.

- MIME::Base64 has been upgraded from 3.08 to 3.13.

  Includes new functions to calculate the length of encoded and decoded base64 strings.

  Now provides **encode_base64url()** and **decode_base64url()** functions to process the base64 scheme for ‘‘URL applications’’.

- Module::Build has been upgraded from version 0.3603 to 0.3800.

  A notable change is the deprecation of several modules. Module::Build::Version has been deprecated and Module::Build now relies on the version pragma directly. Module::Build::ModuleInfo has been deprecated in favor of a standalone copy called Module::Metadata. Module::Build::YAML has been deprecated in favor of CPAN::Meta::YAML.

  Module::Build now also generates *META.json* and *MYMETA.json* files in accordance with version 2 of the CPAN distribution metadata specification, CPAN::Meta::Spec. The older format *META.yml* and *MYMETA.yml* files are still generated.

- Module::CoreList has been upgraded from version 2.29 to 2.47.

  Besides listing the updated core modules of this release, it also stops listing the `Filespec` module. That module never existed in core. The scripts generating Module::CoreList confused it with VMS::Filespec, which actually is a core module as of Perl 5.8.7.

- Module::Load has been upgraded from version 0.16 to 0.18.
- Module::Load::Conditional has been upgraded from version 0.34 to 0.44.
- The mro pragma has been upgraded from version 1.02 to 1.07.
- NDBM_File has been upgraded from version 1.08 to 1.12.

  This fixes a memory leak when DBM filters are used.
- Net::Ping has been upgraded from version 2.36 to 2.38.
- NEXT has been upgraded from version 0.64 to 0.65.
- Object::Accessor has been upgraded from version 0.36 to 0.38.
- ODBM_File has been upgraded from version 1.07 to 1.10.

  This fixes a memory leak when DBM filters are used.
- Opcode has been upgraded from version 1.15 to 1.18.
- The overload pragma has been upgraded from 1.10 to 1.13.

  `overload::Method` can now handle subroutines that are themselves blessed into overloaded classes [perl #71998].

  The documentation has greatly improved. See ''Documentation'' below.
- Params::Check has been upgraded from version 0.26 to 0.28.
- The parent pragma has been upgraded from version 0.223 to 0.225.
- Parse::CPAN::Meta has been upgraded from version 1.40 to 1.4401.

  The latest Parse::CPAN::Meta can now read YAML and JSON files using CPAN::Meta::YAML and JSON::PP, which are now part of the Perl core.
- PerlIO::encoding has been upgraded from version 0.12 to 0.14.
- PerlIO::scalar has been upgraded from 0.07 to 0.11.

  A **read()** after a **seek()** beyond the end of the string no longer thinks it has data to read [perl #78716].
- PerlIO::via has been upgraded from version 0.09 to 0.11.
- Pod::Html has been upgraded from version 1.09 to 1.11.
- Pod::LaTeX has been upgraded from version 0.58 to 0.59.
- Pod::Perldoc has been upgraded from version 3.15_02 to 3.15_03.
- Pod::Simple has been upgraded from version 3.13 to 3.16.
- POSIX has been upgraded from 1.19 to 1.24.

  It now includes constants for POSIX signal constants.
- The re pragma has been upgraded from version 0.11 to 0.18.

  The `use re '/flags'` subpragma is new.

  The **regmust()** function used to crash when called on a regular expression belonging to a pluggable engine. Now it croaks instead.

  **regmust()** no longer leaks memory.
- Safe has been upgraded from version 2.25 to 2.29.

  Coderefs returned by **reval()** and **rdo()** are now wrapped via **wrap_code_refs()** (5.12.1).

  This fixes a possible infinite loop when looking for coderefs.

  It adds several `version::vxs::*` routines to the default share.
- SDBM_File has been upgraded from version 1.06 to 1.09.

- SelfLoader has been upgraded from 1.17 to 1.18.

  It now works in taint mode [perl #72062].

- The sigtrap pragma has been upgraded from version 1.04 to 1.05.

  It no longer tries to modify read-only arguments when generating a backtrace [perl #72340].

- Socket has been upgraded from version 1.87 to 1.94.

  See ''Improved IPv6 support'' above.

- Storable has been upgraded from version 2.22 to 2.27.

  Includes performance improvement for overloaded classes.

  This adds support for serialising code references that contain UTF-8 strings correctly. The Storable minor version number changed as a result, meaning that Storable users who set `$Storable::accept_future_minor` to a `FALSE` value will see errors (see ''FORWARD COMPATIBILITY'' in Storable for more details).

  Freezing no longer gets confused if the Perl stack gets reallocated during freezing [perl #80074].

- Sys::Hostname has been upgraded from version 1.11 to 1.16.
- Term::ANSIColor has been upgraded from version 2.02 to 3.00.
- Term::UI has been upgraded from version 0.20 to 0.26.
- Test::Harness has been upgraded from version 3.17 to 3.23.
- Test::Simple has been upgraded from version 0.94 to 0.98.

  Among many other things, subtests without a `plan` or `no_plan` now have an implicit **done_testing()** added to them.

- Thread::Semaphore has been upgraded from version 2.09 to 2.12.

  It provides two new methods that give more control over the decrementing of semaphores: `down_nb` and `down_force`.

- Thread::Queue has been upgraded from version 2.11 to 2.12.
- The threads pragma has been upgraded from version 1.75 to 1.83.
- The threads::shared pragma has been upgraded from version 1.32 to 1.37.
- Tie::Hash has been upgraded from version 1.03 to 1.04.

  Calling `Tie::Hash->TIEHASH()` used to loop forever. Now it `croak`s.

- Tie::Hash::NamedCapture has been upgraded from version 0.06 to 0.08.
- Tie::RefHash has been upgraded from version 1.38 to 1.39.
- Time::HiRes has been upgraded from version 1.9719 to 1.9721_01.
- Time::Local has been upgraded from version 1.1901_01 to 1.2000.
- Time::Piece has been upgraded from version 1.15_01 to 1.20_01.
- Unicode::Collate has been upgraded from version 0.52_01 to 0.73.

  Unicode::Collate has been updated to use Unicode 6.0.0.

  Unicode::Collate::Locale now supports a plethora of new locales: *ar, be, bg, de__phonebook, hu, hy, kk, mk, nso, om, tn, vi, hr, ig, ja, ko, ru, sq, se, sr, to, uk, zh, zh__big5han, zh__gb2312han, zh__pinyin*, and *zh__stroke*.

  The following modules have been added:

  Unicode::Collate::CJK::Big5 for `zh__big5han` which makes tailoring of CJK Unified Ideographs in the order of CLDR's big5han ordering.

  Unicode::Collate::CJK::GB2312 for `zh__gb2312han` which makes tailoring of CJK Unified Ideographs in the order of CLDR's gb2312han ordering.

  Unicode::Collate::CJK::JISX0208 which makes tailoring of 6355 kanji (CJK Unified Ideographs)

in the JIS X 0208 order.

Unicode::Collate::CJK::Korean which makes tailoring of CJK Unified Ideographs in the order of CLDR's Korean ordering.

Unicode::Collate::CJK::Pinyin for `zh__pinyin` which makes tailoring of CJK Unified Ideographs in the order of CLDR's pinyin ordering.

Unicode::Collate::CJK::Stroke for `zh__stroke` which makes tailoring of CJK Unified Ideographs in the order of CLDR's stroke ordering.

This also sees the switch from using the pure-Perl version of this module to the XS version.

- Unicode::Normalize has been upgraded from version 1.03 to 1.10.
- Unicode::UCD has been upgraded from version 0.27 to 0.32.

  A new function, **Unicode::UCD::num()**, has been added. This function returns the numeric value of the string passed it or `undef` if the string in its entirety has no ''safe'' numeric value. (For more detail, and for the definition of ''safe'', see ''**num()**'' in Unicode::UCD.)

  This upgrade also includes several bug fixes:

  **charinfo()**
  - It is now updated to Unicode Version 6.0.0 with *Corrigendum #8*, excepting that, just as with Perl 5.14, the code point at U+1F514 has no name.
  - Hangul syllable code points have the correct names, and their decompositions are always output without requiring Lingua::KO::Hangul::Util to be installed.
  - CJK (Chinese-Japanese-Korean) code points U+2A700 to U+2B734 and U+2B740 to U+2B81D are now properly handled.
  - Numeric values are now output for those CJK code points that have them.
  - Names output for code points with multiple aliases are now the corrected ones.

  **charscript()**
  This now correctly returns ''Unknown'' instead of `undef` for the script of a code point that hasn't been assigned another one.

  **charblock()**
  This now correctly returns ''No_Block'' instead of `undef` for the block of a code point that hasn't been assigned to another one.
- The version pragma has been upgraded from 0.82 to 0.88.

  Because of a bug, now fixed, the **is_strict()** and **is_lax()** functions did not work when exported (5.12.1).
- The warnings pragma has been upgraded from version 1.09 to 1.12.

  Calling `use warnings` without arguments is now significantly more efficient.
- The warnings::register pragma has been upgraded from version 1.01 to 1.02.

  It is now possible to register warning categories other than the names of packages using warnings::register. See **perllexwarn** (1) for more information.
- XSLoader has been upgraded from version 0.10 to 0.13.
- VMS::DCLsym has been upgraded from version 1.03 to 1.05.

  Two bugs have been fixed [perl #84086]:

  The symbol table name was lost when tying a hash, due to a thinko in `TIEHASH`. The result was that all tied hashes interacted with the local symbol table.

  Unless a symbol table name had been explicitly specified in the call to the constructor, querying the special key `:LOCAL` failed to identify objects connected to the local symbol table.
- The Win32 module has been upgraded from version 0.39 to 0.44.

  This release has several new functions: **Win32::GetSystemMetrics()**, **Win32::GetProductInfo()**,

**Win32::GetOSDisplayName()**.

The names returned by **Win32::GetOSName()** and **Win32::GetOSDisplayName()** have been corrected.

- XS::Typemap has been upgraded from version 0.03 to 0.05.

### Removed Modules and Pragmata

As promised in Perl 5.12.0's release notes, the following modules have been removed from the core distribution, and if needed should be installed from CPAN instead.

- Class::ISA has been removed from the Perl core. Prior version was 0.36.
- Pod::Plainer has been removed from the Perl core. Prior version was 1.02.
- Switch has been removed from the Perl core. Prior version was 2.16.

The removal of Shell has been deferred until after 5.14, as the implementation of Shell shipped with 5.12.0 did not correctly issue the warning that it was to be removed from core.

## Documentation

### New Documentation

*perlgpl*

perlgpl has been updated to contain GPL version 1, as is included in the *README* distributed with Perl (5.12.1).

*Perl 5.12.x delta files*

The perldelta files for Perl 5.12.1 to 5.12.3 have been added from the maintenance branch: perl5121delta, perl5122delta, perl5123delta.

*perlpodstyle*

New style guide for POD documentation, split mostly from the NOTES section of the **pod2man**(1) manpage.

*perlsource, perlinterp, perlhacktut, and perlhacktips*

See ''perlhack and perlrepository revamp'', below.

### Changes to Existing Documentation

*perlmodlib is now complete*

The perlmodlib manpage that came with Perl 5.12.0 was missing several modules due to a bug in the script that generates the list. This has been fixed [perl #74332] (5.12.1).

*Replace incorrect tr/// table in perlebcdic*

perlebcdic contains a helpful table to use in `tr///` to convert between EBCDIC and Latin1/ASCII. The table was the inverse of the one it describes, though the code that used the table worked correctly for the specific example given.

The table has been corrected and the sample code changed to correspond.

The table has also been changed to hex from octal, and the recipes in the pod have been altered to print out leading zeros to make all values the same length.

*Tricks for user-defined casing*

perlunicode now contains an explanation of how to override, mangle and otherwise tweak the way Perl handles upper−, lower− and other-case conversions on Unicode data, and how to provide scoped changes to alter one's own code's behaviour without stomping on anybody else's.

*INSTALL explicitly states that Perl requires a C89 compiler*

This was already true, but it's now Officially Stated For The Record (5.12.2).

*Explanation of `\xHH` and `\oOOO` escapes*

perlop has been updated with more detailed explanation of these two character escapes.

***−0NNN* switch**

In perlrun, the behaviour of the **−0NNN** switch for **−0400** or higher has been clarified (5.12.2).

*Maintenance policy*

perlpolicy now contains the policy on what patches are acceptable for maintenance branches (5.12.1).

*Deprecation policy*

perlpolicy now contains the policy on compatibility and deprecation along with definitions of terms like ‘‘deprecation’’ (5.12.2).

*New descriptions in perldiag*

The following existing diagnostics are now documented:

- Ambiguous use of %c resolved as operator %c
- Ambiguous use of %c{%s} resolved to %c%s
- Ambiguous use of %c{%s[...]} resolved to %c%s[...]
- Ambiguous use of %c{%s{...}} resolved to %c%s{...}
- Ambiguous use of −%s resolved as −&%s()
- Invalid strict version format (%s)
- Invalid version format (%s)
- Invalid version object

*perlbook*

perlbook has been expanded to cover many more popular books.

*SvTRUE macro*

The documentation for the SvTRUE macro in perlapi was simply wrong in stating that get-magic is not processed. It has been corrected.

*op manipulation functions*

Several API functions that process optrees have been newly documented.

*perlvar revamp*

perlvar reorders the variables and groups them by topic. Each variable introduced after Perl 5.000 notes the first version in which it is available. perlvar also has a new section for deprecated variables to note when they were removed.

*Array and hash slices in scalar context*

These are now documented in perldata.

*use locale and formats*

perlform and perllocale have been corrected to state that `use locale` affects formats.

*overload*

overload’s documentation has practically undergone a rewrite. It is now much more straightforward and clear.

*perlhack and perlrepository revamp*

The perlhack document is now much shorter, and focuses on the Perl 5 development process and submitting patches to Perl. The technical content has been moved to several new documents, perlsource, perlinterp, perlhacktut, and perlhacktips. This technical content has been only lightly edited.

The perlrepository document has been renamed to perlgit. This new document is just a how-to on using git with the Perl source code. Any other content that used to be in perlrepository has been moved to perlhack.

*Time::Piece examples*

Examples in perlfaq4 have been updated to show the use of Time::Piece.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

## New Diagnostics

*New Errors*

Closure prototype called
: This error occurs when a subroutine reference passed to an attribute handler is called, if the subroutine is a closure [perl #68560].

Insecure user-defined property %s
: Perl detected tainted data when trying to compile a regular expression that contains a call to a user-defined character property function, meaning `\p{IsFoo}` or `\p{InFoo}`. See ''User-Defined Character Properties'' in perlunicode and perlsec.

panic: gp_free failed to free glob pointer − something is repeatedly re-creating entries
: This new error is triggered if a destructor called on an object in a typeglob that is being freed creates a new typeglob entry containing an object with a destructor that creates a new entry containing an object etc.

Parsing code internal error (%s)
: This new fatal error is produced when parsing code supplied by an extension violates the parser's API in a detectable way.

refcnt: fd %d%s
: This new error only occurs if an internal consistency check fails when a pipe is about to be closed.

Regexp modifier ''/%c'' may not appear twice
: The regular expression pattern has one of the mutually exclusive modifiers repeated.

Regexp modifiers ''/%c'' and ''/%c'' are mutually exclusive
: The regular expression pattern has more than one of the mutually exclusive modifiers.

Using !~ with %s doesn't make sense
: This error occurs when `!~` is used with `s///r` or `y///r`.

*New Warnings*

''\b{'' is deprecated; use ''\b\{'' instead
''\B{'' is deprecated; use ''\B\{'' instead
: Use of an unescaped ''{'' immediately following a `\b` or `\B` is now deprecated in order to reserve its use for Perl itself in a future release.

Operation ''%s'' returns its argument for ...
: Performing an operation requiring Unicode semantics (such as case-folding) on a Unicode surrogate or a non-Unicode character now triggers this warning.

Use of qw(...) as parentheses is deprecated
: See ''Use of qw(...) as parentheses'', above, for details.

## Changes to Existing Diagnostics

- The ''Variable `$foo` is not imported'' warning that precedes a `strict 'vars'` error has now been assigned the ''misc'' category, so that `no warnings` will suppress it [perl #73712].
- **warn()** and **die()** now produce ''Wide character'' warnings when fed a character outside the byte range if `STDERR` is a byte-sized handle.
- The ''Layer does not match this perl'' error message has been replaced with these more helpful messages [perl #73754]:
  - PerlIO layer function table size (%d) does not match size expected by this perl (%d)
  - PerlIO layer instance size (%d) does not match size expected by this perl (%d)
- The ''Found = in conditional'' warning that is emitted when a constant is assigned to a variable in a condition is now withheld if the constant is actually a subroutine or one generated by `use constant`, since the value of the constant may not be known at the time the program is written [perl #77762].
- Previously, if none of the **gethostbyaddr()**, **gethostbyname()** and **gethostent()** functions were implemented on a given platform, they would all die with the message ''Unsupported socket function 'gethostent' called'', with analogous messages for getnet*() and getserv*(). This has been corrected.

- The warning message about unrecognized regular expression escapes passed through has been changed to include any literal ‘‘{’’ following the two-character escape. For example, ‘‘\q{’’ is now emitted instead of ‘‘\q’’.

## Utility Changes

***perlbug** (1)*

- perlbug now looks in the EMAIL environment variable for a return address if the REPLY-TO and REPLYTO variables are empty.
- perlbug did not previously generate a ‘‘From:’’ header, potentially resulting in dropped mail; it now includes that header.
- The user’s address is now used as the Return-Path.

  Many systems these days don’t have a valid Internet domain name, and perlbug@perl.org does not accept email with a return-path that does not resolve. So the user’s address is now passed to sendmail so it’s less likely to get stuck in a mail queue somewhere [perl #82996].
- perlbug now always gives the reporter a chance to change the email address it guesses for them (5.12.2).
- perlbug should no longer warn about uninitialized values when using the **−d** and **−v** options (5.12.2).

*perl5db.pl*

- The remote terminal works after forking and spawns new sessions, one per forked process.

*ptargrep*

- ptargrep is a new utility to apply pattern matching to the contents of files in a tar archive. It comes with `Archive::Tar`.

## Configuration and Compilation

See also ‘‘Naming fixes in Policy_sh.SH may invalidate Policy.sh’’, above.

- CCINCDIR and CCLIBDIR for the mingw64 cross-compiler are now correctly under *$(CCHOME)\mingw\include* and *\lib* rather than immediately below *$(CCHOME)*.

  This means the ‘‘incpath’’, ‘‘libpth’’, ‘‘ldflags’’, ‘‘lddlflags’’ and ‘‘ldflags_nolargefiles’’ values in *Config.pm* and *Config_heavy.pl* are now set correctly.
- `make test.valgrind` has been adjusted to account for *cpan/dist/ext* separation.
- On compilers that support it, **−Wwrite−strings** is now added to cflags by default.
- The Encode module can now (once again) be included in a static Perl build. The special-case handling for this situation got broken in Perl 5.11.0, and has now been repaired.
- The previous default size of a PerlIO buffer (4096 bytes) has been increased to the larger of 8192 bytes and your local BUFSIZ. Benchmarks show that doubling this decade-old default increases read and write performance by around 25% to 50% when using the default layers of perlio on top of unix. To choose a non-default size, such as to get back the old value or to obtain an even larger value, configure with:

  ```
  ./Configure -Accflags=-DPERLIOBUF_DEFAULT_BUFSIZ=N
  ```

  where N is the desired size in bytes; it should probably be a multiple of your page size.
- An ‘‘incompatible operand types’’ error in ternary expressions when building with `clang` has been fixed (5.12.2).
- Perl now skips setuid File::Copy tests on partitions it detects mounted as `nosuid` (5.12.2).

## Platform Support

### New Platforms

AIX

Perl now builds on AIX 4.2 (5.12.1).

### Discontinued Platforms

Apollo DomainOS
: The last vestiges of support for this platform have been excised from the Perl distribution. It was officially discontinued in version 5.12.0. It had not worked for years before that.

MacOS Classic
: The last vestiges of support for this platform have been excised from the Perl distribution. It was officially discontinued in an earlier version.

## Platform-Specific Notes

*AIX*

- *README.aix* has been updated with information about the XL C/C++ V11 compiler suite (5.12.2).

*ARM*

- The `d_u32align` configuration probe on ARM has been fixed (5.12.2).

*Cygwin*

- MakeMaker has been updated to build manpages on cygwin.
- Improved rebase behaviour

  If a DLL is updated on cygwin the old imagebase address is reused. This solves most rebase errors, especially when updating on core DLL's. See <http://www.tishler.net/jason/software/rebase/rebase−2.4.2.README> for more information.
- Support for the standard cygwin dll prefix (needed for FFIs)
- Updated build hints file

*FreeBSD 7*

- FreeBSD 7 no longer contains */usr/bin/objformat*. At build time, Perl now skips the *objformat* check for versions 7 and higher and assumes ELF (5.12.1).

*HP-UX*

- Perl now allows **−Duse64bitint** without promoting to `use64bitall` on HP-UX (5.12.1).

*IRIX*

- Conversion of strings to floating-point numbers is now more accurate on IRIX systems [perl #32380].

*Mac OS X*

- Early versions of Mac OS X (Darwin) had buggy implementations of the **setregid()**, **setreuid()**, setrgid(,) and **setruid()** functions, so Perl would pretend they did not exist.

  These functions are now recognised on Mac OS 10.5 (Leopard; Darwin 9) and higher, as they have been fixed [perl #72990].

*MirBSD*

- Previously if you built Perl with a shared *libperl.so* on MirBSD (the default config), it would work up to the installation; however, once installed, it would be unable to find *libperl*. Path handling is now treated as in the other BSD dialects.

*NetBSD*

- The NetBSD hints file has been changed to make the system malloc the default.

*OpenBSD*

- OpenBSD > 3.7 has a new malloc implementation which is *mmap*−based, and as such can release memory back to the OS; however, Perl's use of this malloc causes a substantial slowdown, so we now default to using Perl's malloc instead [perl #75742].

*OpenVOS*

- Perl now builds again with OpenVOS (formerly known as Stratus VOS) [perl #78132] (5.12.3).

*Solaris*

- DTrace is now supported on Solaris. There used to be build failures, but these have been fixed [perl #73630] (5.12.3).

*VMS*

- Extension building on older (pre 7.3−2) VMS systems was broken because configure.com hit the DCL symbol length limit of 1K. We now work within this limit when assembling the list of extensions in the core build (5.12.1).
- We fixed configuring and building Perl with **−Uuseperlio** (5.12.1).
- `PerlIOUnix_open` now honours the default permissions on VMS.

  When `perlio` became the default and `unix` became the default bottom layer, the most common path for creating files from Perl became `PerlIOUnix_open`, which has always explicitly used `0666` as the permission mask. This prevents inheriting permissions from RMS defaults and ACLs, so to avoid that problem, we now pass `0777` to **open()**. In the VMS CRTL, `0777` has a special meaning over and above intersecting with the current umask; specifically, it allows Unix syscalls to preserve native default permissions (5.12.3).
- The shortening of symbols longer than 31 characters in the core C sources and in extensions is now by default done by the C compiler rather than by xsubpp (which could only do so for generated symbols in XS code). You can reenable xsubpp's symbol shortening by configuring with −Uuseshortenedsymbols, but you'll have some work to do to get the core sources to compile.
- Record-oriented files (record format variable or variable with fixed control) opened for write by the `perlio` layer will now be line-buffered to prevent the introduction of spurious line breaks whenever the perlio buffer fills up.
- *git_version.h* is now installed on VMS. This was an oversight in v5.12.0 which caused some extensions to fail to build (5.12.2).
- Several memory leaks in **stat()** have been fixed (5.12.2).
- A memory leak in **Perl_rename()** due to a double allocation has been fixed (5.12.2).
- A memory leak in **vms_fid_to_name()** (used by **realpath()** and **realname()**> has been fixed (5.12.2).

*Windows*

See also ''**fork()** emulation will not wait for signalled children'' and ''Perl source code is read in text mode on Windows'', above.

- Fixed build process for SDK2003SP1 compilers.
- Compilation with Visual Studio 2010 is now supported.
- When using old 32−bit compilers, the define `_USE_32BIT_TIME_T` is now set in `$Config{ccflags}`. This improves portability when compiling XS extensions using new compilers, but for a Perl compiled with old 32−bit compilers.
- `$Config{gccversion}` is now set correctly when Perl is built using the mingw64 compiler from <http://mingw64.org> [perl #73754].
- When building Perl with the mingw64 x64 cross-compiler `incpath`, `libpth`, `ldflags`, `lddlflags` and `ldflags_nolargefiles` values in *Config.pm* and *Config_heavy.pl* were not previously being set correctly because, with that compiler, the include and lib directories are not immediately below `$(CCHOME)` (5.12.2).
- The build process proceeds more smoothly with mingw and dmake when *C:\MSYS\bin* is in the PATH, due to a `Cwd` fix.
- Support for building with Visual C++ 2010 is now underway, but is not yet complete. See *README.win32* or perlwin32 for more details.
- The option to use an externally-supplied **crypt()**, or to build with no **crypt()** at all, has been removed. Perl supplies its own **crypt()** implementation for Windows, and the political situation that required this part of the distribution to sometimes be omitted is long gone.

## Internal Changes

### New APIs

*CLONE_PARAMS structure added to ease correct thread creation*

Modules that create threads should now create `CLONE_PARAMS` structures by calling the new function **Perl_clone_params_new()**, and free them with **Perl_clone_params_del()**. This will ensure compatibility with any future changes to the internals of the `CLONE_PARAMS` structure layout, and that it is correctly allocated and initialised.

*New parsing functions*

Several functions have been added for parsing Perl statements and expressions. These functions are meant to be used by XS code invoked during Perl parsing, in a recursive-descent manner, to allow modules to augment the standard Perl syntax.

- **parse_stmtseq()** parses a sequence of statements, up to closing brace or EOF.
- **parse_fullstmt()** parses a complete Perl statement, including optional label.
- **parse_barestmt()** parses a statement without a label.
- **parse_block()** parses a code block.
- **parse_label()** parses a statement label, separate from statements.
- `parse_fullexpr()`, `parse_listexpr()`, `parse_termexpr()`, and `parse_arithexpr()` parse expressions at various precedence levels.

*Hints hash API*

A new C API for introspecting the hinthash `%^H` at runtime has been added. See `cop_hints_2hv`, `cop_hints_fetchpvn`, `cop_hints_fetchpvs`, `cop_hints_fetchsv`, and `hv_copy_hints_hv` in perlapi for details.

A new, experimental API has been added for accessing the internal structure that Perl uses for `%^H`. See the functions beginning with `cophh_` in perlapi.

*C interface to **caller()***

The `caller_cx` function has been added as an XSUB-writer's equivalent of **caller()**. See perlapi for details.

*Custom per-subroutine check hooks*

XS code in an extension module can now annotate a subroutine (whether implemented in XS or in Perl) so that nominated XS code will be called at compile time (specifically as part of op checking) to change the op tree of that subroutine. The compile-time check function (supplied by the extension module) can implement argument processing that can't be expressed as a prototype, generate customised compile-time warnings, perform constant folding for a pure function, inline a subroutine consisting of sufficiently simple ops, replace the whole call with a custom op, and so on. This was previously all possible by hooking the `entersub` op checker, but the new mechanism makes it easy to tie the hook to a specific subroutine. See ''cv_set_call_checker'' in perlapi.

To help in writing custom check hooks, several subtasks within standard `entersub` op checking have been separated out and exposed in the API.

*Improved support for custom OPs*

Custom ops can now be registered with the new `custom_op_register` C function and the `XOP` structure. This will make it easier to add new properties of custom ops in the future. Two new properties have been added already, `xop_class` and `xop_peep`.

`xop_class` is one of the OA_*OP constants. It allows B and other introspection mechanisms to work with custom ops that aren't BASEOPs. `xop_peep` is a pointer to a function that will be called for ops of this type from `Perl_rpeep`.

See ''Custom Operators'' in perlguts and ''Custom Operators'' in perlapi for more detail.

The old `PL_custom_op_names`/`PL_custom_op_descs` interface is still supported but discouraged.

*Scope hooks*

It is now possible for XS code to hook into Perl's lexical scope mechanism at compile time, using the new `Perl_blockhook_register` function. See ''Compile-time scope hooks'' in perlguts.

*The recursive part of the peephole optimizer is now hookable*

In addition to `PL_peepp`, for hooking into the toplevel peephole optimizer, a `PL_rpeepp` is now available to hook into the optimizer recursing into side-chains of the optree.

*New non-magical variants of existing functions*

The following functions/macros have been added to the API. The `*_nomg` macros are equivalent to their non-`_nomg` variants, except that they ignore get-magic. Those ending in `_flags` allow one to specify whether get-magic is processed.

```
sv_2bool_flags
SvTRUE_nomg
sv_2nv_flags
SvNV_nomg
sv_cmp_flags
sv_cmp_locale_flags
sv_eq_flags
sv_collxfrm_flags
```

In some of these cases, the non-`_flags` functions have been replaced with wrappers around the new functions.

*pv/pvs/sv versions of existing functions*

Many functions ending with pvn now have equivalent `pv/pvs/sv` versions.

*List op-building functions*

List op-building functions have been added to the API. See op_append_elem, op_append_list, and op_prepend_elem in perlapi.

*`LINKLIST`*

The LINKLIST macro, part of op building that constructs the execution-order op chain, has been added to the API.

*Localisation functions*

The `save_freeop`, `save_op`, `save_pushi32ptr` and `save_pushptrptr` functions have been added to the API.

*Stash names*

A stash can now have a list of effective names in addition to its usual name. The first effective name can be accessed via the `HvENAME` macro, which is now the recommended name to use in MRO linearisations (`HvNAME` being a fallback if there is no `HvENAME`).

These names are added and deleted via `hv_ename_add` and `hv_ename_delete`. These two functions are *not* part of the API.

*New functions for finding and removing magic*

The `mg_findext()` and `sv_unmagicext()` functions have been added to the API. They allow extension authors to find and remove magic attached to scalars based on both the magic type and the magic virtual table, similar to how **sv_magicext()** attaches magic of a certain type and with a given virtual table to a scalar. This eliminates the need for extensions to walk the list of `MAGIC` pointers of an `SV` to find the magic that belongs to them.

*`find_rundefsv`*

This function returns the SV representing `$_`, whether it's lexical or dynamic.

*`Perl_croak_no_modify`*

**Perl_croak_no_modify()** is short-hand for `Perl_croak("%s", PL_no_modify)`.

*`PERL_STATIC_INLINE` define*

The `PERL_STATIC_INLINE` define has been added to provide the best-guess incantation to use for static inline functions, if the C compiler supports C99-style static inline. If it doesn't, it'll give a plain

`static`.

`HAS_STATIC_INLINE` can be used to check if the compiler actually supports inline functions.

*New `pv_escape` option for hexadecimal escapes*

A new option, `PERL_PV_ESCAPE_NONASCII`, has been added to `pv_escape` to dump all characters above ASCII in hexadecimal. Before, one could get all characters as hexadecimal or the Latin1 non-ASCII as octal.

*`lex_start`*

`lex_start` has been added to the API, but is considered experimental.

***op_scope()* and *op_lvalue()***

The **op_scope()** and **op_lvalue()** functions have been added to the API, but are considered experimental.

### C API Changes

*`PERL_POLLUTE` has been removed*

The option to define `PERL_POLLUTE` to expose older 5.005 symbols for backwards compatibility has been removed. Its use was always discouraged, and MakeMaker contains a more specific escape hatch:

```
perl Makefile.PL POLLUTE=1
```

This can be used for modules that have not been upgraded to 5.6 naming conventions (and really should be completely obsolete by now).

*Check API compatibility when loading XS modules*

When Perl's API changes in incompatible ways (which usually happens between major releases), XS modules compiled for previous versions of Perl will no longer work. They need to be recompiled against the new Perl.

The `XS_APIVERSION_BOOTCHECK` macro has been added to ensure that modules are recompiled and to prevent users from accidentally loading modules compiled for old perls into newer perls. That macro, which is called when loading every newly compiled extension, compares the API version of the running perl with the version a module has been compiled for and raises an exception if they don't match.

*Perl_fetch_cop_label*

The first argument of the C API function `Perl_fetch_cop_label` has changed from `struct refcounted_he *` to `COP *`, to insulate the user from implementation details.

This API function was marked as ''may change'', and likely isn't in use outside the core. (Neither an unpacked CPAN nor Google's codesearch finds any other references to it.)

***GvCV()* and *GvGP()* are no longer lvalues**

The new **GvCV_set()** and **GvGP_set()** macros are now provided to replace assignment to those two macros.

This allows a future commit to eliminate some backref magic between GV and CVs, which will require complete control over assignment to the `gp_cv` slot.

***CvGV()* is no longer an lvalue**

Under some circumstances, the **CvGV()** field of a CV is now reference-counted. To ensure consistent behaviour, direct assignment to it, for example `CvGV(cv) = gv` is now a compile-time error. A new macro, `CvGV_set(cv,gv)` has been introduced to run this operation safely. Note that modification of this field is not part of the public API, regardless of this new macro (and despite its being listed in this section).

***CvSTASH()* is no longer an lvalue**

The **CvSTASH()** macro can now only be used as an rvalue. **CvSTASH_set()** has been added to replace assignment to **CvSTASH()**. This is to ensure that backreferences are handled properly. These macros are not part of the API.

*Calling conventions for `newFOROP` and `newWHILEOP`*

The way the parser handles labels has been cleaned up and refactored. As a result, the **newFOROP()** constructor function no longer takes a parameter stating what label is to go in the state op.

The **newWHILEOP()** and **newFOROP()** functions no longer accept a line number as a parameter.

*Flags passed to* `uvuni_to_utf8_flags` *and* `utf8n_to_uvuni`

Some of the flags parameters to **uvuni_to_utf8_flags()** and **utf8n_to_uvuni()** have changed. This is a result of Perl's now allowing internal storage and manipulation of code points that are problematic in some situations. Hence, the default actions for these functions has been complemented to allow these code points. The new flags are documented in perlapi. Code that requires the problematic code points to be rejected needs to change to use the new flags. Some flag names are retained for backward source compatibility, though they do nothing, as they are now the default. However the flags `UNICODE_ALLOW_FDD0`, `UNICODE_ALLOW_FFFF`, `UNICODE_ILLEGAL`, and `UNICODE_IS_ILLEGAL` have been removed, as they stem from a fundamentally broken model of how the Unicode non-character code points should be handled, which is now described in ''Non-character code points'' in perlunicode. See also the Unicode section under ''Selected Bug Fixes''.

### Deprecated C APIs

`Perl_ptr_table_clear`

`Perl_ptr_table_clear` is no longer part of Perl's public API. Calling it now generates a deprecation warning, and it will be removed in a future release.

`sv_compile_2op`

The **sv_compile_2op()** API function is now deprecated. Searches suggest that nothing on CPAN is using it, so this should have zero impact.

It attempted to provide an API to compile code down to an optree, but failed to bind correctly to lexicals in the enclosing scope. It's not possible to fix this problem within the constraints of its parameters and return value.

`find_rundefsvoffset`

The `find_rundefsvoffset` function has been deprecated. It appeared that its design was insufficient for reliably getting the lexical `$_` at run-time.

Use the new `find_rundefsv` function or the `UNDERBAR` macro instead. They directly return the right SV representing `$_`, whether it's lexical or dynamic.

`CALL_FPTR` and `CPERLscope`

Those are left from an old implementation of `MULTIPLICITY` using C++ objects, which was removed in Perl 5.8. Nowadays these macros do exactly nothing, so they shouldn't be used anymore.

For compatibility, they are still defined for external `XS` code. Only extensions defining `PERL_CORE` must be updated now.

### Other Internal Changes

*Stack unwinding*

The protocol for unwinding the C stack at the last stage of a `die` has changed how it identifies the target stack frame. This now uses a separate variable `PL_restartjmpenv`, where previously it relied on the `blk_eval.cur_top_env` pointer in the `eval` context frame that has nominally just been discarded. This change means that code running during various stages of Perl-level unwinding no longer needs to take care to avoid destroying the ghost frame.

*Scope stack entries*

The format of entries on the scope stack has been changed, resulting in a reduction of memory usage of about 10%. In particular, the memory used by the scope stack to record each active lexical variable has been halved.

*Memory allocation for pointer tables*

Memory allocation for pointer tables has been changed. Previously `Perl_ptr_table_store` allocated memory from the same arena system as `SV` bodies and `HE`s, with freed memory remaining bound to those arenas until interpreter exit. Now it allocates memory from arenas private to the specific pointer table, and that memory is returned to the system when `Perl_ptr_table_free` is called. Additionally, allocation and release are both less CPU intensive.

*UNDERBAR*

The UNDERBAR macro now calls `find_rundefsv`. dUNDERBAR is now a noop but should still be used to ensure past and future compatibility.

*String comparison routines renamed*

The `ibcmp_*` functions have been renamed and are now called `foldEQ`, `foldEQ_locale`, and `foldEQ_utf8`. The old names are still available as macros.

*`chop` and `chomp` implementations merged*

The opcode bodies for `chop` and `chomp` and for `schop` and `schomp` have been merged. The implementation functions **Perl_do_chop()** and **Perl_do_chomp()**, never part of the public API, have been merged and moved to a static function in *pp.c*. This shrinks the Perl binary slightly, and should not affect any code outside the core (unless it is relying on the order of side-effects when `chomp` is passed a *list* of values).

## Selected Bug Fixes

### I/O

- Perl no longer produces this warning:

```
$ perl -we 'open(my $f, ">", \my $x); binmode($f, "scalar")'
Use of uninitialized value in binmode at -e line 1.
```

- Opening a glob reference via `open($fh, ">", \*glob)` no longer causes the glob to be corrupted when the filehandle is printed to. This would cause Perl to crash whenever the glob's contents were accessed [perl #77492].
- PerlIO no longer crashes when called recursively, such as from a signal handler. Now it just leaks memory [perl #75556].
- Most I/O functions were not warning for unopened handles unless the "closed" and "unopened" warnings categories were both enabled. Now only `use warnings 'unopened'` is necessary to trigger these warnings, as had always been the intention.
- There have been several fixes to PerlIO layers:

  When `binmode(FH, ":crlf")` pushes the `:crlf` layer on top of the stack, it no longer enables crlf layers lower in the stack so as to avoid unexpected results [perl #38456].

  Opening a file in `:raw` mode now does what it advertises to do (first open the file, then `binmode` it), instead of simply leaving off the top layer [perl #80764].

  The three layers `:pop`, `:utf8`, and `:bytes` didn't allow stacking when opening a file. For example this:

```
open(FH, ">:pop:perlio", "some.file") or die $!;
```

  would throw an "Invalid argument" error. This has been fixed in this release [perl #82484].

### Regular Expression Bug Fixes

- The regular expression engine no longer loops when matching `"\N{LATIN SMALL LIGATURE FF}" =~ /f+/i` and similar expressions [perl #72998] (5.12.1).
- The trie runtime code should no longer allocate massive amounts of memory, fixing #74484.
- Syntax errors in `(?{...})` blocks no longer cause panic messages [perl #2353].
- A pattern like `(?:(o){2})?` no longer causes a "panic" error [perl #39233].
- A fatal error in regular expressions containing `(.*?)` when processing UTF-8 data has been fixed [perl #75680] (5.12.2).
- An erroneous regular expression engine optimisation that caused regex verbs like `*COMMIT` sometimes to be ignored has been removed.
- The regular expression bracketed character class `[\8\9]` was effectively the same as `[89\000]`, incorrectly matching a NULL character. It also gave incorrect warnings that the `8` and `9` were ignored. Now `[\8\9]` is the same as `[89]` and gives legitimate warnings that `\8` and `\9` are unrecognized escape sequences, passed-through.

- A regular expression match in the right-hand side of a global substitution (`s///g`) that is in the same scope will no longer cause match variables to have the wrong values on subsequent iterations. This can happen when an array or hash subscript is interpolated in the right-hand side, as in `s|(.)|@a{ print($1), /./ }|g` [perl #19078].
- Several cases in which characters in the Latin−1 non-ASCII range (0x80 to 0xFF) used not to match themselves, or used to match both a character class and its complement, have been fixed. For instance, U+00E2 could match both `\w` and `\W` [perl #78464] [perl #18281] [perl #60156].
- Matching a Unicode character against an alternation containing characters that happened to match continuation bytes in the former's UTF8 representation (like `qq{\x{30ab}} =~ /\xab|\xa9/`) would cause erroneous warnings [perl #70998].
- The trie optimisation was not taking empty groups into account, preventing ''foo'' from matching `/\A(?:(?:)foo|bar|zot)\z/` [perl #78356].
- A pattern containing a `+` inside a lookahead would sometimes cause an incorrect match failure in a global match (for example, `/(?=(\S+))/g`) [perl #68564].
- A regular expression optimisation would sometimes cause a match with a `{n,m}` quantifier to fail when it should have matched [perl #79152].
- Case-insensitive matching in regular expressions compiled under `use locale` now works much more sanely when the pattern or target string is internally encoded in UTF8. Previously, under these conditions the localeness was completely lost. Now, code points above 255 are treated as Unicode, but code points between 0 and 255 are treated using the current locale rules, regardless of whether the pattern or the string is encoded in UTF8. The few case-insensitive matches that cross the 255/256 boundary are not allowed. For example, 0xFF does not caselessly match the character at 0x178, LATIN CAPITAL LETTER Y WITH DIAERESIS, because 0xFF may not be LATIN SMALL LETTER Y in the current locale, and Perl has no way of knowing if that character even exists in the locale, much less what code point it is.
- The `(?|...)` regular expression construct no longer crashes if the final branch has more sets of capturing parentheses than any other branch. This was fixed in Perl 5.10.1 for the case of a single branch, but that fix did not take multiple branches into account [perl #84746].
- A bug has been fixed in the implementation of `{...}` quantifiers in regular expressions that prevented the code block in `/((\w+)(?{ print $2 }))/{2}` from seeing the `$2` sometimes [perl #84294].

### Syntax/Parsing Bugs

- `when (scalar) {...}` no longer crashes, but produces a syntax error [perl #74114] (5.12.1).
- A label right before a string eval (`foo: eval $string`) no longer causes the label to be associated also with the first statement inside the eval [perl #74290] (5.12.1).
- The `no 5.13.2` form of `no` no longer tries to turn on features or pragmata (like strict) [perl #70075] (5.12.2).
- `BEGIN {require 5.12.0}` now behaves as documented, rather than behaving identically to `use 5.12.0`. Previously, `require` in a `BEGIN` block was erroneously executing the `use feature ':5.12.0'` and `use strict` behaviour, which only `use` was documented to provide [perl #69050].
- A regression introduced in Perl 5.12.0, making `my $x = 3; $x = length(undef)` result in `$x` set to `3` has been fixed. `$x` will now be `undef` [perl #85508] (5.12.2).
- When strict ''refs'' mode is off, `%{...}` in rvalue context returns `undef` if its argument is undefined. An optimisation introduced in Perl 5.12.0 to make `keys %{...}` faster when used as a boolean did not take this into account, causing `keys %{+undef}` (and `keys %$foo` when `$foo` is undefined) to be an error, which it should be so in strict mode only [perl #81750].
- Constant-folding used to cause

  ```
  $text =~ ( 1 ? /phoo/ : /bear/)
  ```

  to turn into

```
$text =~ /phoo/
```

at compile time. Now it correctly matches against $_ [perl #20444].

- Parsing Perl code (either with string `eval` or by loading modules) from within a `UNITCHECK` block no longer causes the interpreter to crash [perl #70614].
- String `eval`s no longer fail after 2 billion scopes have been compiled [perl #83364].
- The parser no longer hangs when encountering certain Unicode characters, such as U+387 [perl #74022].
- Defining a constant with the same name as one of Perl's special blocks (like `INIT`) stopped working in 5.12.0, but has now been fixed [perl #78634].
- A reference to a literal value used as a hash key (`$hash{\"foo"}`) used to be stringified, even if the hash was tied [perl #79178].
- A closure containing an `if` statement followed by a constant or variable is no longer treated as a constant [perl #63540].
- `state` can now be used with attributes. It used to mean the same thing as `my` if any attributes were present [perl #68658].
- Expressions like `@$a > 3` no longer cause `$a` to be mentioned in the ''Use of uninitialized value in numeric gt'' warning when `$a` is undefined (since it is not part of the `>` expression, but the operand of the `@`) [perl #72090].
- Accessing an element of a package array with a hard-coded number (as opposed to an arbitrary expression) would crash if the array did not exist. Usually the array would be autovivified during compilation, but typeglob manipulation could remove it, as in these two cases which used to crash:

```
*d = *a;  print $d[0];
undef *d; print $d[0];
```

- The **-C** command-line option, when used on the shebang line, can now be followed by other options [perl #72434].
- The `B` module was returning `B::OP`s instead of `B::LOGOP`s for `entertry` [perl #80622]. This was due to a bug in the Perl core, not in `B` itself.

### Stashes, Globs and Method Lookup

Perl 5.10.0 introduced a new internal mechanism for caching MROs (method resolution orders, or lists of parent classes; aka ''isa'' caches) to make method lookup faster (so `@ISA` arrays would not have to be searched repeatedly). Unfortunately, this brought with it quite a few bugs. Almost all of these have been fixed now, along with a few MRO-related bugs that existed before 5.10.0:

- The following used to have erratic effects on method resolution, because the ''isa'' caches were not reset or otherwise ended up listing the wrong classes. These have been fixed.

  Aliasing packages by assigning to globs [perl #77358]
  Deleting packages by deleting their containing stash elements
  Undefining the glob containing a package (`undef *Foo::`)
  Undefining an ISA glob (`undef *Foo::ISA`)
  Deleting an ISA stash element (`delete $Foo::{ISA}`)
  Sharing `@ISA` arrays between classes (via `*Foo::ISA = \@Bar::ISA` or `*Foo::ISA = *Bar::ISA`) [perl #77238]

  `undef *Foo::ISA` would even stop a new `@Foo::ISA` array from updating caches.

- Typeglob assignments would crash if the glob's stash no longer existed, so long as the glob assigned to were named `ISA` or the glob on either side of the assignment contained a subroutine.
- `PL_isarev`, which is accessible to Perl via `mro::get_isarev` is now updated properly when packages are deleted or removed from the `@ISA` of other classes. This allows many packages to be created and deleted without causing a memory leak [perl #75176].

In addition, various other bugs related to typeglobs and stashes have been fixed:

- Some work has been done on the internal pointers that link between symbol tables (stashes), typeglobs, and subroutines. This has the effect that various edge cases related to deleting stashes or stash entries (for example, <%FOO:: = ()>), and complex typeglob or code-reference aliasing, will no longer crash the interpreter.
- Assigning a reference to a glob copy now assigns to a glob slot instead of overwriting the glob with a scalar [perl #1804] [perl #77508].
- A bug when replacing the glob of a loop variable within the loop has been fixed [perl #21469]. This means the following code will no longer crash:

```
for $x (...) {
    *x = *y;
}
```

- Assigning a glob to a PVLV used to convert it to a plain string. Now it works correctly, and a PVLV can hold a glob. This would happen when a nonexistent hash or array element was passed to a subroutine:

```
sub { $_[0] = *foo }->($hash{key});
# $_[0] would have been the string "*main::foo"
```

  It also happened when a glob was assigned to, or returned from, an element of a tied array or hash [perl #36051].
- When trying to report `Use of uninitialized value $Foo::BAR`, crashes could occur if the glob holding the global variable in question had been detached from its original stash by, for example, `delete $::{"Foo::"}`. This has been fixed by disabling the reporting of variable names in those cases.
- During the restoration of a localised typeglob on scope exit, any destructors called as a result would be able to see the typeglob in an inconsistent state, containing freed entries, which could result in a crash. This would affect code like this:

```
local *@;
eval { die bless [] }; # puts an object in $@
sub DESTROY {
  local $@; # boom
}
```

  Now the glob entries are cleared before any destructors are called. This also means that destructors can vivify entries in the glob. So Perl tries again and, if the entries are re-created too many times, dies with a ‘‘panic: gp_free ...’’ error message.
- If a typeglob is freed while a subroutine attached to it is still referenced elsewhere, the subroutine is renamed to `__ANON__` in the same package, unless the package has been undefined, in which case the `__ANON__` package is used. This could cause packages to be sometimes autovivified, such as if the package had been deleted. Now this no longer occurs. The `__ANON__` package is also now used when the original package is no longer attached to the symbol table. This avoids memory leaks in some cases [perl #87664].
- Subroutines and package variables inside a package whose name ends with `::` can now be accessed with a fully qualified name.

### Unicode

- What has become known as ‘‘the Unicode Bug’’ is almost completely resolved in this release. Under `use feature 'unicode_strings'` (which is automatically selected by `use 5.012` and above), the internal storage format of a string no longer affects the external semantics. [perl #58182].

  There are two known exceptions:

  1. The now-deprecated, user-defined case-changing functions require utf8−encoded strings to operate. The CPAN module Unicode::Casing has been written to replace this feature without its drawbacks, and the feature is scheduled to be removed in 5.16.

2. **quotemeta()** (and its in-line equivalent `\Q`) can also give different results depending on whether a string is encoded in UTF−8. See ‘‘The ’’Unicode Bug"" in perlunicode.

- Handling of Unicode non-character code points has changed. Previously they were mostly considered illegal, except that in some place only one of the 66 of them was known. The Unicode Standard considers them all legal, but forbids their ‘‘open interchange’’. This is part of the change to allow internal use of any code point (see ‘‘Core Enhancements’’). Together, these changes resolve [perl #38722], [perl #51918], [perl #51936], and [perl #63446].
- Case-insensitive `"/i"` regular expression matching of Unicode characters that match multiple characters now works much more as intended. For example

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /ffi/ui
  ```

  and

  ```
  "ffi" =~ /\N{LATIN SMALL LIGATURE FFI}/ui
  ```

  are both true. Previously, there were many bugs with this feature. What hasn’t been fixed are the places where the pattern contains the multiple characters, but the characters are split up by other things, such as in

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /(f)(f)i/ui
  ```

  or

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /ffi*/ui
  ```

  or

  ```
  "\N{LATIN SMALL LIGATURE FFI}" =~ /[a-f][f-m][g-z]/ui
  ```

  None of these match.

  Also, this matching doesn’t fully conform to the current Unicode Standard, which asks that the matching be made upon the NFD (Normalization Form Decomposed) of the text. However, as of this writing (April 2010), the Unicode Standard is currently in flux about what they will recommend doing with regard in such scenarios. It may be that they will throw out the whole concept of multi-character matches. [perl #71736].
- Naming a deprecated character in `\N{`*`NAME`*`}` no longer leaks memory.
- We fixed a bug that could cause `\N{`*`NAME`*`}` constructs followed by a single `"."` to be parsed incorrectly [perl #74978] (5.12.1).
- `chop` now correctly handles characters above `"\x{7fffffff}"` [perl #73246].
- Passing to `index` an offset beyond the end of the string when the string is encoded internally in UTF8 no longer causes panics [perl #75898].
- **warn()** and **die()** now respect utf8−encoded scalars [perl #45549].
- Sometimes the UTF8 length cache would not be reset on a value returned by substr, causing `length(substr($uni_string, ...))` to give wrong answers. With `${^UTF8CACHE}` set to −1, it would also produce a ‘‘panic’’ error message [perl #77692].

### Ties, Overloading and Other Magic

- Overloading now works properly in conjunction with tied variables. What formerly happened was that most ops checked their arguments for overloading *before* checking for magic, so for example an overloaded object returned by a tied array access would usually be treated as not overloaded [RT #57012].
- Various instances of magic (like tie methods) being called on tied variables too many or too few times have been fixed:
  - `$tied->()` did not always call FETCH [perl #8438].
  - Filetest operators and `y///` and `tr///` were calling FETCH too many times.
  - The `=` operator used to ignore magic on its right-hand side if the scalar happened to hold a typeglob (if a typeglob was the last thing returned from or assigned to a tied scalar) [perl #77498].

- Dereference operators used to ignore magic if the argument was a reference already (such as from a previous FETCH) [perl #72144].
- `splice` now calls set-magic (so changes made by `splice @ISA` are respected by method calls) [perl #78400].
- In-memory files created by `open($fh, ">", \$buffer)` were not calling FETCH/STORE at all [perl #43789] (5.12.2).
- **utf8::is_utf8()** now respects get-magic (like `$1`) (5.12.1).

- Non-commutative binary operators used to swap their operands if the same tied scalar was used for both operands and returned a different value for each FETCH. For instance, if `$t` returned 2 the first time and 3 the second, then `$t/$t` would evaluate to 1.5. This has been fixed [perl #87708].
- String `eval` now detects taintedness of overloaded or tied arguments [perl #75716].
- String `eval` and regular expression matches against objects with string overloading no longer cause memory corruption or crashes [perl #77084].
- readline now honors `<>` overloading on tied arguments.
- `<expr>` always respects overloading now if the expression is overloaded.

  Because ''<> as glob'' was parsed differently from ''<> as filehandle'' from 5.6 onwards, something like `<$foo[0]>` did not handle overloading, even if `$foo[0]` was an overloaded object. This was contrary to the documentation for overload, and meant that `<>` could not be used as a general overloaded iterator operator.
- The fallback behaviour of overloading on binary operators was asymmetric [perl #71286].
- Magic applied to variables in the main package no longer affects other packages. See ''Magic variables outside the main package'' above [perl #76138].
- Sometimes magic (ties, taintedness, etc.) attached to variables could cause an object to last longer than it should, or cause a crash if a tied variable were freed from within a tie method. These have been fixed [perl #81230].
- DESTROY methods of objects implementing ties are no longer able to crash by accessing the tied variable through a weak reference [perl #86328].
- Fixed a regression of **kill()** when a match variable is used for the process ID to kill [perl #75812].
- `$AUTOLOAD` used to remain tainted forever if it ever became tainted. Now it is correctly untainted if an autoloaded method is called and the method name was not tainted.
- `sprintf` now dies when passed a tainted scalar for the format. It did already die for arbitrary expressions, but not for simple scalars [perl #82250].
- `lc`, `uc`, `lcfirst`, and `ucfirst` no longer return untainted strings when the argument is tainted. This has been broken since perl 5.8.9 [perl #87336].

### The Debugger

- The Perl debugger now also works in taint mode [perl #76872].
- Subroutine redefinition works once more in the debugger [perl #48332].
- When **-d** is used on the shebang (`#!`) line, the debugger now has access to the lines of the main program. In the past, this sometimes worked and sometimes did not, depending on the order in which things happened to be arranged in memory [perl #71806].
- A possible memory leak when using **caller()** to set `@DB::args` has been fixed (5.12.2).
- Perl no longer stomps on `$DB::single`, `$DB::trace`, and `$DB::signal` if these variables already have values when `$^P` is assigned to [perl #72422].
- `#line` directives in string evals were not properly updating the arrays of lines of code (`@{"_< ..."}`) that the debugger (or any debugging or profiling module) uses. In threaded builds, they were not being updated at all. In non-threaded builds, the line number was ignored, so any change to the existing line number would cause the lines to be misnumbered [perl #79442].

### Threads

- Perl no longer accidentally clones lexicals in scope within active stack frames in the parent when creating a child thread [perl #73086].
- Several memory leaks in cloning and freeing threaded Perl interpreters have been fixed [perl #77352].
- Creating a new thread when directory handles were open used to cause a crash, because the handles were not cloned, but simply passed to the new thread, resulting in a double free.

  Now directory handles are cloned properly on Windows and on systems that have a `fchdir` function. On other systems, new threads simply do not inherit directory handles from their parent threads [perl #75154].
- The typeglob `*,`, which holds the scalar variable `$,` (output field separator), had the wrong reference count in child threads.
- [perl #78494] When pipes are shared between threads, the `close` function (and any implicit close, such as on thread exit) no longer blocks.
- Perl now does a timely cleanup of SVs that are cloned into a new thread but then discovered to be orphaned (that is, their owners are *not* cloned). This eliminates several ‘‘scalars leaked’’ warnings when joining threads.

### Scoping and Subroutines

- Lvalue subroutines are again able to return copy-on-write scalars. This had been broken since version 5.10.0 [perl #75656] (5.12.3).
- `require` no longer causes `caller` to return the wrong file name for the scope that called `require` and other scopes higher up that had the same file name [perl #68712].
- `sort` with a `($$)`−prototyped comparison routine used to cause the value of `@_` to leak out of the sort. Taking a reference to `@_` within the sorting routine could cause a crash [perl #72334].
- Match variables (like `$1`) no longer persist between calls to a sort subroutine [perl #76026].
- Iterating with `foreach` over an array returned by an lvalue sub now works [perl #23790].
- `$@` is now localised during calls to `binmode` to prevent action at a distance [perl #78844].
- Calling a closure prototype (what is passed to an attribute handler for a closure) now results in a ‘‘Closure prototype called’’ error message instead of a crash [perl #68560].
- Mentioning a read-only lexical variable from the enclosing scope in a string `eval` no longer causes the variable to become writable [perl #19135].

### Signals

- Within signal handlers, `$!` is now implicitly localized.
- CHLD signals are no longer unblocked after a signal handler is called if they were blocked before by `POSIX::sigprocmask` [perl #82040].
- A signal handler called within a signal handler could cause leaks or double-frees. Now fixed [perl #76248].

### Miscellaneous Memory Leaks

- Several memory leaks when loading XS modules were fixed (5.12.2).
- **substr()**, **pos()**, **keys()**, and **vec()** could, when used in combination with lvalues, result in leaking the scalar value they operate on, and cause its destruction to happen too late. This has now been fixed.
- The postincrement and postdecrement operators, `++` and `−−`, used to cause leaks when used on references. This has now been fixed.
- Nested `map` and `grep` blocks no longer leak memory when processing large lists [perl #48004].
- `use` *`VERSION`* and `no` *`VERSION`* no longer leak memory [perl #78436] [perl #69050].
- `.=` followed by `<>` or `readline` would leak memory if `$/` contained characters beyond the octet range and the scalar assigned to happened to be encoded as UTF8 internally [perl #72246].

- `eval 'BEGIN{die}'` no longer leaks memory on non-threaded builds.

### Memory Corruption and Crashes

- **glob()** no longer crashes when `%File::Glob::` is empty and `CORE::GLOBAL::glob` isn't present [perl #75464] (5.12.2).
- **readline()** has been fixed when interrupted by signals so it no longer returns the ''same thing'' as before or random memory.
- When assigning a list with duplicated keys to a hash, the assignment used to return garbage and/or freed values:

  ```
  @a = %h = (list with some duplicate keys);
  ```

  This has now been fixed [perl #31865].
- The mechanism for freeing objects in globs used to leave dangling pointers to freed SVs, meaning Perl users could see corrupted state during destruction.

  Perl now frees only the affected slots of the GV, rather than freeing the GV itself. This makes sure that there are no dangling refs or corrupted state during destruction.
- The interpreter no longer crashes when freeing deeply-nested arrays of arrays. Hashes have not been fixed yet [perl #44225].
- Concatenating long strings under `use encoding` no longer causes Perl to crash [perl #78674].
- Calling `->import` on a class lacking an import method could corrupt the stack, resulting in strange behaviour. For instance,

  ```
  push @a, "foo", $b = bar->import;
  ```

  would assign ''foo'' to `$b` [perl #63790].
- The `recv` function could crash when called with the MSG_TRUNC flag [perl #75082].
- `formline` no longer crashes when passed a tainted format picture. It also taints `$^A` now if its arguments are tainted [perl #79138].
- A bug in how we process filetest operations could cause a segfault. Filetests don't always expect an op on the stack, so we now use TOPs only if we're sure that we're not `stat`ing the _ filehandle. This is indicated by `OPf_KIDS` (as checked in ck_ftst) [perl #74542] (5.12.1).
- **unpack()** now handles scalar context correctly for `%32H` and `%32u`, fixing a potential crash. **split()** would crash because the third item on the stack wasn't the regular expression it expected. `unpack("%2H", ...)` would return both the unpacked result and the checksum on the stack, as would `unpack("%2u", ...)` [perl #73814] (5.12.2).

### Fixes to Various Perl Operators

- The `&`, `|`, and `^` bitwise operators no longer coerce read-only arguments [perl #20661].
- Stringifying a scalar containing ''-0.0'' no longer has the effect of turning false into true [perl #45133].
- Some numeric operators were converting integers to floating point, resulting in loss of precision on 64-bit platforms [perl #77456].
- **sprintf()** was ignoring locales when called with constant arguments [perl #78632].
- Combining the vector (`%v`) flag and dynamic precision would cause `sprintf` to confuse the order of its arguments, making it treat the string as the precision and vice-versa [perl #83194].

### Bugs Relating to the C API

- The C-level `lex_stuff_pvn` function would sometimes cause a spurious syntax error on the last line of the file if it lacked a final semicolon [perl #74006] (5.12.1).
- The `eval_sv` and `eval_pv` C functions now set `$@` correctly when there is a syntax error and no `G_KEEPERR` flag, and never set it if the `G_KEEPERR` flag is present [perl #3719].
- The XS multicall API no longer causes subroutines to lose reference counts if called via the multicall interface from within those very subroutines. This affects modules like List::Util. Calling one of its functions with an active subroutine as the first argument could cause a crash [perl #78070].

- The `SvPVbyte` function available to XS modules now calls magic before downgrading the SV, to avoid warnings about wide characters [perl #72398].
- The ref types in the typemap for XS bindings now support magical variables [perl #72684].
- `sv_catsv_flags` no longer calls `mg_get` on its second argument (the source string) if the flags passed to it do not include SV_GMAGIC. So it now matches the documentation.
- `my_strftime` no longer leaks memory. This fixes a memory leak in `POSIX::strftime` [perl #73520].
- *XSUB.h* now correctly redefines fgets under PERL_IMPLICIT_SYS [perl #55049] (5.12.1).
- XS code using **fputc()** or **fputs()** on Windows could cause an error due to their arguments being swapped [perl #72704] (5.12.1).
- A possible segfault in the `T_PTROBJ` default typemap has been fixed (5.12.2).
- A bug that could cause "Unknown error" messages when `call_sv(code, G_EVAL)` is called from an XS destructor has been fixed (5.12.2).

## Known Problems

This is a list of significant unresolved issues which are regressions from earlier versions of Perl or which affect widely-used CPAN modules.

- `List::Util::first` misbehaves in the presence of a lexical `$_` (typically introduced by `my $_` or implicitly by `given`). The variable that gets set for each iteration is the package variable `$_`, not the lexical `$_`.

  A similar issue may occur in other modules that provide functions which take a block as their first argument, like

  ```
  foo { ... $_ ...} list
  ```

  See also: <https://github.com/Perl/perl5/issues/9798>
- **readline()** returns an empty string instead of a cached previous value when it is interrupted by a signal
- The changes in prototype handling break Switch. A patch has been sent upstream and will hopefully appear on CPAN soon.
- The upgrade to *ExtUtils-MakeMaker-6.57_05* has caused some tests in the *Module-Install* distribution on CPAN to fail. (Specifically, *02_mymeta.t* tests 5 and 21; *18_all_from.t* tests 6 and 15; *19_authors.t* tests 5, 13, 21, and 29; and *20_authors_with_special_characters.t* tests 6, 15, and 23 in version 1.00 of that distribution now fail.)
- On VMS, `Time::HiRes` tests will fail due to a bug in the CRTL's implementation of `setitimer`: previous timer values would be cleared if a timer expired but not if the timer was reset before expiring. HP OpenVMS Engineering have corrected the problem and will release a patch in due course (Quix case # QXCM1001115136).
- On VMS, there were a handful of `Module::Build` test failures we didn't get to before the release; please watch CPAN for updates.

## Errata

### keys(), values(), and each() work on arrays

You can now use the **keys()**, **values()**, and **each()** builtins on arrays; previously you could use them only on hashes. See perlfunc for details. This is actually a change introduced in perl 5.12.0, but it was missed from that release's perl5120delta.

### split() and @_

**split()** no longer modifies `@_` when called in scalar or void context. In void context it now produces a "Useless use of split" warning. This was also a perl 5.12.0 change that missed the perldelta.

## Obituary

Randy Kobes, creator of http://kobesearch.cpan.org/ and contributor/maintainer to several core Perl toolchain modules, passed away on September 18, 2010 after a battle with lung cancer. The community was richer for his involvement. He will be missed.

## Acknowledgements

Perl 5.14.0 represents one year of development since Perl 5.12.0 and contains nearly 550,000 lines of changes across nearly 3,000 files from 150 authors and committers.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.14.0:

Aaron Crane, Abhijit Menon-Sen, Abigail, Ævar Arnfjörð Bjarmason, Alastair Douglas, Alexander Alekseev, Alexander Hartmaier, Alexandr Ciornii, Alex Davies, Alex Vandiver, Ali Polatel, Allen Smith, Andreas König, Andrew Rodland, Andy Armstrong, Andy Dougherty, Aristotle Pagaltzis, Arkturuz, Arvan, A. Sinan Unur, Ben Morrow, Bo Lindbergh, Boris Ratner, Brad Gilbert, Bram, brian d foy, Brian Phillips, Casey West, Charles Bailey, Chas. Owens, Chip Salzenberg, Chris 'BinGOs' Williams, chromatic, Craig A. Berry, Curtis Jewell, Dagfinn Ilmari Mannsåker, Dan Dascalescu, Dave Rolsky, David Caldwell, David Cantrell, David Golden, David Leadbeater, David Mitchell, David Wheeler, Eric Brine, Father Chrysostomos, Fingle Nark, Florian Ragwitz, Frank Wiegand, Franz Fasching, Gene Sullivan, George Greer, Gerard Goossen, Gisle Aas, Goro Fuji, Grant McLean, gregor herrmann, H.Merijn Brand, Hongwen Qiu, Hugo van der Sanden, Ian Goodacre, James E Keenan, James Mastros, Jan Dubois, Jay Hannah, Jerry D. Hedden, Jesse Vincent, Jim Cromie, Jirka HruXka, John Peacock, Joshua ben Jore, Joshua Pritikin, Karl Williamson, Kevin Ryde, kmx, Lars DXXXXXX XXX, Larwan Berke, Leon Brocard, Leon Timmermans, Lubomir Rintel, Lukas Mai, Maik Hentsche, Marty Pauley, Marvin Humphrey, Matt Johnson, Matt S Trout, Max Maischein, Michael Breen, Michael Fig, Michael G Schwern, Michael Parker, Michael Stevens, Michael Witten, Mike Kelly, Moritz Lenz, Nicholas Clark, Nick Cleaton, Nick Johnston, Nicolas Kaiser, Niko Tyni, Noirin Shirley, Nuno Carvalho, Paul Evans, Paul Green, Paul Johnson, Paul Marquess, Peter J. Holzer, Peter John Acklam, Peter Martini, Philippe Bruhat (BooK), Piotr Fusik, Rafael Garcia-Suarez, Rainer Tammer, Reini Urban, Renee Baecker, Ricardo Signes, Richard Möhn, Richard Soderberg, Rob Hoelz, Robin Barker, Ruslan Zakirov, Salvador Fandiño, Salvador Ortiz Garcia, Shlomi Fish, Sinan Unur, Sisyphus, Slaven Rezic, Steffen Müller, Steve Hay, Steven Schubiger, Steve Peters, Sullivan Beck, Tatsuhiko Miyagawa, Tim Bunce, Todd Rinaldo, Tom Christiansen, Tom Hukins, Tony Cook, Tye McQueen, Vadim Konovalov, Vernon Lyon, Vincent Pit, Walt Mankowski, Wolfram Humann, Yves Orton, Zefram, and Zsbán Ambrus.

This is woefully incomplete as it's automatically generated from version control history. In particular, it doesn't include the names of the (very much appreciated) contributors who reported issues in previous versions of Perl that helped make Perl 5.14.0 better. For a more complete list of all of Perl's historical contributors, please see the `AUTHORS` file in the Perl 5.14.0 distribution.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the Perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who are able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please use this address for security issues in the Perl core *only*, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5141delta − what is new for perl v5.14.1

## DESCRIPTION

This document describes differences between the 5.14.0 release and the 5.14.1 release.

If you are upgrading from an earlier release such as 5.12.0, first read perl5140delta, which describes differences between 5.12.0 and 5.14.0.

## Core Enhancements

No changes since 5.14.0.

## Security

No changes since 5.14.0.

## Incompatible Changes

There are no changes intentionally incompatible with 5.14.0. If any exist, they are bugs and reports are welcome.

## Deprecations

There have been no deprecations since 5.14.0.

## Modules and Pragmata

### New Modules and Pragmata

None

### Updated Modules and Pragmata

- B::Deparse has been upgraded from version 1.03 to 1.04, to address two regressions in Perl 5.14.0:

  Deparsing of the `glob` operator and its diamond (`<>`) form now works again. [perl #90898]

  The presence of subroutines named `::::` or `::::::` no longer causes B::Deparse to hang.

- Pod::Perldoc has been upgraded from version 3.15_03 to 3.15_04.

  It corrects the search paths on VMS. [perl #90640]

### Removed Modules and Pragmata

None

## Documentation

### New Documentation

None

### Changes to Existing Documentation

*perlfunc*

- `given`, `when` and `default` are now listed in perlfunc.
- Documentation for `use` now includes a pointer to *if.pm*.

*perllol*

- perllol has been expanded with examples using the new `push $scalar` syntax introduced in Perl 5.14.0.

*perlop*

- The explanation of bitwise operators has been expanded to explain how they work on Unicode strings.
- The section on the triple-dot or yada-yada operator has been moved up, as it used to separate two closely related sections about the comma operator.
- More examples for `m//g` have been added.
- The `<<\FOO` here-doc syntax has been documented.

*perlrun*

- perlrun has undergone a significant clean-up. Most notably, the **−0x...** form of the **−0** flag has been clarified, and the final section on environment variables has been corrected and expanded.

*POSIX*

- The invocation documentation for `WIFEXITED`, `WEXITSTATUS`, `WIFSIGNALED`, `WTERMSIG`, `WIFSTOPPED`, and `WSTOPSIG` was corrected.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

None

### Changes to Existing Diagnostics

None

## Utility Changes

None

## Configuration and Compilation

- *regexp.h* has been modified for compatibility with GCC's `-Werror` option, as used by some projects that include perl's header files.

## Testing

- Some test failures in *dist/Locale-Maketext/t/09_compile.t* that could occur depending on the environment have been fixed. [perl #89896]
- A watchdog timer for *t/re/re.t* was lengthened to accommodate SH-4 systems which were unable to complete the tests before the previous timer ran out.

## Platform Support

### New Platforms

None

### Discontinued Platforms

None

### Platform-Specific Notes

*Solaris*

- Documentation listing the Solaris packages required to build Perl on Solaris 9 and Solaris 10 has been corrected.

*Mac OS X*

- The *lib/locale.t* test script has been updated to work on the upcoming Lion release.
- Mac OS X specific compilation instructions have been clarified.

*Ubuntu Linux*

- The ODBM_File installation process has been updated with the new library paths on Ubuntu natty.

## Internal Changes

- The compiled representation of formats is now stored via the mg_ptr of their PERL_MAGIC_fm. Previously it was stored in the string buffer, beyond **SvLEN()**, the regular end of the string. **SvCOMPILED()** and SvCOMPILED_{on,off}() now exist solely for compatibility for XS code. The first is always 0, the other two now no-ops.

## Bug Fixes

- A bug has been fixed that would cause a ''Use of freed value in iteration'' error if the next two hash elements that would be iterated over are deleted. [perl #85026]
- Passing the same constant subroutine to both `index` and `formline` no longer causes one or the other to fail. [perl #89218]
- 5.14.0 introduced some memory leaks in regular expression character classes such as `[\w\s]`, which have now been fixed.
- An edge case in regular expression matching could potentially loop. This happened only under `/i` in bracketed character classes that have characters with multi-character folds, and the target string to match against includes the first portion of the fold, followed by another character that has a multi-character fold that begins with the remaining portion of the fold, plus some more.

```
"s\N{U+DF}" =~ /[\x{DF}foo]/i
```

is one such case. `\xDF` folds to `"ss"`.

- Several Unicode case-folding bugs have been fixed.
- The new (in 5.14.0) regular expression modifier `/a` when repeated like `/aa` forbids the characters outside the ASCII range that match characters inside that range from matching under `/i`. This did not work under some circumstances, all involving alternation, such as:

```
"\N{KELVIN SIGN}" =~ /k|foo/iaa;
```

succeeded inappropriately. This is now fixed.

- Fixed a case where it was possible that a freed buffer may have been read from when parsing a here document.

## Acknowledgements

Perl 5.14.1 represents approximately four weeks of development since Perl 5.14.0 and contains approximately 3500 lines of changes across 38 files from 17 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.14.1:

Bo Lindbergh, Claudio Ramirez, Craig A. Berry, David Leadbeater, Father Chrysostomos, Jesse Vincent, Jim Cromie, Justin Case, Karl Williamson, Leo Lapworth, Nicholas Clark, Nobuhiro Iwamatsu, smash, Tom Christiansen, Ton Hospel, Vladimir Timofeev, and Zsbán Ambrus.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5142delta − what is new for perl v5.14.2

## DESCRIPTION

This document describes differences between the 5.14.1 release and the 5.14.2 release.

If you are upgrading from an earlier release such as 5.14.0, first read perl5141delta, which describes differences between 5.14.0 and 5.14.1.

## Core Enhancements

No changes since 5.14.0.

## Security

### `File::Glob::bsd_glob()` memory error with GLOB_ALTDIRFUNC (CVE−2011−2728).

Calling `File::Glob::bsd_glob` with the unsupported flag GLOB_ALTDIRFUNC would cause an access violation / segfault. A Perl program that accepts a flags value from an external source could expose itself to denial of service or arbitrary code execution attacks. There are no known exploits in the wild. The problem has been corrected by explicitly disabling all unsupported flags and setting unused function pointers to null. Bug reported by Clément Lecigne.

### `Encode` decode_xs n−byte heap-overflow (CVE−2011−2939)

A bug in `Encode` could, on certain inputs, cause the heap to overflow. This problem has been corrected. Bug reported by Robert Zacek.

## Incompatible Changes

There are no changes intentionally incompatible with 5.14.0. If any exist, they are bugs and reports are welcome.

## Deprecations

There have been no deprecations since 5.14.0.

## Modules and Pragmata

### New Modules and Pragmata

None

### Updated Modules and Pragmata

- CPAN has been upgraded from version 1.9600 to version 1.9600_01.

  CPAN::Distribution has been upgraded from version 1.9602 to 1.9602_01.

  Backported bugfixes from CPAN version 1.9800. Ensures proper detection of `configure_requires` prerequisites from CPAN Meta files in the case where `dynamic_config` is true. [rt.cpan.org #68835]

  Also ensures that `configure_requires` is only checked in META files, not MYMETA files, so protect against MYMETA generation that drops `configure_requires`.

- Encode has been upgraded from version 2.42 to 2.42_01.

  See ‘‘Security’’.

- File::Glob has been upgraded from version 1.12 to version 1.13.

  See ‘‘Security’’.

- PerlIO::scalar has been upgraded from version 0.11 to 0.11_01.

  It fixes a problem with `open my $fh, ">", \$scalar` not working if `$scalar` is a copy-on-write scalar.

### Removed Modules and Pragmata

None

## Platform Support

### New Platforms

None

### Discontinued Platforms

None

### Platform-Specific Notes

HP-UX PA-RISC/64 now supports gcc-4.x

A fix to correct the socketsize now makes the test suite pass on HP-UX PA-RISC for 64bitall builds.

Building on OS X 10.7 Lion and Xcode 4 works again

The build system has been updated to work with the build tools under Mac OS X 10.7.

## Bug Fixes

- In `@INC` filters (subroutines returned by subroutines in `@INC`), `$_` used to misbehave: If returned from a subroutine, it would not be copied, but the variable itself would be returned; and freeing `$_` (e.g., with `undef *_`) would cause perl to crash. This has been fixed [perl #91880].
- Perl 5.10.0 introduced some faulty logic that made "U*" in the middle of a pack template equivalent to "U0" if the input string was empty. This has been fixed [perl #90160].
- `caller` no longer leaks memory when called from the DB package if `@DB::args` was assigned to after the first call to `caller`. Carp was triggering this bug [perl #97010].
- `utf8::decode` had a nasty bug that would modify copy-on-write scalars' string buffers in place (i.e., skipping the copy). This could result in hashes having two elements with the same key [perl #91834].
- Localising a tied variable used to make it read-only if it contained a copy-on-write string.
- Elements of restricted hashes (see the fields pragma) containing copy-on-write values couldn't be deleted, nor could such hashes be cleared (`%hash = ()`).
- Locking a hash element that is a glob copy no longer causes subsequent assignment to it to corrupt the glob.
- A panic involving the combination of the regular expression modifiers `/aa` introduced in 5.14.0 and the `\b` escape sequence has been fixed [perl #95964].

## Known Problems

This is a list of some significant unfixed bugs, which are regressions from 5.12.0.

- `PERL_GLOBAL_STRUCT` is broken.

  Since perl 5.14.0, building with `-DPERL_GLOBAL_STRUCT` hasn't been possible. This means that perl currently doesn't work on any platforms that require it to be built this way, including Symbian.

  While `PERL_GLOBAL_STRUCT` now works again on recent development versions of perl, it actually working on Symbian again hasn't been verified.

  We'd be very interested in hearing from anyone working with Perl on Symbian.

## Acknowledgements

Perl 5.14.2 represents approximately three months of development since Perl 5.14.1 and contains approximately 1200 lines of changes across 61 files from 9 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.14.2:

Craig A. Berry, David Golden, Father Chrysostomos, Florian Ragwitz, H.Merijn Brand, Karl Williamson, Nicholas Clark, Pau Amma and Ricardo Signes.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who be able to help assess

the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5143delta − what is new for perl v5.14.3

## DESCRIPTION

This document describes differences between the 5.14.2 release and the 5.14.3 release.

If you are upgrading from an earlier release such as 5.12.0, first read perl5140delta, which describes differences between 5.12.0 and 5.14.0.

## Core Enhancements

No changes since 5.14.0.

## Security

### `Digest` unsafe use of eval (CVE−2011−3597)

The `Digest->new()` function did not properly sanitize input before using it in an **eval()** call, which could lead to the injection of arbitrary Perl code.

In order to exploit this flaw, the attacker would need to be able to set the algorithm name used, or be able to execute arbitrary Perl code already.

This problem has been fixed.

### Heap buffer overrun in 'x' string repeat operator (CVE−2012−5195)

Poorly written perl code that allows an attacker to specify the count to perl's 'x' string repeat operator can already cause a memory exhaustion denial-of-service attack. A flaw in versions of perl before 5.15.5 can escalate that into a heap buffer overrun; coupled with versions of glibc before 2.16, it possibly allows the execution of arbitrary code.

This problem has been fixed.

## Incompatible Changes

There are no changes intentionally incompatible with 5.14.0. If any exist, they are bugs and reports are welcome.

## Deprecations

There have been no deprecations since 5.14.0.

## Modules and Pragmata

### New Modules and Pragmata

None

### Updated Modules and Pragmata

- PerlIO::scalar was updated to fix a bug in which opening a filehandle to a glob copy caused assertion failures (under debugging) or hangs or other erratic behaviour without debugging.
- ODBM_File and NDBM_File were updated to allow building on GNU/Hurd.
- IPC::Open3 has been updated to fix a regression introduced in perl 5.12, which broke `IPC::Open3::open3($in, $out, $err, '−')`. [perl #95748]
- Digest has been upgraded from version 1.16 to 1.16_01.

  See ''Security''.
- Module::CoreList has been updated to version 2.49_04 to add data for this release.

### Removed Modules and Pragmata

None

## Documentation

### New Documentation

None

### Changes to Existing Documentation

*perlcheat*

- perlcheat was updated to 5.14.

## Configuration and Compilation

- h2ph was updated to search correctly gcc include directories on platforms such as Debian with multi-architecture support.

- In Configure, the test for procselfexe was refactored into a loop.

## Platform Support

### New Platforms

None

### Discontinued Platforms

None

### Platform-Specific Notes

FreeBSD

The FreeBSD hints file was corrected to be compatible with FreeBSD 10.0.

Solaris and NetBSD

Configure was updated for ‘‘procselfexe’’ support on Solaris and NetBSD.

HP-UX

README.hpux was updated to note the existence of a broken header in HP-UX 11.00.

Linux

libutil is no longer used when compiling on Linux platforms, which avoids warnings being emitted.

The system gcc (rather than any other gcc which might be in the compiling user's path) is now used when searching for libraries such as `-lm`.

Mac OS X

The locale tests were updated to reflect the behaviour of locales in Mountain Lion.

GNU/Hurd

Various build and test fixes were included for GNU/Hurd.

LFS support was enabled in GNU/Hurd.

NetBSD

The NetBSD hints file was corrected to be compatible with NetBSD 6.*

## Bug Fixes

- A regression has been fixed that was introduced in 5.14, in `/i` regular expression matching, in which a match improperly fails if the pattern is in UTF-8, the target string is not, and a Latin-1 character precedes a character in the string that should match the pattern. [perl #101710]
- In case-insensitive regular expression pattern matching, no longer on UTF-8 encoded strings does the scan for the start of match only look at the first possible position. This caused matches such as `"f\x{FB00}" =~ /ff/i` to fail.
- The sitecustomize support was made relocatableinc aware, so that -Dusesitecustomize and -Duserelocatableinc may be used together.
- The smartmatch operator (`~~`) was changed so that the right-hand side takes precedence during `Any ~~ Object` operations.
- A bug has been fixed in the tainting support, in which an `index()` operation on a tainted constant would cause all other constants to become tainted. [perl #64804]
- A regression has been fixed that was introduced in perl 5.12, whereby tainting errors were not correctly propagated through `die()`. [perl #111654]
- A regression has been fixed that was introduced in perl 5.14, in which `/[[:lower:]]/i` and `/[[:upper:]]/i` no longer matched the opposite case. [perl #101970]

## Acknowledgements

Perl 5.14.3 represents approximately 12 months of development since Perl 5.14.2 and contains approximately 2,300 lines of changes across 64 files from 22 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.14.3:

Abigail, Andy Dougherty, Carl Hayter, Chris 'BinGOs' Williams, Dave Rolsky, David Mitchell, Dominic Hargreaves, Father Chrysostomos, Florian Ragwitz, H.Merijn Brand, Jilles Tjoelker, Karl Williamson, Leon Timmermans, Michael G Schwern, Nicholas Clark, Niko Tyni, Pino Toscano,

Ricardo Signes, Salvador Fandiño, Samuel Thibault, Steve Hay, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5144delta − what is new for perl v5.14.4

## DESCRIPTION

This document describes differences between the 5.14.3 release and the 5.14.4 release.

If you are upgrading from an earlier release such as 5.12.0, first read perl5140delta, which describes differences between 5.12.0 and 5.14.0.

## Core Enhancements

No changes since 5.14.0.

## Security

This release contains one major, and medium, and a number of minor security fixes. The latter are included mainly to allow the test suite to pass cleanly with the clang compiler's address sanitizer facility.

### CVE−2013−1667: memory exhaustion with arbitrary hash keys

With a carefully crafted set of hash keys (for example arguments on a URL), it is possible to cause a hash to consume a large amount of memory and CPU, and thus possibly to achieve a Denial-of-Service.

This problem has been fixed.

### memory leak in Encode

The UTF−8 encoding implementation in Encode.xs had a memory leak which has been fixed.

### [perl #111594] Socket::unpack_sockaddr_un heap-buffer-overflow

A read buffer overflow could occur when copying `sockaddr` buffers. Fairly harmless.

This problem has been fixed.

### [perl #111586] SDBM_File: fix off-by-one access to global ".dir"

An extra byte was being copied for some string literals. Fairly harmless.

This problem has been fixed.

### off-by-two error in List::Util

A string literal was being used that included two bytes beyond the end of the string. Fairly harmless.

This problem has been fixed.

### [perl #115994] fix segv in regcomp.c:S_join_exact()

Under debugging builds, while marking optimised-out regex nodes as type `OPTIMIZED`, it could treat blocks of exact text as if they were nodes, and thus SEGV. Fairly harmless.

This problem has been fixed.

### [perl #115992] PL_eval_start use-after-free

The statement `local $[;`, when preceded by an `eval`, and when not part of an assignment, could crash. Fairly harmless.

This problem has been fixed.

### wrap-around with IO on long strings

Reading or writing strings greater than 2**31 bytes in size could segfault due to integer wraparound.

This problem has been fixed.

## Incompatible Changes

There are no changes intentionally incompatible with 5.14.0. If any exist, they are bugs and reports are welcome.

## Deprecations

There have been no deprecations since 5.14.0.

## Modules and Pragmata

### New Modules and Pragmata

None

### Updated Modules and Pragmata

The following modules have just the minor code fixes as listed above in ‘‘Security’’ (version numbers have not changed):

Socket
SDBM_File
List::Util

Encode has been upgraded from version 2.42_01 to version 2.42_02.

Module::CoreList has been updated to version 2.49_06 to add data for this release.

### Removed Modules and Pragmata

None.

## Documentation

### New Documentation

None.

### Changes to Existing Documentation

None.

## Diagnostics

No new or changed diagnostics.

## Utility Changes

None

## Configuration and Compilation

No changes.

## Platform Support

### New Platforms

None.

### Discontinued Platforms

None.

### Platform-Specific Notes

VMS

5.14.3 failed to compile on VMS due to incomplete application of a patch series that allowed `userelocatableinc` and `usesitecustomize` to be used simultaneously. Other platforms were not affected and the problem has now been corrected.

## Selected Bug Fixes

- In Perl 5.14.0, `$tainted ~~ @array` stopped working properly. Sometimes it would erroneously fail (when `$tainted` contained a string that occurs in the array *after* the first element) or erroneously succeed (when `undef` occurred after the first element) [perl #93590].

## Known Problems

None.

## Acknowledgements

Perl 5.14.4 represents approximately 5 months of development since Perl 5.14.3 and contains approximately 1,700 lines of changes across 49 files from 12 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.14.4:

Andy Dougherty, Chris ’BinGOs’ Williams, Christian Hansen, Craig A. Berry, Dave Rolsky, David Mitchell, Dominic Hargreaves, Father Chrysostomos, Florian Ragwitz, Reini Urban, Ricardo Signes, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5160delta − what is new for perl v5.16.0

## DESCRIPTION

This document describes differences between the 5.14.0 release and the 5.16.0 release.

If you are upgrading from an earlier release such as 5.12.0, first read perl5140delta, which describes differences between 5.12.0 and 5.14.0.

Some bug fixes in this release have been backported to later releases of 5.14.x. Those are indicated with the 5.14.x version in parentheses.

## Notice

With the release of Perl 5.16.0, the 5.12.x series of releases is now out of its support period. There may be future 5.12.x releases, but only in the event of a critical security issue. Users of Perl 5.12 or earlier should consider upgrading to a more recent release of Perl.

This policy is described in greater detail in perlpolicy.

## Core Enhancements

`use` *`VERSION`*

As of this release, version declarations like `use v5.16` now disable all features before enabling the new feature bundle. This means that the following holds true:

```
use 5.016;
# only 5.16 features enabled here
use 5.014;
# only 5.14 features enabled here (not 5.16)
```

`use v5.12` and higher continue to enable strict, but explicit `use strict` and `no strict` now override the version declaration, even when they come first:

```
no strict;
use 5.012;
# no strict here
```

There is a new ‘‘:default’’ feature bundle that represents the set of features enabled before any version declaration or `use feature` has been seen. Version declarations below 5.10 now enable the ‘‘:default’’ feature set. This does not actually change the behavior of `use v5.8`, because features added to the ‘‘:default’’ set are those that were traditionally enabled by default, before they could be turned off.

`no feature` now resets to the default feature set. To disable all features (which is likely to be a pretty special-purpose request, since it presumably won’t match any named set of semantics) you can now write `no feature ':all'`.

`$[` is now disabled under `use v5.16`. It is part of the default feature set and can be turned on or off explicitly with `use feature 'array_base'`.

`__SUB__`

The new `__SUB__` token, available under the `current_sub` feature (see feature) or `use v5.16`, returns a reference to the current subroutine, making it easier to write recursive closures.

### New and Improved Built-ins

*More consistent* `eval`

The `eval` operator sometimes treats a string argument as a sequence of characters and sometimes as a sequence of bytes, depending on the internal encoding. The internal encoding is not supposed to make any difference, but there is code that relies on this inconsistency.

The new `unicode_eval` and `evalbytes` features (enabled under `use 5.16.0`) resolve this. The `unicode_eval` feature causes `eval $string` to treat the string always as Unicode. The `evalbytes` features provides a function, itself called `evalbytes`, which evaluates its argument always as a string of bytes.

These features also fix oddities with source filters leaking to outer dynamic scopes.

See feature for more detail.

`substr` *lvalue revamp*

When `substr` is called in lvalue or potential lvalue context with two or three arguments, a special lvalue scalar is returned that modifies the original string (the first argument) when assigned to.

Previously, the offsets (the second and third arguments) passed to `substr` would be converted immediately to match the string, negative offsets being translated to positive and offsets beyond the end of the string being truncated.

Now, the offsets are recorded without modification in the special lvalue scalar that is returned, and the original string is not even looked at by `substr` itself, but only when the returned lvalue is read or modified.

These changes result in an incompatible change:

If the original string changes length after the call to `substr` but before assignment to its return value, negative offsets will remember their position from the end of the string, affecting code like this:

```
my $string = "string";
my $lvalue = \substr $string, -4, 2;
print $$lvalue, "\n"; # prints "ri"
$string = "bailing twine";
print $$lvalue, "\n"; # prints "wi"; used to print "il"
```

The same thing happens with an omitted third argument. The returned lvalue will always extend to the end of the string, even if the string becomes longer.

Since this change also allowed many bugs to be fixed (see "The `substr` operator"), and since the behavior of negative offsets has never been specified, the change was deemed acceptable.

*Return value of `tied`*

The value returned by `tied` on a tied variable is now the actual scalar that holds the object to which the variable is tied. This lets ties be weakened with `Scalar::Util::weaken(tied $tied_variable)`.

### Unicode Support

*Supports (almost) Unicode 6.1*

Besides the addition of whole new scripts, and new characters in existing scripts, this new version of Unicode, as always, makes some changes to existing characters. One change that may trip up some applications is that the General Category of two characters in the Latin-1 range, PILCROW SIGN and SECTION SIGN, has been changed from Other_Symbol to Other_Punctuation. The same change has been made for a character in each of Tibetan, Ethiopic, and Aegean. The code points U+3248..U+324F (CIRCLED NUMBER TEN ON BLACK SQUARE through CIRCLED NUMBER EIGHTY ON BLACK SQUARE) have had their General Category changed from Other_Symbol to Other_Numeric. The Line Break property has changes for Hebrew and Japanese; and because of other changes in 6.1, the Perl regular expression construct `\X` now works differently for some characters in Thai and Lao.

New aliases (synonyms) have been defined for many property values; these, along with the previously existing ones, are all cross-indexed in perluniprops.

The return value of `charnames::viacode()` is affected by other changes:

```
 Code point      Old Name             New Name
   U+000A    LINE FEED (LF)        LINE FEED
   U+000C    FORM FEED (FF)        FORM FEED
   U+000D    CARRIAGE RETURN (CR)  CARRIAGE RETURN
   U+0085    NEXT LINE (NEL)       NEXT LINE
   U+008E    SINGLE-SHIFT 2        SINGLE-SHIFT-2
   U+008F    SINGLE-SHIFT 3        SINGLE-SHIFT-3
   U+0091    PRIVATE USE 1         PRIVATE USE-1
   U+0092    PRIVATE USE 2         PRIVATE USE-2
   U+2118    SCRIPT CAPITAL P      WEIERSTRASS ELLIPTIC FUNCTION
```

Perl will accept any of these names as input, but `charnames::viacode()` now returns the new name of each pair. The change for U+2118 is considered by Unicode to be a correction, that is the original name was a mistake (but again, it will remain forever valid to use it to refer to U+2118). But most of these changes are the fallout of the mistake Unicode 6.0 made in naming a character used in Japanese cell phones to be ‘‘BELL’’, which conflicts with the longstanding industry use of (and

Unicode's recommendation to use) that name to mean the ASCII control character at U+0007. Therefore, that name has been deprecated in Perl since v5.14, and any use of it will raise a warning message (unless turned off). The name ''ALERT'' is now the preferred name for this code point, with ''BEL'' an acceptable short form. The name for the new cell phone character, at code point U+1F514, remains undefined in this version of Perl (hence we don't implement quite all of Unicode 6.1), but starting in v5.18, BELL will mean this character, and not U+0007.

Unicode has taken steps to make sure that this sort of mistake does not happen again. The Standard now includes all generally accepted names and abbreviations for control characters, whereas previously it didn't (though there were recommended names for most of them, which Perl used). This means that most of those recommended names are now officially in the Standard. Unicode did not recommend names for the four code points listed above between U+008E and U+008F, and in standardizing them Unicode subtly changed the names that Perl had previously given them, by replacing the final blank in each name by a hyphen. Unicode also officially accepts names that Perl had deprecated, such as FILE SEPARATOR. Now the only deprecated name is BELL. Finally, Perl now uses the new official names instead of the old (now considered obsolete) names for the first four code points in the list above (the ones which have the parentheses in them).

Now that the names have been placed in the Unicode standard, these kinds of changes should not happen again, though corrections, such as to U+2118, are still possible.

Unicode also added some name abbreviations, which Perl now accepts: SP for SPACE; TAB for CHARACTER TABULATION; NEW LINE, END OF LINE, NL, and EOL for LINE FEED; LOCKING-SHIFT ONE for SHIFT OUT; LOCKING-SHIFT ZERO for SHIFT IN; and ZWNBSP for ZERO WIDTH NO-BREAK SPACE.

More details on this version of Unicode are provided in <http://www.unicode.org/versions/Unicode6.1.0/>.

*`use charnames` is no longer needed for `\N{name}`*

When `\N{name}` is encountered, the `charnames` module is now automatically loaded when needed as if the `:full` and `:short` options had been specified. See charnames for more information.

*`\N{...}` can now have Unicode loose name matching*

This is described in the `charnames` item in ''Updated Modules and Pragmata'' below.

*Unicode Symbol Names*

Perl now has proper support for Unicode in symbol names. It used to be that `*{$foo}` would ignore the internal UTF8 flag and use the bytes of the underlying representation to look up the symbol. That meant that `*{"\x{100}"}` and `*{"\xc4\x80"}` would return the same thing. All these parts of Perl have been fixed to account for Unicode:

- Method names (including those passed to `use overload`)
- Typeglob names (including names of variables, subroutines, and filehandles)
- Package names
- `goto`
- Symbolic dereferencing
- Second argument to `bless()` and `tie()`
- Return value of `ref()`
- Subroutine prototypes
- Attributes
- Various warnings and error messages that mention variable names or values, methods, etc.

In addition, a parsing bug has been fixed that prevented `*{é}` from implicitly quoting the name, but instead interpreted it as `*{+é}`, which would cause a strict violation.

`*{"*a::b"}` automatically strips off the `*` if it is followed by an ASCII letter. That has been extended to all Unicode identifier characters.

One-character non-ASCII non-punctuation variables (like `$é`) are now subject to ''Used only once''

warnings. They used to be exempt, as they were treated as punctuation variables.

Also, single-character Unicode punctuation variables (like $X) are now supported [perl #69032].

*Improved ability to mix locales and Unicode, including UTF-8 locales*

An optional parameter has been added to `use locale`

```
use locale ':not_characters';
```

which tells Perl to use all but the `LC_CTYPE` and `LC_COLLATE` portions of the current locale. Instead, the character set is assumed to be Unicode. This lets locales and Unicode be seamlessly mixed, including the increasingly frequent UTF-8 locales. When using this hybrid form of locales, the `:locale` layer to the open pragma can be used to interface with the file system, and there are CPAN modules available for ARGV and environment variable conversions.

Full details are in perllocale.

*New function `fc` and corresponding escape sequence `\F` for Unicode foldcase*

Unicode foldcase is an extension to lowercase that gives better results when comparing two strings case-insensitively. It has long been used internally in regular expression `/i` matching. Now it is available explicitly through the new `fc` function call (enabled by `"use feature 'fc'"`, or `use v5.16`, or explicitly callable via `CORE::fc`) or through the new `\F` sequence in double-quotish strings.

Full details are in "fc" in perlfunc.

*The Unicode `Script_Extensions` property is now supported.*

New in Unicode 6.0, this is an improved `Script` property. Details are in "Scripts" in perlunicode.

### XS Changes

*Improved typemaps for Some Builtin Types*

Most XS authors will know there is a longstanding bug in the OUTPUT typemap for T_AVREF (`AV*`), T_HVREF (`HV*`), T_CVREF (`CV*`), and T_SVREF (`SVREF` or `\$foo`) that requires manually decrementing the reference count of the return value instead of the typemap taking care of this. For backwards-compatibility, this cannot be changed in the default typemaps. But we now provide additional typemaps `T_AVREF_REFCOUNT_FIXED`, etc. that do not exhibit this bug. Using them in your extension is as simple as having one line in your `TYPEMAP` section:

```
HV*    T_HVREF_REFCOUNT_FIXED
```

*`is_utf8_char()`*

The XS-callable function `is_utf8_char()`, when presented with malformed UTF-8 input, can read up to 12 bytes beyond the end of the string. This cannot be fixed without changing its API, and so its use is now deprecated. Use `is_utf8_char_buf()` (described just below) instead.

*Added `is_utf8_char_buf()`*

This function is designed to replace the deprecated "**is_utf8_char()**" function. It includes an extra parameter to make sure it doesn't read past the end of the input buffer.

*Other `is_utf8_foo()` functions, as well as `utf8_to_foo()`, etc.*

Most other XS-callable functions that take UTF-8 encoded input implicitly assume that the UTF-8 is valid (not malformed) with respect to buffer length. Do not do things such as change a character's case or see if it is alphanumeric without first being sure that it is valid UTF-8. This can be safely done for a whole string by using one of the functions `is_utf8_string()`, `is_utf8_string_loc()`, and `is_utf8_string_loclen()`.

*New Pad API*

Many new functions have been added to the API for manipulating lexical pads. See "Pad Data Structures" in perlapi for more information.

### Changes to Special Variables

*$$ can be assigned to*

$$ was made read-only in Perl 5.8.0. But only sometimes: `local $$` would make it writable again. Some CPAN modules were using `local $$` or XS code to bypass the read-only check, so there is no

reason to keep $$ read-only. (This change also allowed a bug to be fixed while maintaining backward compatibility.)

*$^X converted to an absolute path on FreeBSD, OS X and Solaris*

$^X is now converted to an absolute path on OS X, FreeBSD (without needing */proc* mounted) and Solaris 10 and 11. This augments the previous approach of using */proc* on Linux, FreeBSD, and NetBSD (in all cases, where mounted).

This makes relocatable perl installations more useful on these platforms. (See ''Relocatable `@INC`'' in *INSTALL*)

### Debugger Changes

*Features inside the debugger*

The current Perl's feature bundle is now enabled for commands entered in the interactive debugger.

*New option for the debugger's **t** command*

The **t** command in the debugger, which toggles tracing mode, now accepts a numeric argument that determines how many levels of subroutine calls to trace.

*`enable` and `disable`*

The debugger now has `disable` and `enable` commands for disabling existing breakpoints and re-enabling them. See perldebug.

*Breakpoints with file names*

The debugger's ''b'' command for setting breakpoints now lets a line number be prefixed with a file name. See ''b [file]:[line] [condition]'' in perldebug.

### The CORE Namespace

*The `CORE::` prefix*

The `CORE::` prefix can now be used on keywords enabled by feature.pm, even outside the scope of `use feature`.

*Subroutines in the `CORE` namespace*

Many Perl keywords are now available as subroutines in the CORE namespace. This lets them be aliased:

```
BEGIN { *entangle = \&CORE::tie }
entangle $variable, $package, @args;
```

And for prototypes to be bypassed:

```
sub mytie(\[%$*@]$@) {
    my ($ref, $pack, @args) = @_;
    ... do something ...
    goto &CORE::tie;
}
```

Some of these cannot be called through references or via `&foo` syntax, but must be called as barewords.

See CORE for details.

### Other Changes

*Anonymous handles*

Automatically generated file handles are now named __ANONIO__ when the variable name cannot be determined, rather than $__ANONIO__.

*Autoloaded sort Subroutines*

Custom sort subroutines can now be autoloaded [perl #30661]:

```
sub AUTOLOAD { ... }
@sorted = sort foo @list; # uses AUTOLOAD
```

*`continue` no longer requires the "switch" feature*

The `continue` keyword has two meanings. It can introduce a `continue` block after a loop, or it

can exit the current `when` block. Up to now, the latter meaning was valid only with the ''switch'' feature enabled, and was a syntax error otherwise. Since the main purpose of feature.pm is to avoid conflicts with user-defined subroutines, there is no reason for `continue` to depend on it.

*DTrace probes for interpreter phase change*

The `phase-change` probes will fire when the interpreter's phase changes, which tracks the `${^GLOBAL_PHASE}` variable. `arg0` is the new phase name; `arg1` is the old one. This is useful for limiting your instrumentation to one or more of: compile time, run time, or destruct time.

*`__FILE__()` Syntax*

The `__FILE__`, `__LINE__` and `__PACKAGE__` tokens can now be written with an empty pair of parentheses after them. This makes them parse the same way as `time`, `fork` and other built-in functions.

*The `\$` prototype accepts any scalar lvalue*

The `\$` and `\[$]` subroutine prototypes now accept any scalar lvalue argument. Previously they accepted only scalars beginning with `$` and hash and array elements. This change makes them consistent with the way the built-in `read` and `recv` functions (among others) parse their arguments. This means that one can override the built-in functions with custom subroutines that parse their arguments the same way.

*_ in subroutine prototypes*

The _ character in subroutine prototypes is now allowed before `@` or `%`.

## Security

### Use `is_utf8_char_buf()` and not `is_utf8_char()`

The latter function is now deprecated because its API is insufficient to guarantee that it doesn't read (up to 12 bytes in the worst case) beyond the end of its input string. See **is_utf8_char_buf()**.

### Malformed UTF-8 input could cause attempts to read beyond the end of the buffer

Two new XS-accessible functions, `utf8_to_uvchr_buf()` and `utf8_to_uvuni_buf()` are now available to prevent this, and the Perl core has been converted to use them. See ''Internal Changes''.

### `File::Glob::bsd_glob()` memory error with GLOB_ALTDIRFUNC (CVE-2011-2728).

Calling `File::Glob::bsd_glob` with the unsupported flag GLOB_ALTDIRFUNC would cause an access violation / segfault. A Perl program that accepts a flags value from an external source could expose itself to denial of service or arbitrary code execution attacks. There are no known exploits in the wild. The problem has been corrected by explicitly disabling all unsupported flags and setting unused function pointers to null. Bug reported by Clément Lecigne. (5.14.2)

### Privileges are now set correctly when assigning to `$(`

A hypothetical bug (probably unexploitable in practice) because the incorrect setting of the effective group ID while setting `$(` has been fixed. The bug would have affected only systems that have `setresgid()` but not `setregid()`, but no such systems are known to exist.

## Deprecations

### Don't read the Unicode data base files in *lib/unicore*

It is now deprecated to directly read the Unicode data base files. These are stored in the *lib/unicore* directory. Instead, you should use the new functions in Unicode::UCD. These provide a stable API, and give complete information.

Perl may at some point in the future change or remove these files. The file which applications were most likely to have used is *lib/unicore/ToDigit.pl*. ''**prop_invmap()**'' in Unicode::UCD can be used to get at its data instead.

### XS functions `is_utf8_char()`, `utf8_to_uvchr()` and `utf8_to_uvuni()`

This function is deprecated because it could read beyond the end of the input string. Use the new **is_utf8_char_buf()**, `utf8_to_uvchr_buf()` and `utf8_to_uvuni_buf()` instead.

## Future Deprecations

This section serves as a notice of features that are *likely* to be removed or deprecated in the next release of perl (5.18.0). If your code depends on these features, you should contact the Perl 5 Porters via the mailing list <http://lists.perl.org/list/perl5-porters.html> or perlbug to explain your use case and inform

the deprecation process.

### Core Modules

These modules may be marked as deprecated *from the core*. This only means that they will no longer be installed by default with the core distribution, but will remain available on the CPAN.

- CPANPLUS
- Filter::Simple
- PerlIO::mmap
- Pod::LaTeX
- Pod::Parser
- SelfLoader
- Text::Soundex
- Thread.pm

### Platforms with no supporting programmers

These platforms will probably have their special build support removed during the 5.17.0 development series.

- BeOS
- djgpp
- dgux
- EPOC
- MPE/iX
- Rhapsody
- UTS
- VM/ESA

### Other Future Deprecations

- Swapping of $< and $>

  For more information about this future deprecation, see the relevant RT ticket <https://github.com/Perl/perl5/issues/11547>.

- sfio, stdio

  Perl supports being built without PerlIO proper, using a stdio or sfio wrapper instead. A perl build like this will not support IO layers and thus Unicode IO, making it rather handicapped.

  PerlIO supports a `stdio` layer if stdio use is desired, and similarly a sfio layer could be produced.

- Unescaped literal `"{"` in regular expressions.

  Starting with v5.20, it is planned to require a literal `"{"` to be escaped, for example by preceding it with a backslash. In v5.18, a deprecated warning message will be emitted for all such uses. This affects only patterns that are to match a literal `"{"`. Other uses of this character, such as part of a quantifier or sequence as in those below, are completely unaffected:

  ```
  /foo{3,5}/
  /\p{Alphabetic}/
  /\N{DIGIT ZERO}
  ```

  Removing this will permit extensions to Perl's pattern syntax and better error checking for existing syntax. See ''Quantifiers'' in perlre for an example.

- Revamping `"\Q"` semantics in double-quotish strings when combined with other escapes.

  There are several bugs and inconsistencies involving combinations of `\Q` and escapes like `\x`, `\L`, etc., within a `\Q...\E` pair. These need to be fixed, and doing so will necessarily change current behavior. The changes have not yet been settled.

## Incompatible Changes

### Special blocks called in void context

Special blocks (`BEGIN`, `CHECK`, `INIT`, `UNITCHECK`, `END`) are now called in void context. This avoids wasteful copying of the result of the last statement [perl #108794].

### The `overloading` pragma and regexp objects

With `no overloading`, regular expression objects returned by `qr//` are now stringified as ''Regexp=REGEXP(0xbe600d)'' instead of the regular expression itself [perl #108780].

### Two XS typemap Entries removed

Two presumably unused XS typemap entries have been removed from the core typemap: T_DATAUNIT and T_CALLBACK. If you are, against all odds, a user of these, please see the instructions on how to restore them in perlxstypemap.

### Unicode 6.1 has incompatibilities with Unicode 6.0

These are detailed in ''Supports (almost) Unicode 6.1'' above. You can compile this version of Perl to use Unicode 6.0. See ''Hacking Perl to work on earlier Unicode versions (for very serious hackers only)'' in perlunicode.

### Borland compiler

All support for the Borland compiler has been dropped. The code had not worked for a long time anyway.

### Certain deprecated Unicode properties are no longer supported by default

Perl should never have exposed certain Unicode properties that are used by Unicode internally and not meant to be publicly available. Use of these has generated deprecated warning messages since Perl 5.12. The removed properties are Other_Alphabetic, Other_Default_Ignorable_Code_Point, Other_Grapheme_Extend, Other_ID_Continue, Other_ID_Start, Other_Lowercase, Other_Math, and Other_Uppercase.

Perl may be recompiled to include any or all of them; instructions are given in ''Unicode character properties that are NOT accepted by Perl'' in perluniprops.

### Dereferencing IO thingies as typeglobs

The `*{...}` operator, when passed a reference to an IO thingy (as in `*{*STDIN{IO}}`), creates a new typeglob containing just that IO object. Previously, it would stringify as an empty string, but some operators would treat it as undefined, producing an ''uninitialized'' warning. Now it stringifies as __ANONIO__ [perl #96326].

### User-defined case-changing operations

This feature was deprecated in Perl 5.14, and has now been removed. The CPAN module Unicode::Casing provides better functionality without the drawbacks that this feature had, as are detailed in the 5.14 documentation: <http://perldoc.perl.org/5.14.0/perlunicode.html#User−Defined−Case−Mappings−%28for−serious−hackers−only%29>

### XSUBs are now 'static'

XSUB C functions are now 'static', that is, they are not visible from outside the compilation unit. Users can use the new `XS_EXTERNAL(name)` and `XS_INTERNAL(name)` macros to pick the desired linking behavior. The ordinary `XS(name)` declaration for XSUBs will continue to declare non−'static' XSUBs for compatibility, but the XS compiler, ExtUtils::ParseXS (`xsubpp`) will emit 'static' XSUBs by default. ExtUtils::ParseXS's behavior can be reconfigured from XS using the `EXPORT_XSUB_SYMBOLS` keyword. See perlxs for details.

### Weakening read-only references

Weakening read-only references is no longer permitted. It should never have worked anyway, and could sometimes result in crashes.

### Tying scalars that hold typeglobs

Attempting to tie a scalar after a typeglob was assigned to it would instead tie the handle in the typeglob's IO slot. This meant that it was impossible to tie the scalar itself. Similar problems affected `tied` and `untie`: `tied $scalar` would return false on a tied scalar if the last thing returned was a typeglob, and `untie $scalar` on such a tied scalar would do nothing.

We fixed this problem before Perl 5.14.0, but it caused problems with some CPAN modules, so we put in a deprecation cycle instead.

Now the deprecation has been removed and this bug has been fixed. So `tie $scalar` will always tie the scalar, not the handle it holds. To tie the handle, use `tie *$scalar` (with an explicit asterisk). The same applies to `tied *$scalar` and `untie *$scalar`.

**IPC::Open3 no longer provides `xfork()`, `xclose_on_exec()` and `xpipe_anon()`**

All three functions were private, undocumented, and unexported. They do not appear to be used by any code on CPAN. Two have been inlined and one deleted entirely.

**`$$` no longer caches PID**

Previously, if one called **fork**(3) from C, Perl's notion of `$$` could go out of sync with what **getpid()** returns. By always fetching the value of `$$` via **getpid()**, this potential bug is eliminated. Code that depends on the caching behavior will break. As described in Core Enhancements, `$$` is now writable, but it will be reset during a fork.

**`$$` and `getppid()` no longer emulate POSIX semantics under LinuxThreads**

The POSIX emulation of `$$` and `getppid()` under the obsolete LinuxThreads implementation has been removed. This only impacts users of Linux 2.4 and users of Debian GNU/kFreeBSD up to and including 6.0, not the vast majority of Linux installations that use NPTL threads.

This means that `getppid()`, like `$$`, is now always guaranteed to return the OS's idea of the current state of the process, not perl's cached version of it.

See the documentation for $$ for details.

**`$<`, `$>`, `$(` and `$)` are no longer cached**

Similarly to the changes to `$$` and `getppid()`, the internal caching of `$<`, `$>`, `$(` and `$)` has been removed.

When we cached these values our idea of what they were would drift out of sync with reality if someone (e.g., someone embedding perl) called `sete?[ug]id()` without updating `PL_e?[ug]id`. Having to deal with this complexity wasn't worth it given how cheap the `gete?[ug]id()` system call is.

This change will break a handful of CPAN modules that use the XS-level `PL_uid`, `PL_gid`, `PL_euid` or `PL_egid` variables.

The fix for those breakages is to use `PerlProc_gete?[ug]id()` to retrieve them (e.g., `PerlProc_getuid()`), and not to assign to `PL_e?[ug]id` if you change the UID/GID/EUID/EGID. There is no longer any need to do so since perl will always retrieve the up-to-date version of those values from the OS.

**Which Non-ASCII characters get quoted by `quotemeta` and `\Q` has changed**

This is unlikely to result in a real problem, as Perl does not attach special meaning to any non-ASCII character, so it is currently irrelevant which are quoted or not. This change fixes bug [perl #77654] and brings Perl's behavior more into line with Unicode's recommendations. See ''quotemeta'' in perlfunc.

## Performance Enhancements

- Improved performance for Unicode properties in regular expressions

  Matching a code point against a Unicode property is now done via a binary search instead of linear. This means for example that the worst case for a 1000 item property is 10 probes instead of 1000. This inefficiency has been compensated for in the past by permanently storing in a hash the results of a given probe plus the results for the adjacent 64 code points, under the theory that nearby code points are likely to be searched for. A separate hash was used for each mention of a Unicode property in each regular expression. Thus, `qr/\p{foo}abc\p{foo}/` would generate two hashes. Any probes in one instance would be unknown to the other, and the hashes could expand separately to be quite large if the regular expression were used on many different widely-separated code points. Now, however, there is just one hash shared by all instances of a given property. This means that if `\p{foo}` is matched against ''A'' in one regular expression in a thread, the result will be known immediately to all regular expressions, and the relentless march of using up memory is slowed considerably.

- Version declarations with the `use` keyword (e.g., `use 5.012`) are now faster, as they enable features without loading *feature.pm*.

- `local $_` is faster now, as it no longer iterates through magic that it is not going to copy anyway.
- Perl 5.12.0 sped up the destruction of objects whose classes define empty `DESTROY` methods (to prevent autoloading), by simply not calling such empty methods. This release takes this optimization a step further, by not calling any `DESTROY` method that begins with a `return` statement. This can be useful for destructors that are only used for debugging:

  ```
  use constant DEBUG => 1;
  sub DESTROY { return unless DEBUG; ... }
  ```

  Constant-folding will reduce the first statement to `return;` if DEBUG is set to 0, triggering this optimization.
- Assigning to a variable that holds a typeglob or copy-on-write scalar is now much faster. Previously the typeglob would be stringified or the copy-on-write scalar would be copied before being clobbered.
- Assignment to `substr` in void context is now more than twice its previous speed. Instead of creating and returning a special lvalue scalar that is then assigned to, `substr` modifies the original string itself.
- `substr` no longer calculates a value to return when called in void context.
- Due to changes in File::Glob, Perl's `glob` function and its `<...>` equivalent are now much faster. The splitting of the pattern into words has been rewritten in C, resulting in speed-ups of 20% for some cases.

  This does not affect `glob` on VMS, as it does not use File::Glob.
- The short-circuiting operators `&&`, `||`, and `//`, when chained (such as `$a || $b || $c`), are now considerably faster to short-circuit, due to reduced optree traversal.
- The implementation of `s///r` makes one fewer copy of the scalar's value.
- Recursive calls to lvalue subroutines in lvalue scalar context use less memory.

## Modules and Pragmata

### Deprecated Modules

Version::Requirements

Version::Requirements is now DEPRECATED, use CPAN::Meta::Requirements, which is a drop-in replacement. It will be deleted from perl.git blead in v5.17.0.

### New Modules and Pragmata

- arybase — this new module implements the `$[` variable.
- PerlIO::mmap 0.010 has been added to the Perl core.

  The `mmap` PerlIO layer is no longer implemented by perl itself, but has been moved out into the new PerlIO::mmap module.

### Updated Modules and Pragmata

This is only an overview of selected module updates. For a complete list of updates, run:

```
$ corelist --diff 5.14.0 5.16.0
```

You can substitute your favorite version in place of 5.14.0, too.

- Archive::Extract has been upgraded from version 0.48 to 0.58.

  Includes a fix for FreeBSD to only use `unzip` if it is located in `/usr/local/bin`, as FreeBSD 9.0 will ship with a limited `unzip` in `/usr/bin`.
- Archive::Tar has been upgraded from version 1.76 to 1.82.

  Adjustments to handle files >8gb (>0777777777777 octal) and a feature to return the MD5SUM of files in the archive.
- base has been upgraded from version 2.16 to 2.18.

  `base` no longer sets a module's `$VERSION` to "-1" when a module it loads does not define a `$VERSION`. This change has been made because "-1" is not a valid version number under the

new ‘‘lax’’ criteria used internally by `UNIVERSAL::VERSION`. (See version for more on ‘‘lax’’ version criteria.)

`base` no longer internally skips loading modules it has already loaded and instead relies on `require` to inspect `%INC`. This fixes a bug when `base` is used with code that clear `%INC` to force a module to be reloaded.

- Carp has been upgraded from version 1.20 to 1.26.

  It now includes last read filehandle info and puts a dot after the file and line number, just like errors from `die` [perl #106538].

- charnames has been updated from version 1.18 to 1.30.

  `charnames` can now be invoked with a new option, `:loose`, which is like the existing `:full` option, but enables Unicode loose name matching. Details are in ‘‘LOOSE MATCHES’’ in charnames.

- B::Deparse has been upgraded from version 1.03 to 1.14. This fixes numerous deparsing bugs.

- CGI has been upgraded from version 3.52 to 3.59.

  It uses the public and documented FCGI.pm API in CGI::Fast. CGI::Fast was using an FCGI API that was deprecated and removed from documentation more than ten years ago. Usage of this deprecated API with FCGI >= 0.70 or FCGI <= 0.73 introduces a security issue. <https://rt.cpan.org/Public/Bug/Display.html?id=68380>
  <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE−2011−2766>

  Things that may break your code:

  `url()` was fixed to return `PATH_INFO` when it is explicitly requested with either the `path=>1` or `path_info=>1` flag.

  If your code is running under mod_rewrite (or compatible) and you are calling `self_url()` or you are calling `url()` and passing `path_info=>1`, these methods will actually be returning `PATH_INFO` now, as you have explicitly requested or `self_url()` has requested on your behalf.

  The `PATH_INFO` has been omitted in such URLs since the issue was introduced in the 3.12 release in December, 2005.

  This bug is so old your application may have come to depend on it or workaround it. Check for application before upgrading to this release.

  Examples of affected method calls:

  ```
  $q−>url(−absolute => 1, −query => 1, −path_info => 1);
  $q−>url(−path=>1);
  $q−>url(−full=>1,−path=>1);
  $q−>url(−rewrite=>1,−path=>1);
  $q−>self_url();
  ```

  We no longer read from STDIN when the Content-Length is not set, preventing requests with no Content-Length from sometimes freezing. This is consistent with the CGI RFC 3875, and is also consistent with CGI::Simple. However, the old behavior may have been expected by some command-line uses of CGI.pm.

  In addition, the DELETE HTTP verb is now supported.

- Compress::Zlib has been upgraded from version 2.035 to 2.048.

  IO::Compress::Zip and IO::Uncompress::Unzip now have support for LZMA (method 14). There is a fix for a CRC issue in IO::Compress::Unzip and it supports Streamed Stored context now. And fixed a Zip64 issue in IO::Compress::Zip when the content size was exactly 0xFFFFFFFF.

- Digest::SHA has been upgraded from version 5.61 to 5.71.

  Added BITS mode to the addfile method and shasum. This makes partial-byte inputs possible via files/STDIN and lets shasum check all 8074 NIST Msg vectors, where previously special programming was required to do this.

- Encode has been upgraded from version 2.42 to 2.44.

  Missing aliases added, a deep recursion error fixed and various documentation updates.

  Addressed 'decode_xs n−byte heap−overflow' security bug in Unicode.xs (CVE−2011−2939). (5.14.2)

- ExtUtils::CBuilder updated from version 0.280203 to 0.280206.

  The new version appends CFLAGS and LDFLAGS to their Config.pm counterparts.

- ExtUtils::ParseXS has been upgraded from version 2.2210 to 3.16.

  Much of ExtUtils::ParseXS, the module behind the XS compiler `xsubpp`, was rewritten and cleaned up. It has been made somewhat more extensible and now finally uses strictures.

  The typemap logic has been moved into a separate module, ExtUtils::Typemaps. See ''New Modules and Pragmata'', above.

  For a complete set of changes, please see the ExtUtils::ParseXS changelog, available on the CPAN.

- File::Glob has been upgraded from version 1.12 to 1.17.

  On Windows, tilde (~) expansion now checks the `USERPROFILE` environment variable, after checking `HOME`.

  It has a new `:bsd_glob` export tag, intended to replace `:glob`. Like `:glob` it overrides `glob` with a function that does not split the glob pattern into words, but, unlike `:glob`, it iterates properly in scalar context, instead of returning the last file.

  There are other changes affecting Perl's own `glob` operator (which uses File::Glob internally, except on VMS). See ''Performance Enhancements'' and ''Selected Bug Fixes''.

- FindBin updated from version 1.50 to 1.51.

  It no longer returns a wrong result if a script of the same name as the current one exists in the path and is executable.

- HTTP::Tiny has been upgraded from version 0.012 to 0.017.

  Added support for using `$ENV{http_proxy}` to set the default proxy host.

  Adds additional shorthand methods for all common HTTP verbs, a `post_form()` method for POST-ing x−www-form-urlencoded data and a `www_form_urlencode()` utility method.

- IO has been upgraded from version 1.25_04 to 1.25_06, and IO::Handle from version 1.31 to 1.33.

  Together, these upgrades fix a problem with IO::Handle's `getline` and `getlines` methods. When these methods are called on the special ARGV handle, the next file is automatically opened, as happens with the built-in `<>` and `readline` functions. But, unlike the built-ins, these methods were not respecting the caller's use of the open pragma and applying the appropriate I/O layers to the newly-opened file [rt.cpan.org #66474].

- IPC::Cmd has been upgraded from version 0.70 to 0.76.

  Capturing of command output (both `STDOUT` and `STDERR`) is now supported using IPC::Open3 on MSWin32 without requiring IPC::Run.

- IPC::Open3 has been upgraded from version 1.09 to 1.12.

  Fixes a bug which prevented use of `open3` on Windows when `*STDIN`, `*STDOUT` or `*STDERR` had been localized.

  Fixes a bug which prevented duplicating numeric file descriptors on Windows.

  `open3` with ''−'' for the program name works once more. This was broken in version 1.06 (and hence in Perl 5.14.0) [perl #95748].

- Locale::Codes has been upgraded from version 3.16 to 3.21.

  Added Language Extension codes (langext) and Language Variation codes (langvar) as defined in the IANA language registry.

  Added language codes from ISO 639−5

Added language/script codes from the IANA language subtag registry

Fixed an uninitialized value warning [rt.cpan.org #67438].

Fixed the return value for the all_XXX_codes and all_XXX_names functions [rt.cpan.org #69100].

Reorganized modules to move Locale::MODULE to Locale::Codes::MODULE to allow for cleaner future additions. The original four modules (Locale::Language, Locale::Currency, Locale::Country, Locale::Script) will continue to work, but all new sets of codes will be added in the Locale::Codes namespace.

The code2XXX, XXX2code, all_XXX_codes, and all_XXX_names functions now support retired codes. All codesets may be specified by a constant or by their name now. Previously, they were specified only by a constant.

The alias_code function exists for backward compatibility. It has been replaced by rename_country_code. The alias_code function will be removed some time after September, 2013.

All work is now done in the central module (Locale::Codes). Previously, some was still done in the wrapper modules (Locale::Codes::*). Added Language Family codes (langfam) as defined in ISO 639-5.

- Math::BigFloat has been upgraded from version 1.993 to 1.997.

  The `numify` method has been corrected to return a normalized Perl number (the result of `0 + $thing`), instead of a string [rt.cpan.org #66732].

- Math::BigInt has been upgraded from version 1.994 to 1.998.

  It provides a new `bsgn` method that complements the `babs` method.

  It fixes the internal `objectify` function's handling of ''foreign objects'' so they are converted to the appropriate class (Math::BigInt or Math::BigFloat).

- Math::BigRat has been upgraded from version 0.2602 to 0.2603.

  `int()` on a Math::BigRat object containing -1/2 now creates a Math::BigInt containing 0, rather than -0. Math::BigInt does not even support negative zero, so the resulting object was actually malformed [perl #95530].

- Math::Complex has been upgraded from version 1.56 to 1.59 and Math::Trig from version 1.2 to 1.22.

  Fixes include: correct copy constructor usage; fix polarwise formatting with numeric format specifier; and more stable `great_circle_direction` algorithm.

- Module::CoreList has been upgraded from version 2.51 to 2.66.

  The `corelist` utility now understands the `-r` option for displaying Perl release dates and the `--diff` option to print the set of modlib changes between two perl distributions.

- Module::Metadata has been upgraded from version 1.000004 to 1.000009.

  Adds `provides` method to generate a CPAN META provides data structure correctly; use of `package_versions_from_directory` is discouraged.

- ODBM_File has been upgraded from version 1.10 to 1.12.

  The XS code is now compiled with `PERL_NO_GET_CONTEXT`, which will aid performance under ithreads.

- open has been upgraded from version 1.08 to 1.10.

  It no longer turns off layers on standard handles when invoked without the ''':std''' directive. Similarly, when invoked *with* the ''':std''' directive, it now clears layers on STDERR before applying the new ones, and not just on STDIN and STDOUT [perl #92728].

- overload has been upgraded from version 1.13 to 1.18.

  `overload::Overloaded` no longer calls `can` on the class, but uses another means to

determine whether the object has overloading. It was never correct for it to call `can`, as overloading does not respect AUTOLOAD. So classes that autoload methods and implement `can` no longer have to account for overloading [perl #40333].

A warning is now produced for invalid arguments. See ''New Diagnostics''.

- PerlIO::scalar has been upgraded from version 0.11 to 0.14.

  (This is the module that implements `open $fh, '>', \$scalar`.)

  It fixes a problem with `open my $fh, ">", \$scalar` not working if `$scalar` is a copy-on-write scalar. (5.14.2)

  It also fixes a hang that occurs with `readline` or `<$fh>` if a typeglob has been assigned to `$scalar` [perl #92258].

  It no longer assumes during `seek` that `$scalar` is a string internally. If it didn't crash, it was close to doing so [perl #92706]. Also, the internal print routine no longer assumes that the position set by `seek` is valid, but extends the string to that position, filling the intervening bytes (between the old length and the seek position) with nulls [perl #78980].

  Printing to an in-memory handle now works if the `$scalar` holds a reference, stringifying the reference before modifying it. References used to be treated as empty strings.

  Printing to an in-memory handle no longer crashes if the `$scalar` happens to hold a number internally, but no string buffer.

  Printing to an in-memory handle no longer creates scalars that confuse the regular expression engine [perl #108398].

- Pod::Functions has been upgraded from version 1.04 to 1.05.

  *Functions.pm* is now generated at perl build time from annotations in *perlfunc.pod*. This will ensure that Pod::Functions and perlfunc remain in synchronisation.

- Pod::Html has been upgraded from version 1.11 to 1.1502.

  This is an extensive rewrite of Pod::Html to use Pod::Simple under the hood. The output has changed significantly.

- Pod::Perldoc has been upgraded from version 3.15_03 to 3.17.

  It corrects the search paths on VMS [perl #90640]. (5.14.1)

  The **-v** option now fetches the right section for `$0`.

  This upgrade has numerous significant fixes. Consult its changelog on the CPAN for more information.

- POSIX has been upgraded from version 1.24 to 1.30.

  POSIX no longer uses AutoLoader. Any code which was relying on this implementation detail was buggy, and may fail because of this change. The module's Perl code has been considerably simplified, roughly halving the number of lines, with no change in functionality. The XS code has been refactored to reduce the size of the shared object by about 12%, with no change in functionality. More POSIX functions now have tests.

  `sigsuspend` and `pause` now run signal handlers before returning, as the whole point of these two functions is to wait until a signal has arrived, and then return *after* it has been triggered. Delayed, or ''safe'', signals were preventing that from happening, possibly resulting in race conditions [perl #107216].

  `POSIX::sleep` is now a direct call into the underlying OS `sleep` function, instead of being a Perl wrapper on `CORE::sleep`. `POSIX::dup2` now returns the correct value on Win32 (*i.e.*, the file descriptor). `POSIX::SigSet sigsuspend` and `sigpending` and `POSIX::pause` now dispatch safe signals immediately before returning to their caller.

  `POSIX::Termios::setattr` now defaults the third argument to `TCSANOW`, instead of 0. On most platforms `TCSANOW` is defined to be 0, but on some 0 is not a valid parameter, which caused a call with defaults to fail.

- Socket has been upgraded from version 1.94 to 2.001.

  It has new functions and constants for handling IPv6 sockets:

  ```
  pack_ipv6_mreq
  unpack_ipv6_mreq
  IPV6_ADD_MEMBERSHIP
  IPV6_DROP_MEMBERSHIP
  IPV6_MTU
  IPV6_MTU_DISCOVER
  IPV6_MULTICAST_HOPS
  IPV6_MULTICAST_IF
  IPV6_MULTICAST_LOOP
  IPV6_UNICAST_HOPS
  IPV6_V6ONLY
  ```

- Storable has been upgraded from version 2.27 to 2.34.

  It no longer turns copy-on-write scalars into read-only scalars when freezing and thawing.

- Sys::Syslog has been upgraded from version 0.27 to 0.29.

  This upgrade closes many outstanding bugs.

- Term::ANSIColor has been upgraded from version 3.00 to 3.01.

  Only interpret an initial array reference as a list of colors, not any initial reference, allowing the colored function to work properly on objects with stringification defined.

- Term::ReadLine has been upgraded from version 1.07 to 1.09.

  Term::ReadLine now supports any event loop, including unpublished ones and simple IO::Select, loops without the need to rewrite existing code for any particular framework [perl #108470].

- threads::shared has been upgraded from version 1.37 to 1.40.

  Destructors on shared objects used to be ignored sometimes if the objects were referenced only by shared data structures. This has been mostly fixed, but destructors may still be ignored if the objects still exist at global destruction time [perl #98204].

- Unicode::Collate has been upgraded from version 0.73 to 0.89.

  Updated to CLDR 1.9.1

  Locales updated to CLDR 2.0: mk, mt, nb, nn, ro, ru, sk, sr, sv, uk, zh__pinyin, zh__stroke

  Newly supported locales: bn, fa, ml, mr, or, pa, sa, si, si__dictionary, sr_Latn, sv__reformed, ta, te, th, ur, wae.

  Tailored compatibility ideographs as well as unified ideographs for the locales: ja, ko, zh__big5han, zh__gb2312han, zh__pinyin, zh__stroke.

  Locale/*.pl files are now searched for in `@INC`.

- Unicode::Normalize has been upgraded from version 1.10 to 1.14.

  Fixes for the removal of *unicore/CompositionExclusions.txt* from core.

- Unicode::UCD has been upgraded from version 0.32 to 0.43.

  This adds four new functions: `prop_aliases()` and `prop_value_aliases()`, which are used to find all Unicode-approved synonyms for property names, or to convert from one name to another; `prop_invlist` which returns all code points matching a given Unicode binary property; and `prop_invmap` which returns the complete specification of a given Unicode property.

- Win32API::File has been upgraded from version 0.1101 to 0.1200.

  Added SetStdHandle and GetStdHandle functions

### Removed Modules and Pragmata

As promised in Perl 5.14.0's release notes, the following modules have been removed from the core distribution, and if needed should be installed from CPAN instead.

- Devel::DProf has been removed from the Perl core. Prior version was 20110228.00.
- Shell has been removed from the Perl core. Prior version was 0.72_01.
- Several old perl4−style libraries which have been deprecated with 5.14 are now removed:

```
abbrev.pl assert.pl bigfloat.pl bigint.pl bigrat.pl cacheout.pl
complete.pl ctime.pl dotsh.pl exceptions.pl fastcwd.pl flush.pl
getcwd.pl getopt.pl getopts.pl hostname.pl importenv.pl
lib/find{,depth}.pl look.pl newgetopt.pl open2.pl open3.pl
pwd.pl shellwords.pl stat.pl tainted.pl termcap.pl timelocal.pl
```

  They can be found on CPAN as Perl4::CoreLibs.

## Documentation

### New Documentation

*perldtrace*

perldtrace describes Perl's DTrace support, listing the provided probes and gives examples of their use.

*perlexperiment*

This document is intended to provide a list of experimental features in Perl. It is still a work in progress.

*perlootut*

This a new OO tutorial. It focuses on basic OO concepts, and then recommends that readers choose an OO framework from CPAN.

*perlxstypemap*

The new manual describes the XS typemapping mechanism in unprecedented detail and combines new documentation with information extracted from perlxs and the previously unofficial list of all core typemaps.

### Changes to Existing Documentation

*perlapi*

- The HV API has long accepted negative lengths to show that the key is in UTF8. This is now documented.
- The `boolSV()` macro is now documented.

*perlfunc*

- `dbmopen` treats a 0 mode as a special case, that prevents a nonexistent file from being created. This has been the case since Perl 5.000, but was never documented anywhere. Now the perlfunc entry mentions it [perl #90064].
- As an accident of history, `open $fh, '<:', ...` applies the default layers for the platform (`:raw` on Unix, `:crlf` on Windows), ignoring whatever is declared by open.pm. This seems such a useful feature it has been documented in perlfunc and open.
- The entry for `split` has been rewritten. It is now far clearer than before.

*perlguts*

- A new section, Autoloading with XSUBs, has been added, which explains the two APIs for accessing the name of the autoloaded sub.
- Some function descriptions in perlguts were confusing, as it was not clear whether they referred to the function above or below the description. This has been clarified [perl #91790].

*perlobj*

- This document has been rewritten from scratch, and its coverage of various OO concepts has been expanded.

*perlop*

- Documentation of the smartmatch operator has been reworked and moved from perlsyn to perlop where it belongs.

  It has also been corrected for the case of `undef` on the left-hand side. The list of different smart match behaviors had an item in the wrong place.

- Documentation of the ellipsis statement (`...`) has been reworked and moved from perlop to perlsyn.

- The explanation of bitwise operators has been expanded to explain how they work on Unicode strings (5.14.1).

- More examples for `m//g` have been added (5.14.1).

- The `<<\FOO` here-doc syntax has been documented (5.14.1).

*perlpragma*

- There is now a standard convention for naming keys in the `%^H`, documented under Key naming.

*"Laundering and Detecting Tainted Data" in perlsec*

- The example function for checking for taintedness contained a subtle error. `$@` needs to be localized to prevent its changing this global's value outside the function. The preferred method to check for this remains "tainted" in Scalar::Util.

*perllol*

- perllol has been expanded with examples using the new `push $scalar` syntax introduced in Perl 5.14.0 (5.14.1).

*perlmod*

- perlmod now states explicitly that some types of explicit symbol table manipulation are not supported. This codifies what was effectively already the case [perl #78074].

*perlpodstyle*

- The tips on which formatting codes to use have been corrected and greatly expanded.

- There are now a couple of example one-liners for previewing POD files after they have been edited.

*perlre*

- The `(*COMMIT)` directive is now listed in the right section (Verbs without an argument).

*perlrun*

- perlrun has undergone a significant clean-up. Most notably, the **−0x...** form of the **−0** flag has been clarified, and the final section on environment variables has been corrected and expanded (5.14.1).

*perlsub*

- The ($;) prototype syntax, which has existed for rather a long time, is now documented in perlsub. It lets a unary function have the same precedence as a list operator.

*perltie*

- The required syntax for tying handles has been documented.

*perlvar*

- The documentation for `$!` has been corrected and clarified. It used to state that `$!` could be `undef`, which is not the case. It was also unclear whether system calls set C's `errno` or Perl's `$!` [perl #91614].

- Documentation for $$ has been amended with additional cautions regarding changing the process ID.

*Other Changes*

- perlxs was extended with documentation on inline typemaps.

- perlref has a new Circular References section explaining how circularities may not be freed and how to solve that with weak references.
- Parts of perlapi were clarified, and Perl equivalents of some C functions have been added as an additional mode of exposition.
- A few parts of perlre and perlrecharclass were clarified.

### Removed Documentation

*Old OO Documentation*

The old OO tutorials, perltoot, perltooc, and perlboot, have been removed. The perlbot (bag of object tricks) document has been removed as well.

*Development Deltas*

The perldelta files for development releases are no longer packaged with perl. These can still be found in the perl source code repository.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Cannot set tied `@DB::args`

  This error occurs when `caller` tries to set `@DB::args` but finds it tied. Before this error was added, it used to crash instead.

- Cannot tie unreifiable array

  This error is part of a safety check that the `tie` operator does before tying a special array like `@_`. You should never see this message.

- &CORE::%s cannot be called directly

  This occurs when a subroutine in the `CORE::` namespace is called with `&foo` syntax or through a reference. Some subroutines in this package cannot yet be called that way, but must be called as barewords. See "Subroutines in the `CORE` namespace", above.

- Source filters apply only to byte streams

  This new error occurs when you try to activate a source filter (usually by loading a source filter module) within a string passed to `eval` under the `unicode_eval` feature.

*New Warnings*

- defined(@array) is deprecated

  The long-deprecated `defined(@array)` now also warns for package variables. Previously it issued a warning for lexical variables only.

- **length()** used on `%s`

  This new warning occurs when `length` is used on an array or hash, instead of `scalar(@array)` or `scalar(keys %hash)`.

- lvalue attribute `%s` already-defined subroutine

  attributes.pm now emits this warning when the :lvalue attribute is applied to a Perl subroutine that has already been defined, as doing so can have unexpected side-effects.

- overload arg '%s' is invalid

  This warning, in the ''overload'' category, is produced when the overload pragma is given an argument it doesn't recognize, presumably a mistyped operator.

- $[ used in `%s` (did you mean $] ?)

  This new warning exists to catch the mistaken use of `$[` in version checks. `$]`, not `$[`, contains the version number.

- Useless assignment to a temporary

  Assigning to a temporary scalar returned from an lvalue subroutine now produces this warning [perl #31946].

- Useless use of \E

  `\E` does nothing unless preceded by `\Q`, `\L` or `\U`.

### Removed Errors

- ‘‘sort is now a reserved word’’

  This error used to occur when `sort` was called without arguments, followed by `;` or `)`. (E.g., `sort;` would die, but `{sort}` was OK.) This error message was added in Perl 3 to catch code like `close(sort)` which would no longer work. More than two decades later, this message is no longer appropriate. Now `sort` without arguments is always allowed, and returns an empty list, as it did in those cases where it was already allowed [perl #90030].

### Changes to Existing Diagnostics

- The ‘‘Applying pattern match...’’ or similar warning produced when an array or hash is on the left-hand side of the `=~` operator now mentions the name of the variable.
- The ‘‘Attempt to free non-existent shared string’’ has had the spelling of ‘‘non-existent’’ corrected to ‘‘nonexistent’’. It was already listed with the correct spelling in perldiag.
- The error messages for using `default` and `when` outside a topicalizer have been standardized to match the messages for `continue` and loop controls. They now read ’Can’t ‘‘default’’ outside a topicalizer’ and ’Can’t ‘‘when’’ outside a topicalizer’. They both used to be ’Can’t use **when()** outside a topicalizer’ [perl #91514].
- The message, ‘‘Code point 0x%X is not Unicode, no properties match it; all inverse properties do’’ has been changed to ‘‘Code point 0x%X is not Unicode, all \p{} matches fail; all \P{} matches succeed’’.
- Redefinition warnings for constant subroutines used to be mandatory, even occurring under `no warnings`. Now they respect the warnings pragma.
- The ‘‘glob failed’’ warning message is now suppressible via `no warnings` [perl #111656].
- The Invalid version format error message now says ‘‘negative version number’’ within the parentheses, rather than ‘‘non-numeric data’’, for negative numbers.
- The two warnings Possible attempt to put comments in **qw()** list and Possible attempt to separate words with commas are no longer mutually exclusive: the same `qw` construct may produce both.
- The uninitialized warning for `y///r` when `$_` is implicit and undefined now mentions the variable name, just like the non−/r variation of the operator.
- The ’Use of ‘‘foo’’ without parentheses is ambiguous’ warning has been extended to apply also to user-defined subroutines with a (;$) prototype, and not just to built-in functions.
- Warnings that mention the names of lexical (`my`) variables with Unicode characters in them now respect the presence or absence of the `:utf8` layer on the output handle, instead of outputting UTF8 regardless. Also, the correct names are included in the strings passed to `$SIG{__WARN__}` handlers, rather than the raw UTF8 bytes.

## Utility Changes

*h2ph*

- h2ph used to generate code of the form

```
unless(defined(&FOO)) {
  sub FOO () {42;}
}
```

  But the subroutine is a compile-time declaration, and is hence unaffected by the condition. It has now been corrected to emit a string `eval` around the subroutine [perl #99368].

*splain*

- *splain* no longer emits backtraces with the first line number repeated.

  This:

```
Uncaught exception from user code:
        Cannot fwiddle the fwuddle at -e line 1.
 at -e line 1
        main::baz() called at -e line 1
        main::bar() called at -e line 1
        main::foo() called at -e line 1
```

  has become this:

```
Uncaught exception from user code:
        Cannot fwiddle the fwuddle at -e line 1.
        main::baz() called at -e line 1
        main::bar() called at -e line 1
        main::foo() called at -e line 1
```

- Some error messages consist of multiple lines that are listed as separate entries in perldiag. splain has been taught to find the separate entries in these cases, instead of simply failing to find the message.

*zipdetails*

- This is a new utility, included as part of an IO::Compress::Base upgrade.

  zipdetails displays information about the internal record structure of the zip file. It is not concerned with displaying any details of the compressed data stored in the zip file.

## Configuration and Compilation

- *regexp.h* has been modified for compatibility with GCC's **-Werror** option, as used by some projects that include perl's header files (5.14.1).
- `USE_LOCALE{,_COLLATE,_CTYPE,_NUMERIC}` have been added the output of perl -V as they have affect the behavior of the interpreter binary (albeit in only a small area).
- The code and tests for IPC::Open2 have been moved from *ext/IPC-Open2* into *ext/IPC-Open3*, as `IPC::Open2::open2()` is implemented as a thin wrapper around `IPC::Open3::_open3()`, and hence is very tightly coupled to it.
- The magic types and magic vtables are now generated from data in a new script *regen/mg_vtable.pl*, instead of being maintained by hand. As different EBCDIC variants can't agree on the code point for '~', the character to code point conversion is done at build time by *generate_uudmap* to a new generated header *mg_data.h*. `PL_vtbl_bm` and `PL_vtbl_fm` are now defined by the pre-processor as `PL_vtbl_regexp`, instead of being distinct C variables. `PL_vtbl_sig` has been removed.
- Building with `-DPERL_GLOBAL_STRUCT` works again. This configuration is not generally used.
- Perl configured with *MAD* now correctly frees `MADPROP` structures when OPs are freed. `MADPROP`s are now allocated with `PerlMemShared_malloc()`
- *makedef.pl* has been refactored. This should have no noticeable affect on any of the platforms that use it as part of their build (AIX, VMS, Win32).
- `useperlio` can no longer be disabled.
- The file *global.sym* is no longer needed, and has been removed. It contained a list of all exported functions, one of the files generated by *regen/embed.pl* from data in *embed.fnc* and *regen/opcodes*. The code has been refactored so that the only user of *global.sym*, *makedef.pl*, now reads *embed.fnc* and *regen/opcodes* directly, removing the need to store the list of exported functions in an intermediate file.

  As *global.sym* was never installed, this change should not be visible outside the build process.

- *pod/buildtoc*, used by the build process to build perltoc, has been refactored and simplified. It now contains only code to build perltoc; the code to regenerate Makefiles has been moved to *Porting/pod_rules.pl*. It's a bug if this change has any material effect on the build process.
- *pod/roffitall* is now built by *pod/buildtoc*, instead of being shipped with the distribution. Its list of manpages is now generated (and therefore current). See also RT #103202 for an unresolved related issue.
- The man page for `XS::Typemap` is no longer installed. `XS::Typemap` is a test module which is not installed, hence installing its documentation makes no sense.
- The −Dusesitecustomize and −Duserelocatableinc options now work together properly.

## Platform Support

### Platform-Specific Notes

*Cygwin*

- Since version 1.7, Cygwin supports native UTF−8 paths. If Perl is built under that environment, directory and filenames will be UTF−8 encoded.
- Cygwin does not initialize all original Win32 environment variables. See *README.cygwin* for a discussion of the newly-added `Cygwin::sync_winenv()` function [perl #110190] and for further links.

*HP-UX*

- HP-UX PA−RISC/64 now supports gcc−4.x

  A fix to correct the socketsize now makes the test suite pass on HP-UX PA-RISC for 64bitall builds. (5.14.2)

*VMS*

- Remove unnecessary includes, fix miscellaneous compiler warnings and close some unclosed comments on *vms/vms.c*.
- Remove sockadapt layer from the VMS build.
- Explicit support for VMS versions before v7.0 and DEC C versions before v6.0 has been removed.
- Since Perl 5.10.1, the home-grown `stat` wrapper has been unable to distinguish between a directory name containing an underscore and an otherwise-identical filename containing a dot in the same position (e.g., t/test_pl as a directory and t/test.pl as a file). This problem has been corrected.
- The build on VMS now permits names of the resulting symbols in C code for Perl longer than 31 characters. Symbols like `Perl__it_was_the_best_of_times_it_was_the_worst_of_times` can now be created freely without causing the VMS linker to seize up.

*GNU/Hurd*

- Numerous build and test failures on GNU/Hurd have been resolved with hints for building DBM modules, detection of the library search path, and enabling of large file support.

*OpenVOS*

- Perl is now built with dynamic linking on OpenVOS, the minimum supported version of which is now Release 17.1.0.

*SunOS*

The CC workshop C++ compiler is now detected and used on systems that ship without cc.

## Internal Changes

- The compiled representation of formats is now stored via the `mg_ptr` of their `PERL_MAGIC_fm`. Previously it was stored in the string buffer, beyond `SvLEN()`, the regular end of the string. `SvCOMPILED()` and `SvCOMPILED_{on,off}()` now exist solely for compatibility for XS code. The first is always 0, the other two now no-ops. (5.14.1)

- Some global variables have been marked `const`, members in the interpreter structure have been re-ordered, and the opcodes have been re-ordered. The op `OP_AELEMFAST` has been split into `OP_AELEMFAST` and `OP_AELEMFAST_LEX`.
- When empting a hash of its elements (e.g., via undef(%h), or `%h=()`), HvARRAY field is no longer temporarily zeroed. Any destructors called on the freed elements see the remaining elements. Thus, `%h=()` becomes more like `delete $h{$_} for keys %h`.
- Boyer-Moore compiled scalars are now PVMGs, and the Boyer-Moore tables are now stored via the mg_ptr of their `PERL_MAGIC_bm`. Previously they were PVGVs, with the tables stored in the string buffer, beyond `SvLEN()`. This eliminates the last place where the core stores data beyond `SvLEN()`.
- Simplified logic in `Perl_sv_magic()` introduces a small change of behavior for error cases involving unknown magic types. Previously, if `Perl_sv_magic()` was passed a magic type unknown to it, it would

  1. Croak ‘‘Modification of a read-only value attempted’’ if read only
  2. Return without error if the SV happened to already have this magic
  3. otherwise croak ‘‘Don’t know how to handle magic of type \\%o’’

  Now it will always croak ‘‘Don’t know how to handle magic of type \\%o’’, even on read-only values, or SVs which already have the unknown magic type.
- The experimental `fetch_cop_label` function has been renamed to `cop_fetch_label`.
- The `cop_store_label` function has been added to the API, but is experimental.
- *embedvar.h* has been simplified, and one level of macro indirection for PL_* variables has been removed for the default (non-multiplicity) configuration. PERLVAR*() macros now directly expand their arguments to tokens such as `PL_defgv`, instead of expanding to `PL_Idefgv`, with *embedvar.h* defining a macro to map `PL_Idefgv` to `PL_defgv`. XS code which has unwarranted chumminess with the implementation may need updating.
- An API has been added to explicitly choose whether to export XSUB symbols. More detail can be found in the comments for commit e64345f8.
- The `is_gv_magical_sv` function has been eliminated and merged with `gv_fetchpvn_flags`. It used to be called to determine whether a GV should be autovivified in rvalue context. Now it has been replaced with a new `GV_ADDMG` flag (not part of the API).
- The returned code point from the function `utf8n_to_uvuni()` when the input is malformed UTF-8, malformations are allowed, and `utf8` warnings are off is now the Unicode REPLACEMENT CHARACTER whenever the malformation is such that no well-defined code point can be computed. Previously the returned value was essentially garbage. The only malformations that have well-defined values are a zero-length string (0 is the return), and overlong UTF-8 sequences.
- Padlists are now marked `AvREAL`; i.e., reference-counted. They have always been reference-counted, but were not marked real, because *pad.c* did its own clean-up, instead of using the usual clean-up code in *sv.c*. That caused problems in thread cloning, so now the `AvREAL` flag is on, but is turned off in *pad.c* right before the padlist is freed (after *pad.c* has done its custom freeing of the pads).
- All C files that make up the Perl core have been converted to UTF-8.
- These new functions have been added as part of the work on Unicode symbols:

```
HvNAMELEN
HvNAMEUTF8
HvENAMELEN
HvENAMEUTF8
gv_init_pv
gv_init_pvn
gv_init_pvsv
gv_fetchmeth_pv
gv_fetchmeth_pvn
gv_fetchmeth_sv
gv_fetchmeth_pv_autoload
gv_fetchmeth_pvn_autoload
gv_fetchmeth_sv_autoload
gv_fetchmethod_pv_flags
gv_fetchmethod_pvn_flags
gv_fetchmethod_sv_flags
gv_autoload_pv
gv_autoload_pvn
gv_autoload_sv
newGVgen_flags
sv_derived_from_pv
sv_derived_from_pvn
sv_derived_from_sv
sv_does_pv
sv_does_pvn
sv_does_sv
whichsig_pv
whichsig_pvn
whichsig_sv
newCONSTSUB_flags
```

The gv_fetchmethod_*_flags functions, like gv_fetchmethod_flags, are experimental and may change in a future release.

- The following functions were added. These are *not* part of the API:

```
GvNAMEUTF8
GvENAMELEN
GvENAME_HEK
CopSTASH_flags
CopSTASH_flags_set
PmopSTASH_flags
PmopSTASH_flags_set
sv_sethek
HEKfARG
```

There is also a `HEKf` macro corresponding to `SVf`, for interpolating HEKs in formatted strings.

- `sv_catpvn_flags` takes a couple of new internal-only flags, `SV_CATBYTES` and `SV_CATUTF8`, which tell it whether the char array to be concatenated is UTF8. This allows for more efficient concatenation than creating temporary SVs to pass to `sv_catsv`.
- For XS AUTOLOAD subs, `$AUTOLOAD` is set once more, as it was in 5.6.0. This is in addition to setting `SvPVX(cv)`, for compatibility with 5.8 to 5.14. See ‘‘Autoloading with XSUBs’’ in perlguts.
- Perl now checks whether the array (the linearized isa) returned by a MRO plugin begins with the name of the class itself, for which the array was created, instead of assuming that it does. This prevents the first element from being skipped during method lookup. It also means that `mro::get_linear_isa` may return an array with one more element than the MRO plugin provided [perl #94306].

- `PL_curstash` is now reference-counted.
- There are now feature bundle hints in `PL_hints` (`$^H`) that version declarations use, to avoid having to load *feature.pm*. One setting of the hint bits indicates a ''custom'' feature bundle, which means that the entries in `%^H` still apply. *feature.pm* uses that.

  The `HINT_FEATURE_MASK` macro is defined in *perl.h* along with other hints. Other macros for setting and testing features and bundles are in the new *feature.h*. `FEATURE_IS_ENABLED` (which has moved to *feature.h*) is no longer used throughout the codebase, but more specific macros, e.g., `FEATURE_SAY_IS_ENABLED`, that are defined in *feature.h*.
- *lib/feature.pm* is now a generated file, created by the new *regen/feature.pl* script, which also generates *feature.h*.
- Tied arrays are now always `AvREAL`. If `@_` or `DB::args` is tied, it is reified first, to make sure this is always the case.
- Two new functions `utf8_to_uvchr_buf()` and `utf8_to_uvuni_buf()` have been added. These are the same as `utf8_to_uvchr` and `utf8_to_uvuni` (which are now deprecated), but take an extra parameter that is used to guard against reading beyond the end of the input string. See ''utf8_to_uvchr_buf'' in perlapi and ''utf8_to_uvuni_buf'' in perlapi.
- The regular expression engine now does TRIE case insensitive matches under Unicode. This may change the output of `use re 'debug';`, and will speed up various things.
- There is a new `wrap_op_checker()` function, which provides a thread-safe alternative to writing to `PL_check` directly.

## Selected Bug Fixes

### Array and hash

- A bug has been fixed that would cause a ''Use of freed value in iteration'' error if the next two hash elements that would be iterated over are deleted [perl #85026]. (5.14.1)
- Deleting the current hash iterator (the hash element that would be returned by the next call to `each`) in void context used not to free it [perl #85026].
- Deletion of methods via `delete $Class::{method}` syntax used to update method caches if called in void context, but not scalar or list context.
- When hash elements are deleted in void context, the internal hash entry is now freed before the value is freed, to prevent destructors called by that latter freeing from seeing the hash in an inconsistent state. It was possible to cause double-frees if the destructor freed the hash itself [perl #100340].
- A `keys` optimization in Perl 5.12.0 to make it faster on empty hashes caused `each` not to reset the iterator if called after the last element was deleted.
- Freeing deeply nested hashes no longer crashes [perl #44225].
- It is possible from XS code to create hashes with elements that have no values. The hash element and slice operators used to crash when handling these in lvalue context. They now produce a ''Modification of non-creatable hash value attempted'' error message.
- If list assignment to a hash or array triggered destructors that freed the hash or array itself, a crash would ensue. This is no longer the case [perl #107440].
- It used to be possible to free the typeglob of a localized array or hash (e.g., `local @{"x"}; delete $::{x}`), resulting in a crash on scope exit.
- Some core bugs affecting Hash::Util have been fixed: locking a hash element that is a glob copy no longer causes the next assignment to it to corrupt the glob (5.14.2), and unlocking a hash element that holds a copy-on-write scalar no longer causes modifications to that scalar to modify other scalars that were sharing the same string buffer.

### C API fixes

- The `newHVhv` XS function now works on tied hashes, instead of crashing or returning an empty hash.

- The `SvIsCOW` C macro now returns false for read-only copies of typeglobs, such as those created by:

  ```
  $hash{elem} = *foo;
  Hash::Util::lock_value %hash, 'elem';
  ```

  It used to return true.

- The `SvPVutf8` C function no longer tries to modify its argument, resulting in errors [perl #108994].
- `SvPVutf8` now works properly with magical variables.
- `SvPVbyte` now works properly non-PVs.
- When presented with malformed UTF-8 input, the XS-callable functions `is_utf8_string()`, `is_utf8_string_loc()`, and `is_utf8_string_loclen()` could read beyond the end of the input string by up to 12 bytes. This no longer happens. [perl #32080]. However, currently, `is_utf8_char()` still has this defect, see "**is_utf8_char()**" above.
- The C-level `pregcomp` function could become confused about whether the pattern was in UTF8 if the pattern was an overloaded, tied, or otherwise magical scalar [perl #101940].

### Compile-time hints

- Tying `%^H` no longer causes perl to crash or ignore the contents of `%^H` when entering a compilation scope [perl #106282].
- `eval $string` and `require` used not to localize `%^H` during compilation if it was empty at the time the `eval` call itself was compiled. This could lead to scary side effects, like `use re "/m"` enabling other flags that the surrounding code was trying to enable for its caller [perl #68750].
- `eval $string` and `require` no longer localize hints (`$^H` and `%^H`) at run time, but only during compilation of the `$string` or required file. This makes `BEGIN { $^H{foo}=7 }` equivalent to `BEGIN { eval '$^H{foo}=7' }` [perl #70151].
- Creating a BEGIN block from XS code (via `newXS` or `newATTRSUB`) would, on completion, make the hints of the current compiling code the current hints. This could cause warnings to occur in a non-warning scope.

### Copy-on-write scalars

Copy-on-write or shared hash key scalars were introduced in 5.8.0, but most Perl code did not encounter them (they were used mostly internally). Perl 5.10.0 extended them, such that assigning `__PACKAGE__` or a hash key to a scalar would make it copy-on-write. Several parts of Perl were not updated to account for them, but have now been fixed.

- `utf8::decode` had a nasty bug that would modify copy-on-write scalars' string buffers in place (i.e., skipping the copy). This could result in hashes having two elements with the same key [perl #91834]. (5.14.2)
- Lvalue subroutines were not allowing COW scalars to be returned. This was fixed for lvalue scalar context in Perl 5.12.3 and 5.14.0, but list context was not fixed until this release.
- Elements of restricted hashes (see the fields pragma) containing copy-on-write values couldn't be deleted, nor could such hashes be cleared (`%hash = ()`). (5.14.2)
- Localizing a tied variable used to make it read-only if it contained a copy-on-write string. (5.14.2)
- Assigning a copy-on-write string to a stash element no longer causes a double free. Regardless of this change, the results of such assignments are still undefined.
- Assigning a copy-on-write string to a tied variable no longer stops that variable from being tied if it happens to be a PVMG or PVLV internally.
- Doing a substitution on a tied variable returning a copy-on-write scalar used to cause an assertion failure or an "Attempt to free nonexistent shared string" warning.
- This one is a regression from 5.12: In 5.14.0, the bitwise assignment operators `|=`, `^=` and `&=` started leaving the left-hand side undefined if it happened to be a copy-on-write string [perl #108480].

- Storable, Devel::Peek and PerlIO::scalar had similar problems. See ‘‘Updated Modules and Pragmata’’, above.

### The debugger

- *dumpvar.pl*, and therefore the `x` command in the debugger, have been fixed to handle objects blessed into classes whose names contain ‘‘=’’. The contents of such objects used not to be dumped [perl #101814].
- The ‘‘R’’ command for restarting a debugger session has been fixed to work on Windows, or any other system lacking a `POSIX::_SC_OPEN_MAX` constant [perl #87740].
- The `#line 42 foo` directive used not to update the arrays of lines used by the debugger if it occurred in a string eval. This was partially fixed in 5.14, but it worked only for a single `#line 42 foo` in each eval. Now it works for multiple.
- When subroutine calls are intercepted by the debugger, the name of the subroutine or a reference to it is stored in `$DB::sub`, for the debugger to access. Sometimes (such as `$foo = *bar; undef *bar; &$foo`) `$DB::sub` would be set to a name that could not be used to find the subroutine, and so the debugger’s attempt to call it would fail. Now the check to see whether a reference is needed is more robust, so those problems should not happen anymore [rt.cpan.org #69862].
- Every subroutine has a filename associated with it that the debugger uses. The one associated with constant subroutines used to be misallocated when cloned under threads. Consequently, debugging threaded applications could result in memory corruption [perl #96126].

### Dereferencing operators

- `defined(${"..."})`, `defined(*{"..."})`, etc., used to return true for most, but not all built-in variables, if they had not been used yet. This bug affected `${^GLOBAL_PHASE}` and `${^UTF8CACHE}`, among others. It also used to return false if the package name was given as well (`${"::!"}`) [perl #97978, #97492].
- Perl 5.10.0 introduced a similar bug: `defined(*{"foo"})` where ‘‘foo’’ represents the name of a built-in global variable used to return false if the variable had never been used before, but only on the *first* call. This, too, has been fixed.
- Since 5.6.0, `*{ ... }` has been inconsistent in how it treats undefined values. It would die in strict mode or lvalue context for most undefined values, but would be treated as the empty string (with a warning) for the specific scalar return by `undef()` (`&PL_sv_undef` internally). This has been corrected. `undef()` is now treated like other undefined scalars, as in Perl 5.005.

### Filehandle, last-accessed

Perl has an internal variable that stores the last filehandle to be accessed. It is used by `$.` and by `tell` and `eof` without arguments.

- It used to be possible to set this internal variable to a glob copy and then modify that glob copy to be something other than a glob, and still have the last-accessed filehandle associated with the variable after assigning a glob to it again:

```
my $foo = *STDOUT;  # $foo is a glob copy
<$foo>;             # $foo is now the last-accessed handle
$foo = 3;           # no longer a glob
$foo = *STDERR;     # still the last-accessed handle
```

  Now the `$foo = 3` assignment unsets that internal variable, so there is no last-accessed filehandle, just as if `<$foo>` had never happened.

  This also prevents some unrelated handle from becoming the last-accessed handle if `$foo` falls out of scope and the same internal SV gets used for another handle [perl #97988].

- A regression in 5.14 caused these statements not to set that internal variable:

```
my $fh = *STDOUT;
tell $fh;
eof  $fh;
seek $fh, 0,0;
tell     *$fh;
eof      *$fh;
seek     *$fh, 0,0;
readline *$fh;
```

This is now fixed, but `tell *{ *$fh }` still has the problem, and it is not clear how to fix it [perl #106536].

### Filetests and `stat`

The term ''filetests'' refers to the operators that consist of a hyphen followed by a single letter: `-r`, `-x`, `-M`, etc. The term ''stacked'' when applied to filetests means followed by another filetest operator sharing the same operand, as in `-r -x -w $fooo`.

- `stat` produces more consistent warnings. It no longer warns for ''_'' [perl #71002] and no longer skips the warning at times for other unopened handles. It no longer warns about an unopened handle when the operating system's `fstat` function fails.
- `stat` would sometimes return negative numbers for large inode numbers, because it was using the wrong internal C type. [perl #84590]
- `lstat` is documented to fall back to `stat` (with a warning) when given a filehandle. When passed an IO reference, it was actually doing the equivalent of `stat _` and ignoring the handle.
- `-T _` with no preceding `stat` used to produce a confusing ''uninitialized'' warning, even though there is no visible uninitialized value to speak of.
- `-T`, `-B`, `-l` and `-t` now work when stacked with other filetest operators [perl #77388].
- In 5.14.0, filetest ops (`-r`, `-x`, etc.) started calling FETCH on a tied argument belonging to the previous argument to a list operator, if called with a bareword argument or no argument at all. This has been fixed, so `push @foo, $tied, -r` no longer calls FETCH on `$tied`.
- In Perl 5.6, `-l` followed by anything other than a bareword would treat its argument as a file name. That was changed in 5.8 for glob references (`\*foo`), but not for globs themselves (`*foo`). `-l` started returning `undef` for glob references without setting the last stat buffer that the ''_'' handle uses, but only if warnings were turned on. With warnings off, it was the same as 5.6. In other words, it was simply buggy and inconsistent. Now the 5.6 behavior has been restored.
- `-l` followed by a bareword no longer ''eats'' the previous argument to the list operator in whose argument list it resides. Hence, `print "bar", -l foo` now actually prints ''bar'', because `-l` on longer eats it.
- Perl keeps several internal variables to keep track of the last stat buffer, from which file(handle) it originated, what type it was, and whether the last stat succeeded.

  There were various cases where these could get out of synch, resulting in inconsistent or erratic behavior in edge cases (every mention of `-T` applies to `-B` as well):

  - `-T` *`HANDLE`*, even though it does a `stat`, was not resetting the last stat type, so an `lstat _` following it would merrily return the wrong results. Also, it was not setting the success status.
  - Freeing the handle last used by `stat` or a filetest could result in `-T _` using an unrelated handle.
  - `stat` with an IO reference would not reset the stat type or record the filehandle for `-T _` to use.
  - Fatal warnings could cause the stat buffer not to be reset for a filetest operator on an unopened filehandle or `-l` on any handle. Fatal warnings also stopped `-T` from setting `$!`.
  - When the last stat was on an unreadable file, `-T _` is supposed to return `undef`, leaving the last stat buffer unchanged. But it was setting the stat type, causing `lstat _` to stop working.

- `-T` *`FILENAME`* was not resetting the internal stat buffers for unreadable files.

These have all been fixed.

### Formats

- Several edge cases have been fixed with formats and `formline`; in particular, where the format itself is potentially variable (such as with ties and overloading), and where the format and data differ in their encoding. In both these cases, it used to possible for the output to be corrupted [perl #91032].
- `formline` no longer converts its argument into a string in-place. So passing a reference to `formline` no longer destroys the reference [perl #79532].
- Assignment to `$^A` (the format output accumulator) now recalculates the number of lines output.

### `given` and `when`

- `given` was not scoping its implicit `$_` properly, resulting in memory leaks or ''Variable is not available'' warnings [perl #94682].
- `given` was not calling set-magic on the implicit lexical `$_` that it uses. This meant, for example, that `pos` would be remembered from one execution of the same `given` block to the next, even if the input were a different variable [perl #84526].
- `when` blocks are now capable of returning variables declared inside the enclosing `given` block [perl #93548].

### The `glob` operator

- On OSes other than VMS, Perl's `glob` operator (and the `<...>` form) use File::Glob underneath. File::Glob splits the pattern into words, before feeding each word to its `bsd_glob` function.

  There were several inconsistencies in the way the split was done. Now quotation marks (' and ") are always treated as shell-style word delimiters (that allow whitespace as part of a word) and backslashes are always preserved, unless they exist to escape quotation marks. Before, those would only sometimes be the case, depending on whether the pattern contained whitespace. Also, escaped whitespace at the end of the pattern is no longer stripped [perl #40470].
- `CORE::glob` now works as a way to call the default globbing function. It used to respect overrides, despite the `CORE::` prefix.
- Under miniperl (used to configure modules when perl itself is built), `glob` now clears `%ENV` before calling csh, since the latter croaks on some systems if it does not like the contents of the LS_COLORS environment variable [perl #98662].

### Lvalue subroutines

- Explicit return now returns the actual argument passed to return, instead of copying it [perl #72724, #72706].
- Lvalue subroutines used to enforce lvalue syntax (i.e., whatever can go on the left-hand side of `=`) for the last statement and the arguments to return. Since lvalue subroutines are not always called in lvalue context, this restriction has been lifted.
- Lvalue subroutines are less restrictive about what values can be returned. It used to croak on values returned by `shift` and `delete` and from other subroutines, but no longer does so [perl #71172].
- Empty lvalue subroutines (`sub :lvalue {}`) used to return `@_` in list context. All subroutines used to do this, but regular subs were fixed in Perl 5.8.2. Now lvalue subroutines have been likewise fixed.
- Autovivification now works on values returned from lvalue subroutines [perl #7946], as does returning `keys` in lvalue context.
- Lvalue subroutines used to copy their return values in rvalue context. Not only was this a waste of CPU cycles, but it also caused bugs. A `($)` prototype would cause an lvalue sub to copy its return value [perl #51408], and `while(lvalue_sub() =~ m/.../g) { ... }` would loop endlessly [perl #78680].

- When called in potential lvalue context (e.g., subroutine arguments or a list passed to `for`), lvalue subroutines used to copy any read-only value that was returned. E.g., `sub :lvalue { $] }` would not return `$]`, but a copy of it.
- When called in potential lvalue context, an lvalue subroutine returning arrays or hashes used to bind the arrays or hashes to scalar variables, resulting in bugs. This was fixed in 5.14.0 if an array were the first thing returned from the subroutine (but not for `$scalar, @array` or hashes being returned). Now a more general fix has been applied [perl #23790].
- Method calls whose arguments were all surrounded with `my()` or `our()` (as in `$object->method(my($a,$b))`) used to force lvalue context on the subroutine. This would prevent lvalue methods from returning certain values.
- Lvalue sub calls that are not determined to be such at compile time (`&$name` or &{''name''}) are no longer exempt from strict refs if they occur in the last statement of an lvalue subroutine [perl #102486].
- Sub calls whose subs are not visible at compile time, if they occurred in the last statement of an lvalue subroutine, would reject non-lvalue subroutines and die with ''Can't modify non-lvalue subroutine call'' [perl #102486].

  Non-lvalue sub calls whose subs *are* visible at compile time exhibited the opposite bug. If the call occurred in the last statement of an lvalue subroutine, there would be no error when the lvalue sub was called in lvalue context. Perl would blindly assign to the temporary value returned by the non-lvalue subroutine.
- `AUTOLOAD` routines used to take precedence over the actual sub being called (i.e., when autoloading wasn't needed), for sub calls in lvalue or potential lvalue context, if the subroutine was not visible at compile time.
- Applying the `:lvalue` attribute to an XSUB or to an aliased subroutine stub with `sub foo :lvalue;` syntax stopped working in Perl 5.12. This has been fixed.
- Applying the :lvalue attribute to subroutine that is already defined does not work properly, as the attribute changes the way the sub is compiled. Hence, Perl 5.12 began warning when an attempt is made to apply the attribute to an already defined sub. In such cases, the attribute is discarded.

  But the change in 5.12 missed the case where custom attributes are also present: that case still silently and ineffectively applied the attribute. That omission has now been corrected. `sub foo :lvalue :Whatever` (when `foo` is already defined) now warns about the :lvalue attribute, and does not apply it.
- A bug affecting lvalue context propagation through nested lvalue subroutine calls has been fixed. Previously, returning a value in nested rvalue context would be treated as lvalue context by the inner subroutine call, resulting in some values (such as read-only values) being rejected.

### Overloading

- Arithmetic assignment (`$left += $right`) involving overloaded objects that rely on the 'nomethod' override no longer segfault when the left operand is not overloaded.
- Errors that occur when methods cannot be found during overloading now mention the correct package name, as they did in 5.8.x, instead of erroneously mentioning the ''overload'' package, as they have since 5.10.0.
- Undefining `%overload::` no longer causes a crash.

### Prototypes of built-in keywords

- The `prototype` function no longer dies for the `__FILE__`, `__LINE__` and `__PACKAGE__` directives. It now returns an empty-string prototype for them, because they are syntactically indistinguishable from nullary functions like `time`.
- `prototype` now returns `undef` for all overridable infix operators, such as `eq`, which are not callable in any way resembling functions. It used to return incorrect prototypes for some and die for others [perl #94984].
- The prototypes of several built-in functions — `getprotobynumber`, `lock`, `not` and `select`––have been corrected, or at least are now closer to reality than before.

### Regular expressions

- `/[[:ascii:]]/` and `/[[:blank:]]/` now use locale rules under `use locale` when the platform supports that. Previously, they used the platform's native character set.
- `m/[[:ascii:]]/i` and `/\p{ASCII}/i` now match identically (when not under a differing locale). This fixes a regression introduced in 5.14 in which the first expression could match characters outside of ASCII, such as the KELVIN SIGN.
- `/.*/g` would sometimes refuse to match at the end of a string that ends with ''\n''. This has been fixed [perl #109206].
- Starting with 5.12.0, Perl used to get its internal bookkeeping muddled up after assigning `${ qr// }` to a hash element and locking it with Hash::Util. This could result in double frees, crashes, or erratic behavior.
- The new (in 5.14.0) regular expression modifier `/a` when repeated like `/aa` forbids the characters outside the ASCII range that match characters inside that range from matching under `/i`. This did not work under some circumstances, all involving alternation, such as:

  ```
  "\N{KELVIN SIGN}" =~ /k|foo/iaa;
  ```

  succeeded inappropriately. This is now fixed.
- 5.14.0 introduced some memory leaks in regular expression character classes such as `[\w\s]`, which have now been fixed. (5.14.1)
- An edge case in regular expression matching could potentially loop. This happened only under `/i` in bracketed character classes that have characters with multi-character folds, and the target string to match against includes the first portion of the fold, followed by another character that has a multi-character fold that begins with the remaining portion of the fold, plus some more.

  ```
  "s\N{U+DF}" =~ /[\x{DF}foo]/i
  ```

  is one such case. `\xDF` folds to `"ss"`. (5.14.1)
- A few characters in regular expression pattern matches did not match correctly in some circumstances, all involving `/i`. The affected characters are: COMBINING GREEK YPOGEGRAMMENI, GREEK CAPITAL LETTER IOTA, GREEK CAPITAL LETTER UPSILON, GREEK PROSGEGRAMMENI, GREEK SMALL LETTER IOTA WITH DIALYTIKA AND OXIA, GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS, GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND OXIA, GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND TONOS, LATIN SMALL LETTER LONG S, LATIN SMALL LIGATURE LONG S T, and LATIN SMALL LIGATURE ST.
- A memory leak regression in regular expression compilation under threading has been fixed.
- A regression introduced in 5.14.0 has been fixed. This involved an inverted bracketed character class in a regular expression that consisted solely of a Unicode property. That property wasn't getting inverted outside the Latin1 range.
- Three problematic Unicode characters now work better in regex pattern matching under `/i`.

  In the past, three Unicode characters: LATIN SMALL LETTER SHARP S, GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS, and GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND TONOS, along with the sequences that they fold to (including ''ss'' for LATIN SMALL LETTER SHARP S), did not properly match under `/i`. 5.14.0 fixed some of these cases, but introduced others, including a panic when one of the characters or sequences was used in the `(?(DEFINE)` regular expression predicate. The known bugs that were introduced in 5.14 have now been fixed; as well as some other edge cases that have never worked until now. These all involve using the characters and sequences outside bracketed character classes under `/i`. This closes [perl #98546].

  There remain known problems when using certain characters with multi-character folds inside bracketed character classes, including such constructs as `qr/[\N{LATIN SMALL LETTER SHARP}a-z]/i`. These remaining bugs are addressed in [perl #89774].

- RT #78266: The regex engine has been leaking memory when accessing named captures that weren't matched as part of a regex ever since 5.10 when they were introduced; e.g., this would consume over a hundred MB of memory:

```
for (1..10_000_000) {
    if ("foo" =~ /(foo|(?<capture>bar))?/) {
        my $capture = $+{capture}
    }
}
system "ps -o rss $$"'
```

- In 5.14, `/[[:lower:]]/i` and `/[[:upper:]]/i` no longer matched the opposite case. This has been fixed [perl #101970].
- A regular expression match with an overloaded object on the right-hand side would sometimes stringify the object too many times.
- A regression has been fixed that was introduced in 5.14, in `/i` regular expression matching, in which a match improperly fails if the pattern is in UTF-8, the target string is not, and a Latin-1 character precedes a character in the string that should match the pattern. [perl #101710]
- In case-insensitive regular expression pattern matching, no longer on UTF-8 encoded strings does the scan for the start of match look only at the first possible position. This caused matches such as `"f\x{FB00}" =~ /ff/i` to fail.
- The regexp optimizer no longer crashes on debugging builds when merging fixed-string nodes with inconvenient contents.
- A panic involving the combination of the regular expression modifiers `/aa` and the `\b` escape sequence introduced in 5.14.0 has been fixed [perl #95964]. (5.14.2)
- The combination of the regular expression modifiers `/aa` and the `\b` and `\B` escape sequences did not work properly on UTF-8 encoded strings. All non-ASCII characters under `/aa` should be treated as non-word characters, but what was happening was that Unicode rules were used to determine wordness/non-wordness for non-ASCII characters. This is now fixed [perl #95968].
- `(?foo: ...)` no longer loses passed in character set.
- The trie optimization used to have problems with alternations containing an empty `(?:)`, causing `"x" =~ /\A(?>(?:(?:)A|B|C?x))\z/` not to match, whereas it should [perl #111842].
- Use of lexical (`my`) variables in code blocks embedded in regular expressions will no longer result in memory corruption or crashes.

  Nevertheless, these code blocks are still experimental, as there are still problems with the wrong variables being closed over (in loops for instance) and with abnormal exiting (e.g., `die`) causing memory corruption.
- The `\h`, `\H`, `\v` and `\V` regular expression metacharacters used to cause a panic error message when trying to match at the end of the string [perl #96354].
- The abbreviations for four C1 control characters `MW PM`, `RI`, and `ST` were previously unrecognized by `\N{}`, **vianame()**, and **string_vianame()**.
- Mentioning a variable named "&" other than `$&` (i.e., `@&` or `%&`) no longer stops `$&` from working. The same applies to variables named "'" and "`" [perl #24237].
- Creating a `UNIVERSAL::AUTOLOAD` sub no longer stops `%+`, `%-` and `%!` from working some of the time [perl #105024].

### Smartmatching

- `~~` now correctly handles the precedence of Any~~Object, and is not tricked by an overloaded object on the left-hand side.
- In Perl 5.14.0, `$tainted ~~ @array` stopped working properly. Sometimes it would erroneously fail (when `$tainted` contained a string that occurs in the array *after* the first element) or erroneously succeed (when `undef` occurred after the first element) [perl #93590].

### The `sort` operator

- `sort` was not treating `sub {}` and `sub {()}` as equivalent when such a sub was provided as the comparison routine. It used to croak on `sub {()}`.
- `sort` now works once more with custom sort routines that are XSUBs. It stopped working in 5.10.0.
- `sort` with a constant for a custom sort routine, although it produces unsorted results, no longer crashes. It started crashing in 5.10.0.
- Warnings emitted by `sort` when a custom comparison routine returns a non-numeric value now contain ‘‘in sort’’ and show the line number of the `sort` operator, rather than the last line of the comparison routine. The warnings also now occur only if warnings are enabled in the scope where `sort` occurs. Previously the warnings would occur if enabled in the comparison routine’s scope.
- `sort { $a <=> $b }`, which is optimized internally, now produces ‘‘uninitialized’’ warnings for NaNs (not-a-number values), since `<=>` returns `undef` for those. This brings it in line with `sort { 1; $a <=> $b }` and other more complex cases, which are not optimized [perl #94390].

### The `substr` operator

- Tied (and otherwise magical) variables are no longer exempt from the ‘‘Attempt to use reference as lvalue in substr’’ warning.
- That warning now occurs when the returned lvalue is assigned to, not when `substr` itself is called. This makes a difference only if the return value of `substr` is referenced and later assigned to.
- Passing a substring of a read-only value or a typeglob to a function (potential lvalue context) no longer causes an immediate ‘‘Can’t coerce’’ or ‘‘Modification of a read-only value’’ error. That error occurs only if the passed value is assigned to.

  The same thing happens with the ‘‘substr outside of string’’ error. If the lvalue is only read from, not written to, it is now just a warning, as with rvalue `substr`.
- `substr` assignments no longer call FETCH twice if the first argument is a tied variable, just once.

### Support for embedded nulls

Some parts of Perl did not work correctly with nulls (`chr 0`) embedded in strings. That meant that, for instance, `$m = "a\0b"; foo->$m` would call the ‘‘a’’ method, instead of the actual method name contained in `$m`. These parts of perl have been fixed to support nulls:

- Method names
- Typeglob names (including filehandle and subroutine names)
- Package names, including the return value of `ref()`
- Typeglob elements (`*foo{"THING\0stuff"}`)
- Signal names
- Various warnings and error messages that mention variable names or values, methods, etc.

One side effect of these changes is that blessing into ‘‘\0’’ no longer causes `ref()` to return false.

### Threading bugs

- Typeglobs returned from threads are no longer cloned if the parent thread already has a glob with the same name. This means that returned subroutines will now assign to the right package variables [perl #107366].
- Some cases of threads crashing due to memory allocation during cloning have been fixed [perl #90006].
- Thread joining would sometimes emit ‘‘Attempt to free unreferenced scalar’’ warnings if `caller` had been used from the `DB` package before thread creation [perl #98092].
- Locking a subroutine (via `lock &sub`) is no longer a compile-time error for regular subs. For lvalue subroutines, it no longer tries to return the sub as a scalar, resulting in strange side effects like `ref \$_` returning ‘‘CODE’’ in some instances.

`lock &sub` is now a run-time error if threads::shared is loaded (a no-op otherwise), but that may be rectified in a future version.

### Tied variables

- Various cases in which FETCH was being ignored or called too many times have been fixed:
  - `PerlIO::get_layers` [perl #97956]
  - `$tied =~ y/a/b/`, `chop $tied` and `chomp $tied` when `$tied` holds a reference.
  - When calling `local $_` [perl #105912]
  - Four-argument `select`
  - A tied buffer passed to `sysread`
  - `$tied .= <>`
  - Three-argument `open`, the third being a tied file handle (as in `open $fh, ">&", $tied`)
  - `sort` with a reference to a tied glob for the comparison routine.
  - `..` and `...` in list context [perl #53554].
  - `${$tied}`, `@{$tied}`, `%{$tied}` and `*{$tied}` where the tied variable returns a string (`&{}` was unaffected)
  - `defined ${ $tied_variable }`
  - Various functions that take a filehandle argument in rvalue context (`close`, `readline`, etc.) [perl #97482]
  - Some cases of dereferencing a complex expression, such as `${ (), $tied } = 1`, used to call `FETCH` multiple times, but now call it once.
  - `$tied->method` where `$tied` returns a package name — even resulting in a failure to call the method, due to memory corruption
  - Assignments like `*$tied = \&{"..."}` and `*glob = $tied`
  - `chdir`, `chmod`, `chown`, `utime`, `truncate`, `stat`, `lstat` and the filetest ops (`-r`, `-x`, etc.)
- `caller` sets `@DB::args` to the subroutine arguments when called from the DB package. It used to crash when doing so if `@DB::args` happened to be tied. Now it croaks instead.
- Tying an element of `%ENV` or `%^H` and then deleting that element would result in a call to the tie object's DELETE method, even though tying the element itself is supposed to be equivalent to tying a scalar (the element is, of course, a scalar) [perl #67490].
- When Perl autovivifies an element of a tied array or hash (which entails calling STORE with a new reference), it now calls FETCH immediately after the STORE, instead of assuming that FETCH would have returned the same reference. This can make it easier to implement tied objects [perl #35865, #43011].
- Four-argument `select` no longer produces its ''Non-string passed as bitmask'' warning on tied or tainted variables that are strings.
- Localizing a tied scalar that returns a typeglob no longer stops it from being tied till the end of the scope.
- Attempting to `goto` out of a tied handle method used to cause memory corruption or crashes. Now it produces an error message instead [perl #8611].
- A bug has been fixed that occurs when a tied variable is used as a subroutine reference: if the last thing assigned to or returned from the variable was a reference or typeglob, the `\&$tied` could either crash or return the wrong subroutine. The reference case is a regression introduced in Perl 5.10.0. For typeglobs, it has probably never worked till now.

### Version objects and vstrings

- The bitwise complement operator (and possibly other operators, too) when passed a vstring would leave vstring magic attached to the return value, even though the string had changed. This meant that `version->new(~v1.2.3)` would create a version looking like ''v1.2.3'' even though the

string passed to `version->new` was actually ''\376\375\374''. This also caused B::Deparse to deparse `~v1.2.3` incorrectly, without the `~` [perl #29070].

- Assigning a vstring to a magic (e.g., tied, `$!`) variable and then assigning something else used to blow away all magic. This meant that tied variables would come undone, `$!` would stop getting updated on failed system calls, `$|` would stop setting autoflush, and other mischief would take place. This has been fixed.
- `version->new("version")` and `printf "%vd", "version"` no longer crash [perl #102586].
- Version comparisons, such as those that happen implicitly with `use v5.43`, no longer cause locale settings to change [perl #105784].
- Version objects no longer cause memory leaks in boolean context [perl #109762].

### Warnings, redefinition

- Subroutines from the `autouse` namespace are once more exempt from redefinition warnings. This used to work in 5.005, but was broken in 5.6 for most subroutines. For subs created via XS that redefine subroutines from the `autouse` package, this stopped working in 5.10.
- New XSUBs now produce redefinition warnings if they overwrite existing subs, as they did in 5.8.x. (The `autouse` logic was reversed in 5.10–14. Only subroutines from the `autouse` namespace would warn when clobbered.)
- `newCONSTSUB` used to use compile-time warning hints, instead of run-time hints. The following code should never produce a redefinition warning, but it used to, if `newCONSTSUB` redefined an existing subroutine:

```
use warnings;
BEGIN {
    no warnings;
    some_XS_function_that_calls_new_CONSTSUB();
}
```

- Redefinition warnings for constant subroutines are on by default (what are known as severe warnings in perldiag). This occurred only when it was a glob assignment or declaration of a Perl subroutine that caused the warning. If the creation of XSUBs triggered the warning, it was not a default warning. This has been corrected.
- The internal check to see whether a redefinition warning should occur used to emit ''uninitialized'' warnings in cases like this:

```
use warnings "uninitialized";
use constant {u => undef, v => undef};
sub foo(){u}
sub foo(){v}
```

### Warnings, "Uninitialized"

- Various functions that take a filehandle argument in rvalue context (`close`, `readline`, etc.) used to warn twice for an undefined handle [perl #97482].
- `dbmopen` now only warns once, rather than three times, if the mode argument is `undef` [perl #90064].
- The `+=` operator does not usually warn when the left-hand side is `undef`, but it was doing so for tied variables. This has been fixed [perl #44895].
- A bug fix in Perl 5.14 introduced a new bug, causing ''uninitialized'' warnings to report the wrong variable if the operator in question had two operands and one was `%{...}` or `@{...}`. This has been fixed [perl #103766].
- `..` and `...` in list context now mention the name of the variable in ''uninitialized'' warnings for string (as opposed to numeric) ranges.

### Weak references

- Weakening the first argument to an automatically-invoked `DESTROY` method could result in erroneous ''DESTROY created new reference'' errors or crashes. Now it is an error to weaken a read-only reference.

- Weak references to lexical hashes going out of scope were not going stale (becoming undefined), but continued to point to the hash.
- Weak references to lexical variables going out of scope are now broken before any magical methods (e.g., DESTROY on a tie object) are called. This prevents such methods from modifying the variable that will be seen the next time the scope is entered.
- Creating a weak reference to an `@ISA` array or accessing the array index (`$#ISA`) could result in confused internal bookkeeping for elements later added to the `@ISA` array. For instance, creating a weak reference to the element itself could push that weak reference on to `@ISA`; and elements added after use of `$#ISA` would be ignored by method lookup [perl #85670].

### Other notable fixes

- `quotemeta` now quotes consistently the same non-ASCII characters under `use feature 'unicode_strings'`, regardless of whether the string is encoded in UTF-8 or not, hence fixing the last vestiges (we hope) of the notorious ‘‘The ’’Unicode Bug"" in perlunicode. [perl #77654].

  Which of these code points is quoted has changed, based on Unicode’s recommendations. See ‘‘quotemeta’’ in perlfunc for details.
- `study` is now a no-op, presumably fixing all outstanding bugs related to study causing regex matches to behave incorrectly!
- When one writes `open foo || die`, which used to work in Perl 4, a ‘‘Precedence problem’’ warning is produced. This warning used erroneously to apply to fully-qualified bareword handle names not followed by `||`. This has been corrected.
- After package aliasing (`*foo:: = *bar::`), `select` with 0 or 1 argument would sometimes return a name that could not be used to refer to the filehandle, or sometimes it would return `undef` even when a filehandle was selected. Now it returns a typeglob reference in such cases.
- `PerlIO::get_layers` no longer ignores some arguments that it thinks are numeric, while treating others as filehandle names. It is now consistent for flat scalars (i.e., not references).
- Unrecognized switches on `#!` line

  If a switch, such as **-x**, that cannot occur on the `#!` line is used there, perl dies with ‘‘Can’t emulate...’’.

  It used to produce the same message for switches that perl did not recognize at all, whether on the command line or the `#!` line.

  Now it produces the ‘‘Unrecognized switch’’ error message [perl #104288].
- `system` now temporarily blocks the SIGCHLD signal handler, to prevent the signal handler from stealing the exit status [perl #105700].
- The `%n` formatting code for `printf` and `sprintf`, which causes the number of characters to be assigned to the next argument, now actually assigns the number of characters, instead of the number of bytes.

  It also works now with special lvalue functions like `substr` and with nonexistent hash and array elements [perl #3471, #103492].
- Perl skips copying values returned from a subroutine, for the sake of speed, if doing so would make no observable difference. Because of faulty logic, this would happen with the result of `delete`, `shift` or `splice`, even if the result was referenced elsewhere. It also did so with tied variables about to be freed [perl #91844, #95548].
- `utf8::decode` now refuses to modify read-only scalars [perl #91850].
- Freeing `$_` inside a `grep` or `map` block, a code block embedded in a regular expression, or an `@INC` filter (a subroutine returned by a subroutine in `@INC`) used to result in double frees or crashes [perl #91880, #92254, #92256].
- `eval` returns `undef` in scalar context or an empty list in list context when there is a run-time error. When `eval` was passed a string in list context and a syntax error occurred, it used to return a list containing a single undefined element. Now it returns an empty list in list context for all errors [perl #80630].

- `goto &func` no longer crashes, but produces an error message, when the unwinding of the current subroutine's scope fires a destructor that undefines the subroutine being ''goneto'' [perl #99850].
- Perl now holds an extra reference count on the package that code is currently compiling in. This means that the following code no longer crashes [perl #101486]:

  ```
  package Foo;
  BEGIN {*Foo:: = *Bar::}
  sub foo;
  ```

- The `x` repetition operator no longer crashes on 64−bit builds with large repeat counts [perl #94560].
- Calling `require` on an implicit `$_` when `*CORE::GLOBAL::require` has been overridden does not segfault anymore, and `$_` is now passed to the overriding subroutine [perl #78260].
- `use` and `require` are no longer affected by the I/O layers active in the caller's scope (enabled by open.pm) [perl #96008].
- `our $::é; $é` (which is invalid) no longer produces the ''Compilation error at lib/utf8_heavy.pl...'' error message, which it started emitting in 5.10.0 [perl #99984].
- On 64−bit systems, `read()` now understands large string offsets beyond the 32−bit range.
- Errors that occur when processing subroutine attributes no longer cause the subroutine's op tree to leak.
- Passing the same constant subroutine to both `index` and `formline` no longer causes one or the other to fail [perl #89218]. (5.14.1)
- List assignment to lexical variables declared with attributes in the same statement (`my ($x,@y) : blimp = (72,94)`) stopped working in Perl 5.8.0. It has now been fixed.
- Perl 5.10.0 introduced some faulty logic that made ''U*'' in the middle of a pack template equivalent to ''U0'' if the input string was empty. This has been fixed [perl #90160]. (5.14.2)
- Destructors on objects were not called during global destruction on objects that were not referenced by any scalars. This could happen if an array element were blessed (e.g., `bless \$a[0]`) or if a closure referenced a blessed variable (`bless \my @a; sub foo { @a }`).

  Now there is an extra pass during global destruction to fire destructors on any objects that might be left after the usual passes that check for objects referenced by scalars [perl #36347].
- Fixed a case where it was possible that a freed buffer may have been read from when parsing a here document [perl #90128]. (5.14.1)
- `each(`*`ARRAY`*`)` is now wrapped in `defined(...)`, like `each(`*`HASH`*`)`, inside a `while` condition [perl #90888].
- A problem with context propagation when a `do` block is an argument to `return` has been fixed. It used to cause `undef` to be returned in certain cases of a `return` inside an `if` block which itself is followed by another `return`.
- Calling `index` with a tainted constant no longer causes constants in subsequently compiled code to become tainted [perl #64804].
- Infinite loops like `1 while 1` used to stop `strict 'subs'` mode from working for the rest of the block.
- For list assignments like `($a,$b) = ($b,$a)`, Perl has to make a copy of the items on the right-hand side before assignment them to the left. For efficiency's sake, it assigns the values on the right straight to the items on the left if no one variable is mentioned on both sides, as in `($a,$b) = ($c,$d)`. The logic for determining when it can cheat was faulty, in that `&&` and `||` on the right-hand side could fool it. So `($a,$b) = $some_true_value && ($b,$a)` would end up assigning the value of `$b` to both scalars.
- Perl no longer tries to apply lvalue context to the string in `("string", $variable) ||= 1` (which used to be an error). Since the left-hand side of `||=` is evaluated in scalar context, that's a scalar comma operator, which gives all but the last item void context. There is no such thing as

void lvalue context, so it was a mistake for Perl to try to force it [perl #96942].

- `caller` no longer leaks memory when called from the DB package if `@DB::args` was assigned to after the first call to `caller`. Carp was triggering this bug [perl #97010]. (5.14.2)
- `close` and similar filehandle functions, when called on built-in global variables (like `$+`), used to die if the variable happened to hold the undefined value, instead of producing the usual ''Use of uninitialized value'' warning.
- When autovivified file handles were introduced in Perl 5.6.0, `readline` was inadvertently made to autovivify when called as `readline($foo)` (but not as `<$foo>`). It has now been fixed never to autovivify.
- Calling an undefined anonymous subroutine (e.g., what `$x` holds after `undef &{$x = sub{}}`) used to cause a ''Not a CODE reference'' error, which has been corrected to ''Undefined subroutine called'' [perl #71154].
- Causing `@DB::args` to be freed between uses of `caller` no longer results in a crash [perl #93320].
- `setpgrp($foo)` used to be equivalent to `($foo, setpgrp)`, because `setpgrp` was ignoring its argument if there was just one. Now it is equivalent to `setpgrp($foo,0)`.
- `shmread` was not setting the scalar flags correctly when reading from shared memory, causing the existing cached numeric representation in the scalar to persist [perl #98480].
- `++` and `--` now work on copies of globs, instead of dying.
- `splice()` doesn't warn when truncating

  You can now limit the size of an array using `splice(@a,MAX_LEN)` without worrying about warnings.
- `$$` is no longer tainted. Since this value comes directly from `getpid()`, it is always safe.
- The parser no longer leaks a filehandle if STDIN was closed before parsing started [perl #37033].
- `die;` with a non-reference, non-string, or magical (e.g., tainted) value in $@ now properly propagates that value [perl #111654].

## Known Problems

- On Solaris, we have two kinds of failure.

  If *make* is Sun's *make*, we get an error about a badly formed macro assignment in the *Makefile*. That happens when *./Configure* tries to make depends. *Configure* then exits 0, but further *make*-ing fails.

  If *make* is *gmake*, *Configure* completes, then we get errors related to */usr/include/stdbool.h*
- On Win32, a number of tests hang unless STDERR is redirected. The cause of this is still under investigation.
- When building as root with a umask that prevents files from being other-readable, *t/op/filetest.t* will fail. This is a test bug, not a bug in perl's behavior.
- Configuring with a recent gcc and link-time-optimization, such as `Configure -Doptimize='-O2 -flto'` fails because the optimizer optimizes away some of Configure's tests. A workaround is to omit the `-flto` flag when running Configure, but add it back in while actually building, something like

  ```
  sh Configure -Doptimize=-O2
  make OPTIMIZE='-O2 -flto'
  ```
- The following CPAN modules have test failures with perl 5.16. Patches have been submitted for all of these, so hopefully there will be new releases soon:
  - Date::Pcalc version 6.1
  - Module::CPANTS::Analyse version 0.85

    This fails due to problems in Module::Find 0.10 and File::MMagic 1.27.

- PerlIO::Util version 0.72

## Acknowledgements

Perl 5.16.0 represents approximately 12 months of development since Perl 5.14.0 and contains approximately 590,000 lines of changes across 2,500 files from 139 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.16.0:

Aaron Crane, Abhijit Menon-Sen, Abigail, Alan Haggai Alavi, Alberto Simões, Alexandr Ciornii, Andreas König, Andy Dougherty, Aristotle Pagaltzis, Bo Johansson, Bo Lindbergh, Breno G. de Oliveira, brian d foy, Brian Fraser, Brian Greenfield, Carl Hayter, Chas. Owens, Chia-liang Kao, Chip Salzenberg, Chris 'BinGOs' Williams, Christian Hansen, Christopher J. Madsen, chromatic, Claes Jacobsson, Claudio Ramirez, Craig A. Berry, Damian Conway, Daniel Kahn Gillmor, Darin McBride, Dave Rolsky, David Cantrell, David Golden, David Leadbeater, David Mitchell, Dee Newcum, Dennis Kaarsemaker, Dominic Hargreaves, Douglas Christopher Wilson, Eric Brine, Father Chrysostomos, Florian Ragwitz, Frederic Briere, George Greer, Gerard Goossen, Gisle Aas, H.Merijn Brand, Hojung Youn, Ian Goodacre, James E Keenan, Jan Dubois, Jerry D. Hedden, Jesse Luehrs, Jesse Vincent, Jilles Tjoelker, Jim Cromie, Jim Meyering, Joel Berger, Johan Vromans, Johannes Plunien, John Hawkinson, John P. Linderman, John Peacock, Joshua ben Jore, Juerd Waalboer, Karl Williamson, Karthik Rajagopalan, Keith Thompson, Kevin J. Woolley, Kevin Ryde, Laurent Dami, Leo Lapworth, Leon Brocard, Leon Timmermans, Louis Strous, Lukas Mai, Marc Green, Marcel Grünauer, Mark A. Stratman, Mark Dootson, Mark Jason Dominus, Martin Hasch, Matthew Horsfall, Max Maischein, Michael G Schwern, Michael Witten, Mike Sheldrake, Moritz Lenz, Nicholas Clark, Niko Tyni, Nuno Carvalho, Pau Amma, Paul Evans, Paul Green, Paul Johnson, Perlover, Peter John Acklam, Peter Martini, Peter Scott, Phil Monsen, Pino Toscano, Rafael Garcia-Suarez, Rainer Tammer, Reini Urban, Ricardo Signes, Robin Barker, Rodolfo Carvalho, Salvador Fandiño, Sam Kimbrel, Samuel Thibault, Shawn M Moore, Shigeya Suzuki, Shirakata Kentaro, Shlomi Fish, Sisyphus, Slaven Rezic, Spiros Denaxas, Steffen Müller, Steffen Schwigon, Stephen Bennett, Stephen Oberholtzer, Stevan Little, Steve Hay, Steve Peters, Thomas Sibley, Thorsten Glaser, Timothe Litt, Todd Rinaldo, Tom Christiansen, Tom Hukins, Tony Cook, Vadim Konovalov, Vincent Pit, Vladimir Timofeev, Walt Mankowski, Yves Orton, Zefram, Zsbán Ambrus, ÆEvar Arnfjörð Bjarmason.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at <http://rt.perl.org/perlbug/>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please use this address only for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5161delta − what is new for perl v5.16.1

## DESCRIPTION

This document describes differences between the 5.16.0 release and the 5.16.1 release.

If you are upgrading from an earlier release such as 5.14.0, first read perl5160delta, which describes differences between 5.14.0 and 5.16.0.

## Security

### an off-by-two error in Scalar-List-Util has been fixed

The bugfix was in Scalar-List-Util 1.23_04, and perl 5.16.1 includes Scalar-List-Util 1.25.

## Incompatible Changes

There are no changes intentionally incompatible with 5.16.0 If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Scalar::Util and List::Util have been upgraded from version 1.23 to version 1.25.
- B::Deparse has been updated from version 1.14 to 1.14_01. An ‘‘uninitialized’’ warning emitted by B::Deparse has been squashed [perl #113464].

## Configuration and Compilation

- Building perl with some Windows compilers used to fail due to a problem with miniperl’s `glob` operator (which uses the `perlglob` program) deleting the PATH environment variable [perl #113798].

## Platform Support

### Platform-Specific Notes

VMS

All C header files from the top-level directory of the distribution are now installed on VMS, providing consistency with a long-standing practice on other platforms. Previously only a subset were installed, which broke non-core extension builds for extensions that depended on the missing include files.

## Selected Bug Fixes

- A regression introduced in Perl v5.16.0 involving `tr/SEARCHLIST/REPLACEMENTLIST/` has been fixed. Only the first instance is supposed to be meaningful if a character appears more than once in *SEARCHLIST*. Under some circumstances, the final instance was overriding all earlier ones. [perl #113584]
- `B::COP::stashlen` has been added. This provides access to an internal field added in perl 5.16 under threaded builds. It was broken at the last minute before 5.16 was released [perl #113034].
- The re pragma will no longer clobber `$_`. [perl #113750]
- Unicode 6.1 published an incorrect alias for one of the Canonical_Combining_Class property’s values (which range between 0 and 254). The alias `CCC133` should have been `CCC132`. Perl now overrides the data file furnished by Unicode to give the correct value.
- Duplicating scalar filehandles works again. [perl #113764]
- Under threaded perls, a runtime code block in a regular expression could corrupt the package name stored in the op tree, resulting in bad reads in `caller`, and possibly crashes [perl #113060].
- For efficiency’s sake, many operators and built-in functions return the same scalar each time. Lvalue subroutines and subroutines in the CORE:: namespace were allowing this implementation detail to leak through. `print &CORE::uc("a"), &CORE::uc("b")` used to print ‘‘BB’’. The same thing would happen with an lvalue subroutine returning the return value of `uc`. Now the value is copied in such cases [perl #113044].
- `__SUB__` now works in special blocks (`BEGIN`, `END`, etc.).
- Formats that reference lexical variables from outside no longer result in crashes.

## Known Problems

There are no new known problems, but consult ‘‘Known Problems’’ in perl5160delta to see those identified in the 5.16.0 release.

## Acknowledgements

Perl 5.16.1 represents approximately 2 months of development since Perl 5.16.0 and contains approximately 14,000 lines of changes across 96 files from 8 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.16.1:

Chris ’BinGOs’ Williams, Craig A. Berry, Father Chrysostomos, Karl Williamson, Paul Johnson, Reini Urban, Ricardo Signes, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5162delta − what is new for perl v5.16.2

## DESCRIPTION

This document describes differences between the 5.16.1 release and the 5.16.2 release.

If you are upgrading from an earlier release such as 5.16.0, first read perl5161delta, which describes differences between 5.16.0 and 5.16.1.

## Incompatible Changes

There are no changes intentionally incompatible with 5.16.0 If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 2.70 to version 2.76.

## Configuration and Compilation

- configuration should no longer be confused by ls colorization

## Platform Support

### Platform-Specific Notes

AIX

Configure now always adds −qlanglvl=extc99 to the CC flags on AIX when using xlC. This will make it easier to compile a number of XS-based modules that assume C99 [perl #113778].

## Selected Bug Fixes

- fix /\h/ equivalence with /[\h]/

  see [perl #114220]

## Known Problems

There are no new known problems.

## Acknowledgements

Perl 5.16.2 represents approximately 2 months of development since Perl 5.16.1 and contains approximately 740 lines of changes across 20 files from 9 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.16.2:

Andy Dougherty, Craig A. Berry, Darin McBride, Dominic Hargreaves, Karen Etheridge, Karl Williamson, Peter Martini, Ricardo Signes, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5163delta − what is new for perl v5.16.3

## DESCRIPTION

This document describes differences between the 5.16.2 release and the 5.16.3 release.

If you are upgrading from an earlier release such as 5.16.1, first read perl5162delta, which describes differences between 5.16.1 and 5.16.2.

## Core Enhancements

No changes since 5.16.0.

## Security

This release contains one major and a number of minor security fixes. These latter are included mainly to allow the test suite to pass cleanly with the clang compiler's address sanitizer facility.

### CVE−2013−1667: memory exhaustion with arbitrary hash keys

With a carefully crafted set of hash keys (for example arguments on a URL), it is possible to cause a hash to consume a large amount of memory and CPU, and thus possibly to achieve a Denial-of-Service.

This problem has been fixed.

### wrap-around with IO on long strings

Reading or writing strings greater than 2**31 bytes in size could segfault due to integer wraparound.

This problem has been fixed.

### memory leak in Encode

The UTF−8 encoding implementation in Encode.xs had a memory leak which has been fixed.

## Incompatible Changes

There are no changes intentionally incompatible with 5.16.0. If any exist, they are bugs and reports are welcome.

## Deprecations

There have been no deprecations since 5.16.0.

## Modules and Pragmata

### Updated Modules and Pragmata

- Encode has been upgraded from version 2.44 to version 2.44_01.
- Module::CoreList has been upgraded from version 2.76 to version 2.76_02.
- XS::APItest has been upgraded from version 0.38 to version 0.39.

## Known Problems

None.

## Acknowledgements

Perl 5.16.3 represents approximately 4 months of development since Perl 5.16.2 and contains approximately 870 lines of changes across 39 files from 7 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.16.3:

Andy Dougherty, Chris 'BinGOs' Williams, Dave Rolsky, David Mitchell, Michael Schroeder, Ricardo Signes, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output

of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5180delta − what is new for perl v5.18.0

## DESCRIPTION

This document describes differences between the v5.16.0 release and the v5.18.0 release.

If you are upgrading from an earlier release such as v5.14.0, first read perl5160delta, which describes differences between v5.14.0 and v5.16.0.

## Core Enhancements

### New mechanism for experimental features

Newly-added experimental features will now require this incantation:

```
no warnings "experimental::feature_name";
use feature "feature_name";  # would warn without the prev line
```

There is a new warnings category, called ‘‘experimental’’, containing warnings that the feature pragma emits when enabling experimental features.

Newly-added experimental features will also be given special warning IDs, which consist of ‘‘experimental::’’ followed by the name of the feature. (The plan is to extend this mechanism eventually to all warnings, to allow them to be enabled or disabled individually, and not just by category.)

By saying

```
no warnings "experimental::feature_name";
```

you are taking responsibility for any breakage that future changes to, or removal of, the feature may cause.

Since some features (like `~~` or `my $_`) now emit experimental warnings, and you may want to disable them in code that is also run on perls that do not recognize these warning categories, consider using the `if` pragma like this:

```
no if $] >= 5.018, warnings => "experimental::feature_name";
```

Existing experimental features may begin emitting these warnings, too. Please consult perlexperiment for information on which features are considered experimental.

### Hash overhaul

Changes to the implementation of hashes in perl v5.18.0 will be one of the most visible changes to the behavior of existing code.

By default, two distinct hash variables with identical keys and values may now provide their contents in a different order where it was previously identical.

When encountering these changes, the key to cleaning up from them is to accept that **hashes are unordered collections** and to act accordingly.

*Hash randomization*

The seed used by Perl’s hash function is now random. This means that the order which keys/values will be returned from functions like `keys()`, `values()`, and `each()` will differ from run to run.

This change was introduced to make Perl’s hashes more robust to algorithmic complexity attacks, and also because we discovered that it exposes hash ordering dependency bugs and makes them easier to track down.

Toolchain maintainers might want to invest in additional infrastructure to test for things like this. Running tests several times in a row and then comparing results will make it easier to spot hash order dependencies in code. Authors are strongly encouraged not to expose the key order of Perl’s hashes to insecure audiences.

Further, every hash has its own iteration order, which should make it much more difficult to determine what the current hash seed is.

*New hash functions*

Perl v5.18 includes support for multiple hash functions, and changed the default (to ONE_AT_A_TIME_HARD), you can choose a different algorithm by defining a symbol at compile time. For a current list, consult the *INSTALL* document. Note that as of Perl v5.18 we can only recommend

use of the default or SIPHASH. All the others are known to have security issues and are for research purposes only.

*PERL_HASH_SEED environment variable now takes a hex value*

PERL_HASH_SEED no longer accepts an integer as a parameter; instead the value is expected to be a binary value encoded in a hex string, such as ''0xf5867c55039dc724''. This is to make the infrastructure support hash seeds of arbitrary lengths, which might exceed that of an integer. (SipHash uses a 16 byte seed.)

*PERL_PERTURB_KEYS environment variable added*

The PERL_PERTURB_KEYS environment variable allows one to control the level of randomization applied to `keys` and friends.

When PERL_PERTURB_KEYS is 0, perl will not randomize the key order at all. The chance that `keys` changes due to an insert will be the same as in previous perls, basically only when the bucket size is changed.

When PERL_PERTURB_KEYS is 1, perl will randomize keys in a non-repeatable way. The chance that `keys` changes due to an insert will be very high. This is the most secure and default mode.

When PERL_PERTURB_KEYS is 2, perl will randomize keys in a repeatable way. Repeated runs of the same program should produce the same output every time.

PERL_HASH_SEED implies a non-default PERL_PERTURB_KEYS setting. Setting PERL_HASH_SEED=0 (exactly one 0) implies PERL_PERTURB_KEYS=0 (hash key randomization disabled); setting PERL_HASH_SEED to any other value implies PERL_PERTURB_KEYS=2 (deterministic and repeatable hash key randomization). Specifying PERL_PERTURB_KEYS explicitly to a different level overrides this behavior.

***Hash::Util::hash_seed()** now returns a string*

**Hash::Util::hash_seed()** now returns a string instead of an integer. This is to make the infrastructure support hash seeds of arbitrary lengths which might exceed that of an integer. (SipHash uses a 16 byte seed.)

*Output of PERL_HASH_SEED_DEBUG has been changed*

The environment variable PERL_HASH_SEED_DEBUG now makes perl show both the hash function perl was built with, *and* the seed, in hex, in use for that process. Code parsing this output, should it exist, must change to accommodate the new format. Example of the new format:

```
$ PERL_HASH_SEED_DEBUG=1 ./perl -e1
HASH_FUNCTION = MURMUR3 HASH_SEED = 0x1476bb9f
```

### Upgrade to Unicode 6.2

Perl now supports Unicode 6.2. A list of changes from Unicode 6.1 is at <http://www.unicode.org/versions/Unicode6.2.0>.

### Character name aliases may now include non-Latin1-range characters

It is possible to define your own names for characters for use in `\N{...}`, `charnames::vianame()`, etc. These names can now be comprised of characters from the whole Unicode range. This allows for names to be in your native language, and not just English. Certain restrictions apply to the characters that may be used (you can't define a name that has punctuation in it, for example). See ''CUSTOM ALIASES'' in charnames.

### New DTrace probes

The following new DTrace probes have been added:

- `op-entry`
- `loading-file`
- `loaded-file`

`${^LAST_FH}`

This new variable provides access to the filehandle that was last read. This is the handle used by `$.` and by `tell` and `eof` without arguments.

### Regular Expression Set Operations

This is an **experimental** feature to allow matching against the union, intersection, etc., of sets of code points, similar to Unicode::Regex::Set. It can also be used to extend `/x` processing to [bracketed] character classes, and as a replacement of user-defined properties, allowing more complex expressions than they do. See ''Extended Bracketed Character Classes'' in perlrecharclass.

### Lexical subroutines

This new feature is still considered **experimental**. To enable it:

```
use 5.018;
no warnings "experimental::lexical_subs";
use feature "lexical_subs";
```

You can now declare subroutines with `state sub foo`, `my sub foo`, and `our sub foo`. (`state sub` requires that the ''state'' feature be enabled, unless you write it as `CORE::state sub foo`.)

`state sub` creates a subroutine visible within the lexical scope in which it is declared. The subroutine is shared between calls to the outer sub.

`my sub` declares a lexical subroutine that is created each time the enclosing block is entered. `state sub` is generally slightly faster than `my sub`.

`our sub` declares a lexical alias to the package subroutine of the same name.

For more information, see ''Lexical Subroutines'' in perlsub.

### Computed Labels

The loop controls `next`, `last` and `redo`, and the special `dump` operator, now allow arbitrary expressions to be used to compute labels at run time. Previously, any argument that was not a constant was treated as the empty string.

### More CORE:: subs

Several more built-in functions have been added as subroutines to the CORE:: namespace − namely, those non-overridable keywords that can be implemented without custom parsers: `defined`, `delete`, `exists`, `glob`, `pos`, `prototype`, `scalar`, `split`, `study`, and `undef`.

As some of these have prototypes, `prototype('CORE::...')` has been changed to not make a distinction between overridable and non-overridable keywords. This is to make `prototype('CORE::pos')` consistent with `prototype(&CORE::pos)`.

### `kill` with negative signal names

`kill` has always allowed a negative signal number, which kills the process group instead of a single process. It has also allowed signal names. But it did not behave consistently, because negative signal names were treated as 0. Now negative signals names like `-INT` are supported and treated the same way as −2 [perl #112990].

## Security

### See also: hash overhaul

Some of the changes in the hash overhaul were made to enhance security. Please read that section.

### `Storable` security warning in documentation

The documentation for `Storable` now includes a section which warns readers of the danger of accepting Storable documents from untrusted sources. The short version is that deserializing certain types of data can lead to loading modules and other code execution. This is documented behavior and wanted behavior, but this opens an attack vector for malicious entities.

### `Locale::Maketext` allowed code injection via a malicious template

If users could provide a translation string to Locale::Maketext, this could be used to invoke arbitrary Perl subroutines available in the current process.

This has been fixed, but it is still possible to invoke any method provided by `Locale::Maketext` itself or a subclass that you are using. One of these methods in turn will invoke the Perl core's `sprintf` subroutine.

In summary, allowing users to provide translation strings without auditing them is a bad idea.

This vulnerability is documented in CVE−2012−6329.

### Avoid calling memset with a negative count

Poorly written perl code that allows an attacker to specify the count to perl's `x` string repeat operator can already cause a memory exhaustion denial-of-service attack. A flaw in versions of perl before v5.15.5 can escalate that into a heap buffer overrun; coupled with versions of glibc before 2.16, it possibly allows the execution of arbitrary code.

The flaw addressed to this commit has been assigned identifier CVE-2012-5195 and was researched by Tim Brown.

## Incompatible Changes

### See also: hash overhaul

Some of the changes in the hash overhaul are not fully compatible with previous versions of perl. Please read that section.

### An unknown character name in `\N{...}` is now a syntax error

Previously, it warned, and the Unicode REPLACEMENT CHARACTER was substituted. Unicode now recommends that this situation be a syntax error. Also, the previous behavior led to some confusing warnings and behaviors, and since the REPLACEMENT CHARACTER has no use other than as a stand-in for some unknown character, any code that has this problem is buggy.

### Formerly deprecated characters in `\N{}` character name aliases are now errors.

Since v5.12.0, it has been deprecated to use certain characters in user-defined `\N{...}` character names. These now cause a syntax error. For example, it is now an error to begin a name with a digit, such as in

```
my $undraftable = "\N{4F}";    # Syntax error!
```

or to have commas anywhere in the name. See ''CUSTOM ALIASES'' in charnames.

### `\N{BELL}` now refers to U+1F514 instead of U+0007

Unicode 6.0 reused the name ''BELL'' for a different code point than it traditionally had meant. Since Perl v5.14, use of this name still referred to U+0007, but would raise a deprecation warning. Now, ''BELL'' refers to U+1F514, and the name for U+0007 is ''ALERT''. All the functions in charnames have been correspondingly updated.

### New Restrictions in Multi-Character Case-Insensitive Matching in Regular Expression Bracketed Character Classes

Unicode has now withdrawn their previous recommendation for regular expressions to automatically handle cases where a single character can match multiple characters case-insensitively, for example, the letter LATIN SMALL LETTER SHARP S and the sequence `ss`. This is because it turns out to be impracticable to do this correctly in all circumstances. Because Perl has tried to do this as best it can, it will continue to do so. (We are considering an option to turn it off.) However, a new restriction is being added on such matches when they occur in [bracketed] character classes. People were specifying things such as `/[\0-\xff]/i`, and being surprised that it matches the two character sequence `ss` (since LATIN SMALL LETTER SHARP S occurs in this range). This behavior is also inconsistent with using a property instead of a range: `\p{Block=Latin1}` also includes LATIN SMALL LETTER SHARP S, but `/[\p{Block=Latin1}]/i` does not match `ss`. The new rule is that for there to be a multi-character case-insensitive match within a bracketed character class, the character must be explicitly listed, and not as an end point of a range. This more closely obeys the Principle of Least Astonishment. See ''Bracketed Character Classes'' in perlrecharclass. Note that a bug [perl #89774], now fixed as part of this change, prevented the previous behavior from working fully.

### Explicit rules for variable names and identifiers

Due to an oversight, single character variable names in v5.16 were completely unrestricted. This opened the door to several kinds of insanity. As of v5.18, these now follow the rules of other identifiers, in addition to accepting characters that match the `\p{POSIX_Punct}` property.

There is no longer any difference in the parsing of identifiers specified by using braces versus without braces. For instance, perl used to allow `${foo:bar}` (with a single colon) but not `$foo:bar`. Now that both are handled by a single code path, they are both treated the same way: both are forbidden. Note that this change is about the range of permissible literal identifiers, not other expressions.

### Vertical tabs are now whitespace

No one could recall why `\s` didn't match `\cK`, the vertical tab. Now it does. Given the extreme rarity of that character, very little breakage is expected. That said, here's what it means:

`\s` in a regex now matches a vertical tab in all circumstances.

Literal vertical tabs in a regex literal are ignored when the `/x` modifier is used.

Leading vertical tabs, alone or mixed with other whitespace, are now ignored when interpreting a string as a number. For example:

```
$dec = " \cK \t 123";
$hex = " \cK \t 0xF";

say 0 + $dec;   # was 0 with warning, now 123
say int $dec;   # was 0, now 123
say oct $hex;   # was 0, now  15
```

### `/(?{})/` and `/(??{})/` have been heavily reworked

The implementation of this feature has been almost completely rewritten. Although its main intent is to fix bugs, some behaviors, especially related to the scope of lexical variables, will have changed. This is described more fully in the ''Selected Bug Fixes'' section.

### Stricter parsing of substitution replacement

It is no longer possible to abuse the way the parser parses `s///e` like this:

```
%_=(_,"Just another ");
$_="Perl hacker,\n";
s//_}->{_/e;print
```

### `given` now aliases the global `$_`

Instead of assigning to an implicit lexical `$_`, `given` now makes the global `$_` an alias for its argument, just like `foreach`. However, it still uses lexical `$_` if there is lexical `$_` in scope (again, just like `foreach`) [perl #114020].

### The smartmatch family of features are now experimental

Smart match, added in v5.10.0 and significantly revised in v5.10.1, has been a regular point of complaint. Although there are a number of ways in which it is useful, it has also proven problematic and confusing for both users and implementors of Perl. There have been a number of proposals on how to best address the problem. It is clear that smartmatch is almost certainly either going to change or go away in the future. Relying on its current behavior is not recommended.

Warnings will now be issued when the parser sees `~~`, `given`, or `when`. To disable these warnings, you can add this line to the appropriate scope:

```
no if $] >= 5.018, warnings => "experimental::smartmatch";
```

Consider, though, replacing the use of these features, as they may change behavior again before becoming stable.

### Lexical `$_` is now experimental

Since it was introduced in Perl v5.10, it has caused much confusion with no obvious solution:

- Various modules (e.g., List::Util) expect callback routines to use the global `$_`. `use List::Util 'first'; my $_; first { $_ == 1 } @list` does not work as one would expect.
- A `my $_` declaration earlier in the same file can cause confusing closure warnings.
- The ''_'' subroutine prototype character allows called subroutines to access your lexical `$_`, so it is not really private after all.
- Nevertheless, subroutines with a ''(@)'' prototype and methods cannot access the caller's lexical `$_`, unless they are written in XS.
- But even XS routines cannot access a lexical `$_` declared, not in the calling subroutine, but in an outer scope, iff that subroutine happened not to mention `$_` or use any operators that default to `$_`.

It is our hope that lexical `$_` can be rehabilitated, but this may cause changes in its behavior. Please use it with caution until it becomes stable.

**readline() with `$/ = \N` now reads N characters, not N bytes**

Previously, when reading from a stream with I/O layers such as `encoding`, the **readline()** function, otherwise known as the `<>` operator, would read *N* bytes from the top-most layer. [perl #79960]

Now, *N* characters are read instead.

There is no change in behaviour when reading from streams with no extra layers, since bytes map exactly to characters.

**Overridden `glob` is now passed one argument**

`glob` overrides used to be passed a magical undocumented second argument that identified the caller. Nothing on CPAN was using this, and it got in the way of a bug fix, so it was removed. If you really need to identify the caller, see Devel::Callsite on CPAN.

**Here doc parsing**

The body of a here document inside a quote-like operator now always begins on the line after the ''<<foo'' marker. Previously, it was documented to begin on the line following the containing quote-like operator, but that was only sometimes the case [perl #114040].

**Alphanumeric operators must now be separated from the closing delimiter of regular expressions**

You may no longer write something like:

```
m/a/and 1
```

Instead you must write

```
m/a/ and 1
```

with whitespace separating the operator from the closing delimiter of the regular expression. Not having whitespace has resulted in a deprecation warning since Perl v5.14.0.

**qw(...) can no longer be used as parentheses**

`qw` lists used to fool the parser into thinking they were always surrounded by parentheses. This permitted some surprising constructions such as `foreach $x qw(a b c) {...}`, which should really be written `foreach $x (qw(a b c)) {...}`. These would sometimes get the lexer into the wrong state, so they didn't fully work, and the similar `foreach qw(a b c) {...}` that one might expect to be permitted never worked at all.

This side effect of `qw` has now been abolished. It has been deprecated since Perl v5.13.11. It is now necessary to use real parentheses everywhere that the grammar calls for them.

**Interaction of lexical and default warnings**

Turning on any lexical warnings used first to disable all default warnings if lexical warnings were not already enabled:

```
$*; # deprecation warning
use warnings "void";
$#; # void warning; no deprecation warning
```

Now, the `debugging`, `deprecated`, `glob`, `inplace` and `malloc` warnings categories are left on when turning on lexical warnings (unless they are turned off by `no warnings`, of course).

This may cause deprecation warnings to occur in code that used to be free of warnings.

Those are the only categories consisting only of default warnings. Default warnings in other categories are still disabled by `use warnings "category"`, as we do not yet have the infrastructure for controlling individual warnings.

`state sub` **and** `our sub`

Due to an accident of history, `state sub` and `our sub` were equivalent to a plain `sub`, so one could even create an anonymous sub with `our sub { ... }`. These are now disallowed outside of the ''lexical_subs'' feature. Under the ''lexical_subs'' feature they have new meanings described in ''Lexical Subroutines'' in perlsub.

**Defined values stored in environment are forced to byte strings**

A value stored in an environment variable has always been stringified when inherited by child processes.

In this release, when assigning to `%ENV`, values are immediately stringified, and converted to be only a

byte string.

First, it is forced to be only a string. Then if the string is utf8 and the equivalent of `utf8::downgrade()` works, that result is used; otherwise, the equivalent of `utf8::encode()` is used, and a warning is issued about wide characters (''Diagnostics'').

#### `require` dies for unreadable files

When `require` encounters an unreadable file, it now dies. It used to ignore the file and continue searching the directories in `@INC` [perl #113422].

#### `gv_fetchmeth_*` and SUPER

The various `gv_fetchmeth_*` XS functions used to treat a package whose named ended with `::SUPER` specially. A method lookup on the `Foo::SUPER` package would be treated as a `SUPER` method lookup on the `Foo` package. This is no longer the case. To do a `SUPER` lookup, pass the `Foo` stash and the `GV_SUPER` flag.

#### `split`'s first argument is more consistently interpreted

After some changes earlier in v5.17, `split`'s behavior has been simplified: if the PATTERN argument evaluates to a string containing one space, it is treated the way that a *literal* string containing one space once was.

## Deprecations

### Module removals

The following modules will be removed from the core distribution in a future release, and will at that time need to be installed from CPAN. Distributions on CPAN which require these modules will need to list them as prerequisites.

The core versions of these modules will now issue `"deprecated"`-category warnings to alert you to this fact. To silence these deprecation warnings, install the modules in question from CPAN.

Note that these are (with rare exceptions) fine modules that you are encouraged to continue to use. Their disinclusion from core primarily hinges on their necessity to bootstrapping a fully functional, CPAN-capable Perl installation, not usually on concerns over their design.

encoding

The use of this pragma is now strongly discouraged. It conflates the encoding of source text with the encoding of I/O data, reinterprets escape sequences in source text (a questionable choice), and introduces the UTF-8 bug to all runtime handling of character strings. It is broken as designed and beyond repair.

For using non-ASCII literal characters in source text, please refer to utf8. For dealing with textual I/O data, please refer to Encode and open.

Archive::Extract
B::Lint
B::Lint::Debug
CPANPLUS and all included `CPANPLUS::*` modules
Devel::InnerPackage
Log::Message
Log::Message::Config
Log::Message::Handlers
Log::Message::Item
Log::Message::Simple
Module::Pluggable
Module::Pluggable::Object
Object::Accessor
Pod::LaTeX
Term::UI
Term::UI::History

### Deprecated Utilities

The following utilities will be removed from the core distribution in a future release as their associated modules have been deprecated. They will remain available with the applicable CPAN distribution.

cpanp
`cpanp-run-perl`
cpan2dist
These items are part of the `CPANPLUS` distribution.

pod2latex
This item is part of the `Pod::LaTeX` distribution.

### PL_sv_objcount

This interpreter-global variable used to track the total number of Perl objects in the interpreter. It is no longer maintained and will be removed altogether in Perl v5.20.

### Five additional characters should be escaped in patterns with `/x`

When a regular expression pattern is compiled with `/x`, Perl treats 6 characters as white space to ignore, such as SPACE and TAB. However, Unicode recommends 11 characters be treated thusly. We will conform with this in a future Perl version. In the meantime, use of any of the missing characters will raise a deprecation warning, unless turned off. The five characters are:

```
U+0085 NEXT LINE
U+200E LEFT-TO-RIGHT MARK
U+200F RIGHT-TO-LEFT MARK
U+2028 LINE SEPARATOR
U+2029 PARAGRAPH SEPARATOR
```

### User-defined charnames with surprising whitespace

A user-defined character name with trailing or multiple spaces in a row is likely a typo. This now generates a warning when defined, on the assumption that uses of it will be unlikely to include the excess whitespace.

### Various XS-callable functions are now deprecated

All the functions used to classify characters will be removed from a future version of Perl, and should not be used. With participating C compilers (e.g., gcc), compiling any file that uses any of these will generate a warning. These were not intended for public use; there are equivalent, faster, macros for most of them.

See ''Character classes'' in perlapi. The complete list is:

`is_uni_alnum`, `is_uni_alnumc`, `is_uni_alnumc_lc`, `is_uni_alnum_lc`, `is_uni_alpha`, `is_uni_alpha_lc`, `is_uni_ascii`, `is_uni_ascii_lc`, `is_uni_blank`, `is_uni_blank_lc`, `is_uni_cntrl`, `is_uni_cntrl_lc`, `is_uni_digit`, `is_uni_digit_lc`, `is_uni_graph`, `is_uni_graph_lc`, `is_uni_idfirst`, `is_uni_idfirst_lc`, `is_uni_lower`, `is_uni_lower_lc`, `is_uni_print`, `is_uni_print_lc`, `is_uni_punct`, `is_uni_punct_lc`, `is_uni_space`, `is_uni_space_lc`, `is_uni_upper`, `is_uni_upper_lc`, `is_uni_xdigit`, `is_uni_xdigit_lc`, `is_utf8_alnum`, `is_utf8_alnumc`, `is_utf8_alpha`, `is_utf8_ascii`, `is_utf8_blank`, `is_utf8_char`, `is_utf8_cntrl`, `is_utf8_digit`, `is_utf8_graph`, `is_utf8_idcont`, `is_utf8_idfirst`, `is_utf8_lower`, `is_utf8_mark`, `is_utf8_perl_space`, `is_utf8_perl_word`, `is_utf8_posix_digit`, `is_utf8_print`, `is_utf8_punct`, `is_utf8_space`, `is_utf8_upper`, `is_utf8_xdigit`, `is_utf8_xidcont`, `is_utf8_xidfirst`.

In addition these three functions that have never worked properly are deprecated: `to_uni_lower_lc`, `to_uni_title_lc`, and `to_uni_upper_lc`.

### Certain rare uses of backslashes within regexes are now deprecated

There are three pairs of characters that Perl recognizes as metacharacters in regular expression patterns: `{}`, `[]`, and `()`. These can be used as well to delimit patterns, as in:

```
m{foo}
s(foo)(bar)
```

Since they are metacharacters, they have special meaning to regular expression patterns, and it turns out that you can't turn off that special meaning by the normal means of preceding them with a backslash, if you use them, paired, within a pattern delimited by them. For example, in

```
m{foo\{1,3\}}
```

the backslashes do not change the behavior, and this matches `"f o"` followed by one to three more occurrences of `"o"`.

Usages like this, where they are interpreted as metacharacters, are exceedingly rare; we think there are none, for example, in all of CPAN. Hence, this deprecation should affect very little code. It does give notice, however, that any such code needs to change, which will in turn allow us to change the behavior in future Perl versions so that the backslashes do have an effect, and without fear that we are silently breaking any existing code.

#### Splitting the tokens `(?` and `(*` in regular expressions

A deprecation warning is now raised if the `(` and `?` are separated by white space or comments in `(?...)` regular expression constructs. Similarly, if the `(` and `*` are separated in `(*VERB...)` constructs.

#### Pre-PerlIO IO implementations

In theory, you can currently build perl without PerlIO. Instead, you'd use a wrapper around stdio or sfio. In practice, this isn't very useful. It's not well tested, and without any support for IO layers or (thus) Unicode, it's not much of a perl. Building without PerlIO will most likely be removed in the next version of perl.

PerlIO supports a `stdio` layer if stdio use is desired. Similarly a sfio layer could be produced in the future, if needed.

### Future Deprecations

- Platforms without support infrastructure

  Both Windows CE and z/OS have been historically under-maintained, and are currently neither successfully building nor regularly being smoke tested. Efforts are underway to change this situation, but it should not be taken for granted that the platforms are safe and supported. If they do not become buildable and regularly smoked, support for them may be actively removed in future releases. If you have an interest in these platforms and you can lend your time, expertise, or hardware to help support these platforms, please let the perl development effort know by emailing `perl5-porters@perl.org`.

  Some platforms that appear otherwise entirely dead are also on the short list for removal between now and v5.20.0:

  DG/UX
  NeXT

  We also think it likely that current versions of Perl will no longer build AmigaOS, DJGPP, NetWare (natively), OS/2 and Plan 9. If you are using Perl on such a platform and have an interest in ensuring Perl's future on them, please contact us.

  We believe that Perl has long been unable to build on mixed endian architectures (such as PDP-11s), and intend to remove any remaining support code. Similarly, code supporting the long unmaintained GNU dld will be removed soon if no-one makes themselves known as an active user.

- Swapping of $< and $>

  Perl has supported the idiom of swapping $< and $> (and likewise $( and $)) to temporarily drop permissions since 5.0, like this:

  ```
  ($<, $>) = ($>, $<);
  ```

  However, this idiom modifies the real user/group id, which can have undesirable side-effects, is no longer useful on any platform perl supports and complicates the implementation of these variables and list assignment in general.

  As an alternative, assignment only to `$>` is recommended:

  ```
  local $> = $<;
  ```

  See also: Setuid Demystified <http://www.cs.berkeley.edu/~daw/papers/setuid-usenix02.pdf>.

- `microperl`, long broken and of unclear present purpose, will be removed.
- Revamping `"\Q"` semantics in double-quotish strings when combined with other escapes.

  There are several bugs and inconsistencies involving combinations of `\Q` and escapes like `\x`, `\L`, etc., within a `\Q...\E` pair. These need to be fixed, and doing so will necessarily change current behavior. The changes have not yet been settled.
- Use of `$x`, where `x` stands for any actual (non-printing) C0 control character will be disallowed in a future Perl version. Use `${x}` instead (where again `x` stands for a control character), or better, `$^A` , where `^` is a caret (CIRCUMFLEX ACCENT), and `A` stands for any of the characters listed at the end of ''OPERATOR DIFFERENCES'' in perlebcdic.

## Performance Enhancements

- Lists of lexical variable declarations (`my($x, $y)`) are now optimised down to a single op and are hence faster than before.
- A new C preprocessor define `NO_TAINT_SUPPORT` was added that, if set, disables Perl's taint support altogether. Using the −T or −t command line flags will cause a fatal error. Beware that both core tests as well as many a CPAN distribution's tests will fail with this change. On the upside, it provides a small performance benefit due to reduced branching.

  **Do not enable this unless you know exactly what you are getting yourself into.**
- `pack` with constant arguments is now constant folded in most cases [perl #113470].
- Speed up in regular expression matching against Unicode properties. The largest gain is for `\X`, the Unicode ''extended grapheme cluster.'' The gain for it is about 35% − 40%. Bracketed character classes, e.g., `[0−9\x{100}]` containing code points above 255 are also now faster.
- On platforms supporting it, several former macros are now implemented as static inline functions. This should speed things up slightly on non-GCC platforms.
- The optimisation of hashes in boolean context has been extended to affect `scalar(%hash)`, `%hash ? ... : ...`, and `sub { %hash || ... }`.
- Filetest operators manage the stack in a fractionally more efficient manner.
- Globs used in a numeric context are now numified directly in most cases, rather than being numified via stringification.
- The `x` repetition operator is now folded to a single constant at compile time if called in scalar context with constant operands and no parentheses around the left operand.

## Modules and Pragmata

### New Modules and Pragmata

- Config::Perl::V version 0.16 has been added as a dual-lifed module. It provides structured data retrieval of `perl -V` output including information only known to the `perl` binary and not available via Config.

### Updated Modules and Pragmata

For a complete list of updates, run:

```
$ corelist --diff 5.16.0 5.18.0
```

You can substitute your favorite version in place of `5.16.0`, too.

- Archive::Extract has been upgraded to 0.68.

  Work around an edge case on Linux with Busybox's unzip.
- Archive::Tar has been upgraded to 1.90.

  ptar now supports the −T option as well as dashless options [rt.cpan.org #75473], [rt.cpan.org #75475].

  Auto-encode filenames marked as UTF−8 [rt.cpan.org #75474].

  Don't use `tell` on IO::Zlib handles [rt.cpan.org #64339].

  Don't try to `chown` on symlinks.

- autodie has been upgraded to 2.13.

  `autodie` now plays nicely with the 'open' pragma.

- B has been upgraded to 1.42.

  The `stashoff` method of COPs has been added. This provides access to an internal field added in perl 5.16 under threaded builds [perl #113034].

  `B::COP::stashpv` now supports UTF-8 package names and embedded NULs.

  All `CVf_*` and `GVf_*` and more SV-related flag values are now provided as constants in the `B::` namespace and available for export. The default export list has not changed.

  This makes the module work with the new pad API.

- B::Concise has been upgraded to 0.95.

  The `-nobanner` option has been fixed, and `format`s can now be dumped. When passed a sub name to dump, it will check also to see whether it is the name of a format. If a sub and a format share the same name, it will dump both.

  This adds support for the new `OpMAYBE_TRUEBOOL` and `OPpTRUEBOOL` flags.

- B::Debug has been upgraded to 1.18.

  This adds support (experimentally) for `B::PADLIST`, which was added in Perl 5.17.4.

- B::Deparse has been upgraded to 1.20.

  Avoid warning when run under `perl -w`.

  It now deparses loop controls with the correct precedence, and multiple statements in a `format` line are also now deparsed correctly.

  This release suppresses trailing semicolons in formats.

  This release adds stub deparsing for lexical subroutines.

  It no longer dies when deparsing `sort` without arguments. It now correctly omits the comma for `system $prog @args` and `exec $prog @args`.

- bignum, bigint and bigrat have been upgraded to 0.33.

  The overrides for `hex` and `oct` have been rewritten, eliminating several problems, and making one incompatible change:

  - Formerly, whichever of `use bigint` or `use bigrat` was compiled later would take precedence over the other, causing `hex` and `oct` not to respect the other pragma when in scope.
  - Using any of these three pragmata would cause `hex` and `oct` anywhere else in the program to evaluate their arguments in list context and prevent them from inferring `$_` when called without arguments.
  - Using any of these three pragmata would make `oct("1234")` return 1234 (for any number not beginning with 0) anywhere in the program. Now ‘‘1234’’ is translated from octal to decimal, whether within the pragma's scope or not.
  - The global overrides that facilitate lexical use of `hex` and `oct` now respect any existing overrides that were in place before the new overrides were installed, falling back to them outside of the scope of `use bignum`.
  - `use bignum "hex"`, `use bignum "oct"` and similar invocations for bigint and bigrat now export a `hex` or `oct` function, instead of providing a global override.

- Carp has been upgraded to 1.29.

  Carp is no longer confused when `caller` returns undef for a package that has been deleted.

  The `longmess()` and `shortmess()` functions are now documented.

- CGI has been upgraded to 3.63.

  Unrecognized HTML escape sequences are now handled better, problematic trailing newlines are no longer inserted after <form> tags by `startform()` or `start_form()`, and bogus ''Insecure Dependency'' warnings appearing with some versions of perl are now worked around.

- Class::Struct has been upgraded to 0.64.

  The constructor now respects overridden accessor methods [perl #29230].

- Compress::Raw::Bzip2 has been upgraded to 2.060.

  The misuse of Perl's ''magic'' API has been fixed.

- Compress::Raw::Zlib has been upgraded to 2.060.

  Upgrade bundled zlib to version 1.2.7.

  Fix build failures on Irix, Solaris, and Win32, and also when building as C++ [rt.cpan.org #69985], [rt.cpan.org #77030], [rt.cpan.org #75222].

  The misuse of Perl's ''magic'' API has been fixed.

  `compress()`, `uncompress()`, `memGzip()` and `memGunzip()` have been speeded up by making parameter validation more efficient.

- CPAN::Meta::Requirements has been upgraded to 2.122.

  Treat undef requirements to `from_string_hash` as 0 (with a warning).

  Added `requirements_for_module` method.

- CPANPLUS has been upgraded to 0.9135.

  Allow adding *blib/script* to PATH.

  Save the history between invocations of the shell.

  Handle multiple `makemakerargs` and `makeflags` arguments better.

  This resolves issues with the SQLite source engine.

- Data::Dumper has been upgraded to 2.145.

  It has been optimized to only build a seen-scalar hash as necessary, thereby speeding up serialization drastically.

  Additional tests were added in order to improve statement, branch, condition and subroutine coverage. On the basis of the coverage analysis, some of the internals of Dumper.pm were refactored. Almost all methods are now documented.

- DB_File has been upgraded to 1.827.

  The main Perl module no longer uses the `"@_"` construct.

- Devel::Peek has been upgraded to 1.11.

  This fixes compilation with C++ compilers and makes the module work with the new pad API.

- Digest::MD5 has been upgraded to 2.52.

  Fix `Digest::Perl::MD5` OO fallback [rt.cpan.org #66634].

- Digest::SHA has been upgraded to 5.84.

  This fixes a double-free bug, which might have caused vulnerabilities in some cases.

- DynaLoader has been upgraded to 1.18.

  This is due to a minor code change in the XS for the VMS implementation.

  This fixes warnings about using `CODE` sections without an `OUTPUT` section.

- Encode has been upgraded to 2.49.

  The Mac alias x−mac-ce has been added, and various bugs have been fixed in Encode::Unicode, Encode::UTF7 and Encode::GSM0338.

- Env has been upgraded to 1.04.

  Its SPLICE implementation no longer misbehaves in list context.

- ExtUtils::CBuilder has been upgraded to 0.280210.

  Manifest files are now correctly embedded for those versions of VC++ which make use of them. [perl #111782, #111798].

  A list of symbols to export can now be passed to `link()` when on Windows, as on other OSes [perl #115100].

- ExtUtils::ParseXS has been upgraded to 3.18.

  The generated C code now avoids unnecessarily incrementing `PL_amagic_generation` on Perl versions where it's done automatically (or on current Perl where the variable no longer exists).

  This avoids a bogus warning for initialised XSUB non-parameters [perl #112776].

- File::Copy has been upgraded to 2.26.

  `copy()` no longer zeros files when copying into the same directory, and also now fails (as it has long been documented to do) when attempting to copy a file over itself.

- File::DosGlob has been upgraded to 1.10.

  The internal cache of file names that it keeps for each caller is now freed when that caller is freed. This means `use File::DosGlob 'glob'; eval 'scalar <*>'` no longer leaks memory.

- File::Fetch has been upgraded to 0.38.

  Added the 'file_default' option for URLs that do not have a file component.

  Use `File::HomeDir` when available, and provide `PERL5_CPANPLUS_HOME` to override the autodetection.

  Always re-fetch *CHECKSUMS* if `fetchdir` is set.

- File::Find has been upgraded to 1.23.

  This fixes inconsistent unixy path handling on VMS.

  Individual files may now appear in list of directories to be searched [perl #59750].

- File::Glob has been upgraded to 1.20.

  File::Glob has had exactly the same fix as File::DosGlob. Since it is what Perl's own `glob` operator itself uses (except on VMS), this means `eval 'scalar <*>'` no longer leaks.

  A space-separated list of patterns return long lists of results no longer results in memory corruption or crashes. This bug was introduced in Perl 5.16.0. [perl #114984]

- File::Spec::Unix has been upgraded to 3.40.

  `abs2rel` could produce incorrect results when given two relative paths or the root directory twice [perl #111510].

- File::stat has been upgraded to 1.07.

  `File::stat` ignores the filetest pragma, and warns when used in combination therewith. But it was not warning for `-r`. This has been fixed [perl #111640].

  `-p` now works, and does not return false for pipes [perl #111638].

  Previously `File::stat`'s overloaded `-x` and `-X` operators did not give the correct results for directories or executable files when running as root. They had been treating executable permissions for root just like for any other user, performing group membership tests *etc* for files not owned by root. They now follow the correct Unix behaviour − for a directory they are always true, and for a file if any of the three execute permission bits are set then they report that root can execute the file. Perl's builtin `-x` and `-X` operators have always been correct.

- File::Temp has been upgraded to 0.23

  Fixes various bugs involving directory removal. Defers unlinking tempfiles if the initial unlink fails, which fixes problems on NFS.

- GDBM_File has been upgraded to 1.15.

  The undocumented optional fifth parameter to `TIEHASH` has been removed. This was intended to provide control of the callback used by `gdbm*` functions in case of fatal errors (such as filesystem problems), but did not work (and could never have worked). No code on CPAN even attempted to use it. The callback is now always the previous default, `croak`. Problems on some platforms with how the `C croak` function is called have also been resolved.

- Hash::Util has been upgraded to 0.15.

  `hash_unlocked` and `hashref_unlocked` now returns true if the hash is unlocked, instead of always returning false [perl #112126].

  `hash_unlocked`, `hashref_unlocked`, `lock_hash_recurse` and `unlock_hash_recurse` are now exportable [perl #112126].

  Two new functions, `hash_locked` and `hashref_locked`, have been added. Oddly enough, these two functions were already exported, even though they did not exist [perl #112126].

- HTTP::Tiny has been upgraded to 0.025.

  Add SSL verification features [github #6], [github #9].

  Include the final URL in the response hashref.

  Add `local_address` option.

  This improves SSL support.

- IO has been upgraded to 1.28.

  `sync()` can now be called on read-only file handles [perl #64772].

  IO::Socket tries harder to cache or otherwise fetch socket information.

- IPC::Cmd has been upgraded to 0.80.

  Use `POSIX::_exit` instead of `exit` in `run_forked` [rt.cpan.org #76901].

- IPC::Open3 has been upgraded to 1.13.

  The `open3()` function no longer uses `POSIX::close()` to close file descriptors since that breaks the ref-counting of file descriptors done by PerlIO in cases where the file descriptors are shared by PerlIO streams, leading to attempts to close the file descriptors a second time when any such PerlIO streams are closed later on.

- Locale::Codes has been upgraded to 3.25.

  It includes some new codes.

- Memoize has been upgraded to 1.03.

  Fix the `MERGE` cache option.

- Module::Build has been upgraded to 0.4003.

  Fixed bug where modules without `$VERSION` might have a version of '0' listed in 'provides' metadata, which will be rejected by PAUSE.

  Fixed bug in PodParser to allow numerals in module names.

  Fixed bug where giving arguments twice led to them becoming arrays, resulting in install paths like ***ARRAY****(0xdeadbeef)/lib/Foo.pm*.

  A minor bug fix allows markup to be used around the leading ‘‘Name’’ in a POD ‘‘abstract’’ line, and some documentation improvements have been made.

- Module::CoreList has been upgraded to 2.90

  Version information is now stored as a delta, which greatly reduces the size of the *CoreList.pm* file.

  This restores compatibility with older versions of perl and cleans up the corelist data for various modules.

- Module::Load::Conditional has been upgraded to 0.54.

  Fix use of `requires` on perls installed to a path with spaces.

  Various enhancements include the new use of Module::Metadata.

- Module::Metadata has been upgraded to 1.000011.

  The creation of a Module::Metadata object for a typical module file has been sped up by about 40%, and some spurious warnings about `$VERSION`s have been suppressed.

- Module::Pluggable has been upgraded to 4.7.

  Amongst other changes, triggers are now allowed on events, which gives a powerful way to modify behaviour.

- Net::Ping has been upgraded to 2.41.

  This fixes some test failures on Windows.

- Opcode has been upgraded to 1.25.

  Reflect the removal of the boolkeys opcode and the addition of the clonecv, introcv and padcv opcodes.

- overload has been upgraded to 1.22.

  `no overload` now warns for invalid arguments, just like `use overload`.

- PerlIO::encoding has been upgraded to 0.16.

  This is the module implementing the ‘‘:encoding(...)’’ I/O layer. It no longer corrupts memory or crashes when the encoding back-end reallocates the buffer or gives it a typeglob or shared hash key scalar.

- PerlIO::scalar has been upgraded to 0.16.

  The buffer scalar supplied may now only contain code points 0xFF or lower. [perl #109828]

- Perl::OSType has been upgraded to 1.003.

  This fixes a bug detecting the VOS operating system.

- Pod::Html has been upgraded to 1.18.

  The option `--libpods` has been reinstated. It is deprecated, and its use does nothing other than issue a warning that it is no longer supported.

  Since the HTML files generated by pod2html claim to have a UTF-8 charset, actually write the files out using UTF-8 [perl #111446].

- Pod::Simple has been upgraded to 3.28.

  Numerous improvements have been made, mostly to Pod::Simple::XHTML, which also has a compatibility change: the `codes_in_verbatim` option is now disabled by default. See *cpan/Pod-Simple/ChangeLog* for the full details.

- re has been upgraded to 0.23

  Single character [class]es like `/[s]/` or `/[s]/i` are now optimized as if they did not have the brackets, i.e. `/s/` or `/s/i`.

  See note about `op_comp` in the ‘‘Internal Changes’’ section below.

- Safe has been upgraded to 2.35.

  Fix interactions with `Devel::Cover`.

Don't eval code under `no strict`.

- Scalar::Util has been upgraded to version 1.27.

  Fix an overloading issue with `sum`.

  `first` and `reduce` now check the callback first (so `&first(1)` is disallowed).

  Fix `tainted` on magical values [rt.cpan.org #55763].

  Fix `sum` on previously magical values [rt.cpan.org #61118].

  Fix reading past the end of a fixed buffer [rt.cpan.org #72700].

- Search::Dict has been upgraded to 1.07.

  No longer require `stat` on filehandles.

  Use `fc` for casefolding.

- Socket has been upgraded to 2.009.

  Constants and functions required for IP multicast source group membership have been added.

  `unpack_sockaddr_in()` and `unpack_sockaddr_in6()` now return just the IP address in scalar context, and `inet_ntop()` now guards against incorrect length scalars being passed in.

  This fixes an uninitialized memory read.

- Storable has been upgraded to 2.41.

  Modifying `$_[0]` within `STORABLE_freeze` no longer results in crashes [perl #112358].

  An object whose class implements `STORABLE_attach` is now thawed only once when there are multiple references to it in the structure being thawed [perl #111918].

  Restricted hashes were not always thawed correctly [perl #73972].

  Storable would croak when freezing a blessed REF object with a `STORABLE_freeze()` method [perl #113880].

  It can now freeze and thaw vstrings correctly. This causes a slight incompatible change in the storage format, so the format version has increased to 2.9.

  This contains various bugfixes, including compatibility fixes for older versions of Perl and vstring handling.

- Sys::Syslog has been upgraded to 0.32.

  This contains several bug fixes relating to `getservbyname()`, `setlogsock()`and log levels in `syslog()`, together with fixes for Windows, Haiku-OS and GNU/kFreeBSD. See *cpan/Sys−Syslog/Changes* for the full details.

- Term::ANSIColor has been upgraded to 4.02.

  Add support for italics.

  Improve error handling.

- Term::ReadLine has been upgraded to 1.10. This fixes the use of the **cpan** and **cpanp** shells on Windows in the event that the current drive happens to contain a *\dev\tty* file.

- Test::Harness has been upgraded to 3.26.

  Fix glob semantics on Win32 [rt.cpan.org #49732].

  Don't use `Win32::GetShortPathName` when calling perl [rt.cpan.org #47890].

  Ignore −T when reading shebang [rt.cpan.org #64404].

  Handle the case where we don't know the wait status of the test more gracefully.

  Make the test summary 'ok' line overridable so that it can be changed to a plugin to make the output of prove idempotent.

  Don't run world-writable files.

- Text::Tabs and Text::Wrap have been upgraded to 2012.0818. Support for Unicode combining characters has been added to them both.
- threads::shared has been upgraded to 1.31.

  This adds the option to warn about or ignore attempts to clone structures that can't be cloned, as opposed to just unconditionally dying in that case.

  This adds support for dual-valued values as created by Scalar::Util::dualvar.
- Tie::StdHandle has been upgraded to 4.3.

  `READ` now respects the offset argument to `read` [perl #112826].
- Time::Local has been upgraded to 1.2300.

  Seconds values greater than 59 but less than 60 no longer cause `timegm()` and `timelocal()` to croak.
- Unicode::UCD has been upgraded to 0.53.

  This adds a function **all_casefolds()** that returns all the casefolds.
- Win32 has been upgraded to 0.47.

  New APIs have been added for getting and setting the current code page.

### Removed Modules and Pragmata

- Version::Requirements has been removed from the core distribution. It is available under a different name: CPAN::Meta::Requirements.

## Documentation

### Changes to Existing Documentation

*perlcheat*

- perlcheat has been reorganized, and a few new sections were added.

*perldata*

- Now explicitly documents the behaviour of hash initializer lists that contain duplicate keys.

*perldiag*

- The explanation of symbolic references being prevented by ''strict refs'' now doesn't assume that the reader knows what symbolic references are.

*perlfaq*

- perlfaq has been synchronized with version 5.0150040 from CPAN.

*perlfunc*

- The return value of `pipe` is now documented.
- Clarified documentation of `our`.

*perlop*

- Loop control verbs (`dump`, `goto`, `next`, `last` and `redo`) have always had the same precedence as assignment operators, but this was not documented until now.

*Diagnostics*

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Unterminated delimiter for here document

  This message now occurs when a here document label has an initial quotation mark but the final quotation mark is missing.

  This replaces a bogus and misleading error message about not finding the label itself [perl #114104].

- panic: child pseudo-process was never scheduled

  This error is thrown when a child pseudo-process in the ithreads implementation on Windows was not scheduled within the time period allowed and therefore was not able to initialize properly [perl #88840].

- Group name must start with a non-digit word character in regex; marked by <-- HERE in m/%s/

  This error has been added for `(?&0)`, which is invalid. It used to produce an incomprehensible error message [perl #101666].

- Can't use an undefined value as a subroutine reference

  Calling an undefined value as a subroutine now produces this error message. It used to, but was accidentally disabled, first in Perl 5.004 for non-magical variables, and then in Perl v5.14 for magical (e.g., tied) variables. It has now been restored. In the mean time, undef was treated as an empty string [perl #113576].

- Experimental "%s" subs not enabled

  To use lexical subs, you must first enable them:

  ```
  no warnings 'experimental::lexical_subs';
  use feature 'lexical_subs';
  my sub foo { ... }
  ```

*New Warnings*

- 'Strings with code points over 0xFF may not be mapped into in-memory file handles'
- '%s' resolved to '\o{%s}%d'
- 'Trailing white-space in a charnames alias definition is deprecated'
- 'A sequence of multiple spaces in a charnames alias definition is deprecated'
- 'Passing malformed UTF-8 to "%s" is deprecated'
- Subroutine "&%s" is not available

  (W closure) During compilation, an inner named subroutine or eval is attempting to capture an outer lexical subroutine that is not currently available. This can happen for one of two reasons. First, the lexical subroutine may be declared in an outer anonymous subroutine that has not yet been created. (Remember that named subs are created at compile time, while anonymous subs are created at run-time.) For example,

  ```
  sub { my sub a {...} sub f { \&a } }
  ```

  At the time that f is created, it can't capture the current the "a" sub, since the anonymous subroutine hasn't been created yet. Conversely, the following won't give a warning since the anonymous subroutine has by now been created and is live:

  ```
  sub { my sub a {...} eval 'sub f { \&a }' }->();
  ```

  The second situation is caused by an eval accessing a variable that has gone out of scope, for example,

  ```
  sub f {
      my sub a {...}
      sub { eval '\&a' }
  }
  f()->();
  ```

  Here, when the '\&a' in the eval is being compiled, f() is not currently being executed, so its &a is not available for capture.

- "%s" subroutine &%s masks earlier declaration in same `%s`

  (W misc) A "my" or "state" subroutine has been redeclared in the current scope or statement, effectively eliminating all access to the previous instance. This is almost always a typographical error. Note that the earlier subroutine will still exist until the end of the scope or until all closure references to it are destroyed.

- The %s feature is experimental

  (S experimental) This warning is emitted if you enable an experimental feature via `use feature`. Simply suppress the warning if you want to use the feature, but know that in doing so you are taking the risk of using an experimental feature which may change or be removed in a future Perl version:

  ```
  no warnings "experimental::lexical_subs";
  use feature "lexical_subs";
  ```

- sleep(%u) too large

  (W overflow) You called `sleep` with a number that was larger than it can reliably handle and `sleep` probably slept for less time than requested.

- Wide character in setenv

  Attempts to put wide characters into environment variables via `%ENV` now provoke this warning.

- "Invalid negative number (%s) in chr"

  `chr()` now warns when passed a negative value [perl #83048].

- "Integer overflow in srand"

  `srand()` now warns when passed a value that doesn't fit in a `UV` (since the value will be truncated rather than overflowing) [perl #40605].

- "−i used with no filenames on the command line, reading from STDIN"

  Running perl with the `-i` flag now warns if no input files are provided on the command line [perl #113410].

### Changes to Existing Diagnostics

- $* is no longer supported

  The warning that use of `$*` and `$#` is no longer supported is now generated for every location that references them. Previously it would fail to be generated if another variable using the same typeglob was seen first (e.g. `@*` before `$*`), and would not be generated for the second and subsequent uses. (It's hard to fix the failure to generate warnings at all without also generating them every time, and warning every time is consistent with the warnings that `$[` used to generate.)

- The warnings for `\b{` and `\B{` were added. They are a deprecation warning which should be turned off by that category. One should not have to turn off regular regexp warnings as well to get rid of these.

- Constant(%s): Call to &{$^H{%s}} did not return a defined value

  Constant overloading that returns `undef` results in this error message. For numeric constants, it used to say ''Constant(undef)''. ''undef'' has been replaced with the number itself.

- The error produced when a module cannot be loaded now includes a hint that the module may need to be installed: ''Can't locate hopping.pm in `@INC` (you may need to install the hopping module) (@INC contains: ...)''

- vector argument not supported with alpha versions

  This warning was not suppressible, even with `no warnings`. Now it is suppressible, and has been moved from the ''internal'' category to the ''printf'' category.

- `Can't do {n,m} with n > m in regex; marked by <-- HERE in m/%s/`

  This fatal error has been turned into a warning that reads:

  Quantifier {n,m} with n > m can't match in regex

  (W regexp) Minima should be less than or equal to maxima. If you really want your regexp to match something 0 times, just put {0}.

- The ''Runaway prototype'' warning that occurs in bizarre cases has been removed as being unhelpful and inconsistent.

- The ‘‘Not a format reference’’ error has been removed, as the only case in which it could be triggered was a bug.
- The ‘‘Unable to create sub named %s’’ error has been removed for the same reason.
- The ’Can’t use ‘‘my %s’’ in sort comparison’ error has been downgraded to a warning, ’‘‘my %s’’ used in sort comparison’ (with ’state’ instead of ’my’ for state variables). In addition, the heuristics for guessing whether lexical $a or $b has been misused have been improved to generate fewer false positives. Lexical $a and $b are no longer disallowed if they are outside the sort block. Also, a named unary or list operator inside the sort block no longer causes the $a or $b to be ignored [perl #86136].

## Utility Changes

*h2xs*

- *h2xs* no longer produces invalid code for empty defines. [perl #20636]

## Configuration and Compilation

- Added useversionedarchname option to Configure

  When set, it includes ’api_versionstring’ in ’archname’. E.g. x86_64−linux−5.13.6−thread−multi. It is unset by default.

  This feature was requested by Tim Bunce, who observed that INSTALL_BASE creates a library structure that does not differentiate by perl version. Instead, it places architecture specific files in ‘‘$install_base/lib/perl5/$archname’’. This makes it difficult to use a common INSTALL_BASE library path with multiple versions of perl.

  By setting −Duseversionedarchname, the $archname will be distinct for architecture *and* API version, allowing mixed use of INSTALL_BASE.

- Add a PERL_NO_INLINE_FUNCTIONS option

  If PERL_NO_INLINE_FUNCTIONS is defined, don’t include ‘‘inline.h’’

  This permits test code to include the perl headers for definitions without creating a link dependency on the perl library (which may not exist yet).

- Configure will honour the external MAILDOMAIN environment variable, if set.
- installman no longer ignores the silent option
- Both META.yml and META.json files are now included in the distribution.
- *Configure* will now correctly detect isblank() when compiling with a C++ compiler.
- The pager detection in *Configure* has been improved to allow responses which specify options after the program name, e.g. **/usr/bin/less −R**, if the user accepts the default value. This helps **perldoc** when handling ANSI escapes [perl #72156].

## Testing

- The test suite now has a section for tests that require very large amounts of memory. These tests won’t run by default; they can be enabled by setting the PERL_TEST_MEMORY environment variable to the number of gibibytes of memory that may be safely used.

## Platform Support

### Discontinued Platforms

BeOS

BeOS was an operating system for personal computers developed by Be Inc, initially for their BeBox hardware. The OS Haiku was written as an open source replacement for/continuation of BeOS, and its perl port is current and actively maintained.

UTS Global

Support code relating to UTS global has been removed. UTS was a mainframe version of System V created by Amdahl, subsequently sold to UTS Global. The port has not been touched since before Perl v5.8.0, and UTS Global is now defunct.

VM/ESA

Support for VM/ESA has been removed. The port was tested on 2.3.0, which IBM ended service on in March 2002. 2.4.0 ended service in June 2003, and was superseded by Z/VM. The current

version of Z/VM is V6.2.0, and scheduled for end of service on 2015/04/30.

MPE/IX
: Support for MPE/IX has been removed.

EPOC
: Support code relating to EPOC has been removed. EPOC was a family of operating systems developed by Psion for mobile devices. It was the predecessor of Symbian. The port was last updated in April 2002.

Rhapsody
: Support for Rhapsody has been removed.

### Platform-Specific Notes

*AIX*

Configure now always adds `-qlanglvl=extc99` to the CC flags on AIX when using xlC. This will make it easier to compile a number of XS-based modules that assume C99 [perl #113778].

*clang++*

There is now a workaround for a compiler bug that prevented compiling with clang++ since Perl v5.15.7 [perl #112786].

*C++*

When compiling the Perl core as C++ (which is only semi-supported), the mathom functions are now compiled as `extern "C"`, to ensure proper binary compatibility. (However, binary compatibility isn't generally guaranteed anyway in the situations where this would matter.)

*Darwin*

Stop hardcoding an alignment on 8 byte boundaries to fix builds using -Dusemorebits.

*Haiku*

Perl should now work out of the box on Haiku R1 Alpha 4.

*MidnightBSD*

`libc_r` was removed from recent versions of MidnightBSD and older versions work better with `pthread`. Threading is now enabled using `pthread` which corrects build errors with threading enabled on 0.4-CURRENT.

*Solaris*

In Configure, avoid running sed commands with flags not supported on Solaris.

*VMS*

- Where possible, the case of filenames and command-line arguments is now preserved by enabling the CRTL features `DECC$EFS_CASE_PRESERVE` and `DECC$ARGV_PARSE_STYLE` at start-up time. The latter only takes effect when extended parse is enabled in the process from which Perl is run.
- The character set for Extended Filename Syntax (EFS) is now enabled by default on VMS. Among other things, this provides better handling of dots in directory names, multiple dots in filenames, and spaces in filenames. To obtain the old behavior, set the logical name `DECC$EFS_CHARSET` to `DISABLE`.
- Fixed linking on builds configured with `-Dusemymalloc=y`.
- Experimental support for building Perl with the HP C++ compiler is available by configuring with `-Dusecxx`.
- All C header files from the top-level directory of the distribution are now installed on VMS, providing consistency with a long-standing practice on other platforms. Previously only a subset were installed, which broke non-core extension builds for extensions that depended on the missing include files.
- Quotes are now removed from the command verb (but not the parameters) for commands spawned via `system`, backticks, or a piped `open`. Previously, quotes on the verb were passed through to DCL, which would fail to recognize the command. Also, if the verb is actually a path to an image or command procedure on an ODS-5 volume, quoting it now allows the path to contain spaces.

- The **a2p** build has been fixed for the HP C++ compiler on OpenVMS.

*Win32*

- Perl can now be built using Microsoft's Visual C++ 2012 compiler by specifying CCTYPE=MSVC110 (or MSVC110FREE if you are using the free Express edition for Windows Desktop) in *win32/Makefile*.
- The option to build without `USE_SOCKETS_AS_HANDLES` has been removed.
- Fixed a problem where perl could crash while cleaning up threads (including the main thread) in threaded debugging builds on Win32 and possibly other platforms [perl #114496].
- A rare race condition that would lead to sleep taking more time than requested, and possibly even hanging, has been fixed [perl #33096].
- `link` on Win32 now attempts to set `$!` to more appropriate values based on the Win32 API error code. [perl #112272]

  Perl no longer mangles the environment block, e.g. when launching a new sub-process, when the environment contains non-ASCII characters. Known problems still remain, however, when the environment contains characters outside of the current ANSI codepage (e.g. see the item about Unicode in `%ENV` in <http://perl5.git.perl.org/perl.git/blob/HEAD:/Porting/todo.pod>). [perl #113536]
- Building perl with some Windows compilers used to fail due to a problem with miniperl's `glob` operator (which uses the `perlglob` program) deleting the PATH environment variable [perl #113798].
- A new makefile option, `USE_64_BIT_INT`, has been added to the Windows makefiles. Set this to ''define'' when building a 32−bit perl if you want it to use 64−bit integers.

  Machine code size reductions, already made to the DLLs of XS modules in Perl v5.17.2, have now been extended to the perl DLL itself.

  Building with VC++ 6.0 was inadvertently broken in Perl v5.17.2 but has now been fixed again.

*WinCE*

Building on WinCE is now possible once again, although more work is required to fully restore a clean build.

## Internal Changes

- Synonyms for the misleadingly named `av_len()` have been created: `av_top_index()` and `av_tindex`. All three of these return the number of the highest index in the array, not the number of elements it contains.
- **SvUPGRADE()** is no longer an expression. Originally this macro (and its underlying function, **sv_upgrade()**) were documented as boolean, although in reality they always croaked on error and never returned false. In 2005 the documentation was updated to specify a void return value, but **SvUPGRADE()** was left always returning 1 for backwards compatibility. This has now been removed, and **SvUPGRADE()** is now a statement with no return value.

  So this is now a syntax error:

  ```
  if (!SvUPGRADE(sv)) { croak(...); }
  ```

  If you have code like that, simply replace it with

  ```
  SvUPGRADE(sv);
  ```

  or to avoid compiler warnings with older perls, possibly

  ```
  (void)SvUPGRADE(sv);
  ```
- Perl has a new copy-on-write mechanism that allows any SvPOK scalar to be upgraded to a copy-on-write scalar. A reference count on the string buffer is stored in the string buffer itself. This feature is **not enabled by default**.

  It can be enabled in a perl build by running *Configure* with **−Accflags=−DPERL_NEW_COPY_ON_WRITE**, and we would encourage XS authors to try

their code with such an enabled perl, and provide feedback. Unfortunately, there is not yet a good guide to updating XS code to cope with COW. Until such a document is available, consult the perl5−porters mailing list.

It breaks a few XS modules by allowing copy-on-write scalars to go through code paths that never encountered them before.

- Copy-on-write no longer uses the SvFAKE and SvREADONLY flags. Hence, SvREADONLY indicates a true read-only SV.

  Use the SvIsCOW macro (as before) to identify a copy-on-write scalar.

- `PL_glob_index` is gone.

- The private Perl_croak_no_modify has had its context parameter removed. It is now has a void prototype. Users of the public API croak_no_modify remain unaffected.

- Copy-on-write (shared hash key) scalars are no longer marked read-only. `SvREADONLY` returns false on such an SV, but `SvIsCOW` still returns true.

- A new op type, `OP_PADRANGE` has been introduced. The perl peephole optimiser will, where possible, substitute a single padrange op for a pushmark followed by one or more pad ops, and possibly also skipping list and nextstate ops. In addition, the op can carry out the tasks associated with the RHS of a `my(...) = @_` assignment, so those ops may be optimised away too.

- Case-insensitive matching inside a [bracketed] character class with a multi-character fold no longer excludes one of the possibilities in the circumstances that it used to. [perl #89774].

- `PL_formfeed` has been removed.

- The regular expression engine no longer reads one byte past the end of the target string. While for all internally well-formed scalars this should never have been a problem, this change facilitates clever tricks with string buffers in CPAN modules. [perl #73542]

- Inside a BEGIN block, `PL_compcv` now points to the currently-compiling subroutine, rather than the BEGIN block itself.

- `mg_length` has been deprecated.

- `sv_len` now always returns a byte count and `sv_len_utf8` a character count. Previously, `sv_len` and `sv_len_utf8` were both buggy and would sometimes returns bytes and sometimes characters. `sv_len_utf8` no longer assumes that its argument is in UTF−8. Neither of these creates UTF−8 caches for tied or overloaded values or for non-PVs any more.

- `sv_mortalcopy` now copies string buffers of shared hash key scalars when called from XS modules [perl #79824].

- The new `RXf_MODIFIES_VARS` flag can be set by custom regular expression engines to indicate that the execution of the regular expression may cause variables to be modified. This lets `s///` know to skip certain optimisations. Perl's own regular expression engine sets this flag for the special backtracking verbs that set `$REGMARK` and `$REGERROR`.

- The APIs for accessing lexical pads have changed considerably.

  `PADLIST`s are now longer AVs, but their own type instead. `PADLIST`s now contain a `PAD` and a `PADNAMELIST` of `PADNAME`s, rather than AVs for the pad and the list of pad names. `PAD`s, `PADNAMELIST`s, and `PADNAME`s are to be accessed as such through the newly added pad API instead of the plain AV and SV APIs. See perlapi for details.

- In the regex API, the numbered capture callbacks are passed an index indicating what match variable is being accessed. There are special index values for the `$``, `$&`, `$'` variables. Previously the same three values were used to retrieve `${^PREMATCH}`, `${^MATCH}`, `${^POSTMATCH}` too, but these have now been assigned three separate values. See ''Numbered capture callbacks'' in perlreapi.

- `PL_sawampersand` was previously a boolean indicating that any of `$``, `$&`, `$'` had been seen; it now contains three one-bit flags indicating the presence of each of the variables individually.

- The `CV *` typemap entry now supports `&{}` overloading and typeglobs, just like `&{...}` [perl #96872].
- The `SVf_AMAGIC` flag to indicate overloading is now on the stash, not the object. It is now set automatically whenever a method or `@ISA` changes, so its meaning has changed, too. It now means ''potentially overloaded''. When the overload table is calculated, the flag is automatically turned off if there is no overloading, so there should be no noticeable slowdown.

  The staleness of the overload tables is now checked when overload methods are invoked, rather than during `bless`.

  ''A'' magic is gone. The changes to the handling of the `SVf_AMAGIC` flag eliminate the need for it.

  `PL_amagic_generation` has been removed as no longer necessary. For XS modules, it is now a macro alias to `PL_na`.

  The fallback overload setting is now stored in a stash entry separate from overloadedness itself.
- The character-processing code has been cleaned up in places. The changes should be operationally invisible.
- The `study` function was made a no-op in v5.16. It was simply disabled via a `return` statement; the code was left in place. Now the code supporting what `study` used to do has been removed.
- Under threaded perls, there is no longer a separate PV allocated for every COP to store its package name (`cop->stashpv`). Instead, there is an offset (`cop->stashoff`) into the new `PL_stashpad` array, which holds stash pointers.
- In the pluggable regex API, the `regexp_engine` struct has acquired a new field `op_comp`, which is currently just for perl's internal use, and should be initialized to NULL by other regex plugin modules.
- A new function `alloccopstash` has been added to the API, but is considered experimental. See perlapi.
- Perl used to implement get magic in a way that would sometimes hide bugs in code that could call **mg_get()** too many times on magical values. This hiding of errors no longer occurs, so long-standing bugs may become visible now. If you see magic-related errors in XS code, check to make sure it, together with the Perl API functions it uses, calls **mg_get()** only once on **SvGMAGICAL()** values.
- OP allocation for CVs now uses a slab allocator. This simplifies memory management for OPs allocated to a CV, so cleaning up after a compilation error is simpler and safer [perl #111462][perl #112312].
- `PERL_DEBUG_READONLY_OPS` has been rewritten to work with the new slab allocator, allowing it to catch more violations than before.
- The old slab allocator for ops, which was only enabled for `PERL_IMPLICIT_SYS` and `PERL_DEBUG_READONLY_OPS`, has been retired.

## Selected Bug Fixes

- Here document terminators no longer require a terminating newline character when they occur at the end of a file. This was already the case at the end of a string eval [perl #65838].
- `-DPERL_GLOBAL_STRUCT` builds now free the global struct **after** they've finished using it.
- A trailing '/' on a path in `@INC` will no longer have an additional '/' appended.
- The `:crlf` layer now works when unread data doesn't fit into its own buffer. [perl #112244].
- `ungetc()` now handles UTF-8 encoded data. [perl #116322].
- A bug in the core typemap caused any C types that map to the T_BOOL core typemap entry to not be set, updated, or modified when the T_BOOL variable was used in an OUTPUT: section with an exception for RETVAL. T_BOOL in an INPUT: section was not affected. Using a T_BOOL return type for an XSUB (RETVAL) was not affected. A side effect of fixing this bug is, if a T_BOOL is specified in the OUTPUT: section (which previous did nothing to the SV), and a read only SV (literal) is passed to the XSUB, croaks like ''Modification of a read-only value attempted'' will

happen. [perl #115796]

- On many platforms, providing a directory name as the script name caused perl to do nothing and report success. It should now universally report an error and exit nonzero. [perl #61362]
- `sort {undef} ...` under fatal warnings no longer crashes. It had begun crashing in Perl v5.16.
- Stashes blessed into each other (`bless \%Foo::, 'Bar'; bless \%Bar::, 'Foo'`) no longer result in double frees. This bug started happening in Perl v5.16.
- Numerous memory leaks have been fixed, mostly involving fatal warnings and syntax errors.
- Some failed regular expression matches such as `'f' =~ /../g` were not resetting `pos`. Also, ''match-once'' patterns (`m?...?g`) failed to reset it, too, when invoked a second time [perl #23180].
- Several bugs involving `local *ISA` and `local *Foo::` causing stale MRO caches have been fixed.
- Defining a subroutine when its typeglob has been aliased no longer results in stale method caches. This bug was introduced in Perl v5.10.
- Localising a typeglob containing a subroutine when the typeglob's package has been deleted from its parent stash no longer produces an error. This bug was introduced in Perl v5.14.
- Under some circumstances, `local *method=...` would fail to reset method caches upon scope exit.
- `/[.foo.]/` is no longer an error, but produces a warning (as before) and is treated as `/[.fo]/` [perl #115818].
- `goto $tied_var` now calls FETCH before deciding what type of goto (subroutine or label) this is.
- Renaming packages through glob assignment (`*Foo:: = *Bar::; *Bar:: = *Baz::`) in combination with `m?...?` and `reset` no longer makes threaded builds crash.
- A number of bugs related to assigning a list to hash have been fixed. Many of these involve lists with repeated keys like `(1, 1, 1, 1)`.
  - The expression `scalar(%h = (1, 1, 1, 1))` now returns `4`, not `2`.
  - The return value of `%h = (1, 1, 1)` in list context was wrong. Previously this would return `(1, undef, 1)`, now it returns `(1, undef)`.
  - Perl now issues the same warning on `($s, %h) = (1, {})` as it does for `(%h) = ({})`, ''Reference found where even-sized list expected''.
  - A number of additional edge cases in list assignment to hashes were corrected. For more details see commit 23b7025ebc.
- Attributes applied to lexical variables no longer leak memory. [perl #114764]
- `dump`, `goto`, `last`, `next`, `redo` or `require` followed by a bareword (or version) and then an infix operator is no longer a syntax error. It used to be for those infix operators (like `+`) that have a different meaning where a term is expected. [perl #105924]
- `require a::b . 1` and `require a::b + 1` no longer produce erroneous ambiguity warnings. [perl #107002]
- Class method calls are now allowed on any string, and not just strings beginning with an alphanumeric character. [perl #105922]
- An empty pattern created with `qr//` used in `m///` no longer triggers the ''empty pattern reuses last pattern'' behaviour. [perl #96230]
- Tying a hash during iteration no longer results in a memory leak.
- Freeing a tied hash during iteration no longer results in a memory leak.
- List assignment to a tied array or hash that dies on STORE no longer results in a memory leak.

- If the hint hash (`%^H`) is tied, compile-time scope entry (which copies the hint hash) no longer leaks memory if FETCH dies. [perl #107000]
- Constant folding no longer inappropriately triggers the special `split " "` behaviour. [perl #94490]
- `defined scalar(@array)`, `defined do { &foo }`, and similar constructs now treat the argument to `defined` as a simple scalar. [perl #97466]
- Running a custom debugging that defines no `*DB::DB` glob or provides a subroutine stub for `&DB::DB` no longer results in a crash, but an error instead. [perl #114990]
- `reset ""` now matches its documentation. `reset` only resets `m?...?` patterns when called with no argument. An empty string for an argument now does nothing. (It used to be treated as no argument.) [perl #97958]
- `printf` with an argument returning an empty list no longer reads past the end of the stack, resulting in erratic behaviour. [perl #77094]
- `--subname` no longer produces erroneous ambiguity warnings. [perl #77240]
- `v10` is now allowed as a label or package name. This was inadvertently broken when v-strings were added in Perl v5.6. [perl #56880]
- `length`, `pos`, `substr` and `sprintf` could be confused by ties, overloading, references and typeglobs if the stringification of such changed the internal representation to or from UTF-8. [perl #114410]
- utf8::encode now calls FETCH and STORE on tied variables. utf8::decode now calls STORE (it was already calling FETCH).
- `$tied =~ s/$non_utf8/$utf8/` no longer loops infinitely if the tied variable returns a Latin-1 string, shared hash key scalar, or reference or typeglob that stringifies as ASCII or Latin-1. This was a regression from v5.12.
- `s///` without /e is now better at detecting when it needs to forego certain optimisations, fixing some buggy cases:
  - Match variables in certain constructs (&&, `||`, `..` and others) in the replacement part; e.g., `s/(.)/$l{$a||$1}/g`. [perl #26986]
  - Aliases to match variables in the replacement.
  - `$REGERROR` or `$REGMARK` in the replacement. [perl #49190]
  - An empty pattern (`s//$foo/`) that causes the last-successful pattern to be used, when that pattern contains code blocks that modify the variables in the replacement.
- The taintedness of the replacement string no longer affects the taintedness of the return value of `s///e`.
- The `$|` autoflush variable is created on-the-fly when needed. If this happened (e.g., if it was mentioned in a module or eval) when the currently-selected filehandle was a typeglob with an empty IO slot, it used to crash. [perl #115206]
- Line numbers at the end of a string eval are no longer off by one. [perl #114658]
- `@INC` filters (subroutines returned by subroutines in `@INC`) that set `$_` to a copy-on-write scalar no longer cause the parser to modify that string buffer in place.
- `length($object)` no longer returns the undefined value if the object has string overloading that returns undef. [perl #115260]
- The use of `PL_stashcache`, the stash name lookup cache for method calls, has been restored,

  Commit da6b625f78f5f133 in August 2011 inadvertently broke the code that looks up values in `PL_stashcache`. As it's only a cache, quite correctly everything carried on working without it.
- The error "Can't localize through a reference" had disappeared in v5.16.0 when `local %$ref` appeared on the last line of an lvalue subroutine. This error disappeared for `\local %$ref` in perl v5.8.1. It has now been restored.

- The parsing of here-docs has been improved significantly, fixing several parsing bugs and crashes and one memory leak, and correcting wrong subsequent line numbers under certain conditions.
- Inside an eval, the error message for an unterminated here-doc no longer has a newline in the middle of it [perl #70836].
- A substitution inside a substitution pattern (`s/${s|||}//`) no longer confuses the parser.
- It may be an odd place to allow comments, but `s//"" # hello/e` has always worked, *unless* there happens to be a null character before the first #. Now it works even in the presence of nulls.
- An invalid range in `tr///` or `y///` no longer results in a memory leak.
- String eval no longer treats a semicolon-delimited quote-like operator at the very end (`eval 'q;;'`) as a syntax error.
- `warn {$_ => 1} + 1` is no longer a syntax error. The parser used to get confused with certain list operators followed by an anonymous hash and then an infix operator that shares its form with a unary operator.
- `(caller $n)[6]` (which gives the text of the eval) used to return the actual parser buffer. Modifying it could result in crashes. Now it always returns a copy. The string returned no longer has "\n;" tacked on to the end. The returned text also includes here-doc bodies, which used to be omitted.
- The UTF-8 position cache is now reset when accessing magical variables, to avoid the string buffer and the UTF-8 position cache getting out of sync [perl #114410].
- Various cases of get magic being called twice for magical UTF-8 strings have been fixed.
- This code (when not in the presence of `$&` etc)

  ```
  $_ = 'x' x 1_000_000;
  1 while /(.)/;
  ```

  used to skip the buffer copy for performance reasons, but suffered from `$1` etc changing if the original string changed. That's now been fixed.
- Perl doesn't use PerlIO anymore to report out of memory messages, as PerlIO might attempt to allocate more memory.
- In a regular expression, if something is quantified with `{n,m}` where `n > m`, it can't possibly match. Previously this was a fatal error, but now is merely a warning (and that something won't match). [perl #82954].
- It used to be possible for formats defined in subroutines that have subsequently been undefined and redefined to close over variables in the wrong pad (the newly-defined enclosing sub), resulting in crashes or "Bizarre copy" errors.
- Redefinition of XSUBs at run time could produce warnings with the wrong line number.
- The `%vd` sprintf format does not support version objects for alpha versions. It used to output the format itself (%vd) when passed an alpha version, and also emit an "Invalid conversion in printf" warning. It no longer does, but produces the empty string in the output. It also no longer leaks memory in this case.
- `$obj->SUPER::method` calls in the main package could fail if the SUPER package had already been accessed by other means.
- Stash aliasing (`*foo:: = *bar::`) no longer causes SUPER calls to ignore changes to methods or `@ISA` or use the wrong package.
- Method calls on packages whose names end in ::SUPER are no longer treated as SUPER method calls, resulting in failure to find the method. Furthermore, defining subroutines in such packages no longer causes them to be found by SUPER method calls on the containing package [perl #114924].
- `\w` now matches the code points U+200C (ZERO WIDTH NON-JOINER) and U+200D (ZERO WIDTH JOINER). `\W` no longer matches these. This change is because Unicode corrected their definition of what `\w` should match.

- `dump LABEL` no longer leaks its label.
- Constant folding no longer changes the behaviour of functions like `stat()` and `truncate()` that can take either filenames or handles. `stat 1 ? foo : bar` nows treats its argument as a file name (since it is an arbitrary expression), rather than the handle ‘‘foo’’.
- `truncate FOO, $len` no longer falls back to treating ‘‘FOO’’ as a file name if the filehandle has been deleted. This was broken in Perl v5.16.0.
- Subroutine redefinitions after sub-to-glob and glob-to-glob assignments no longer cause double frees or panic messages.
- `s///` now turns vstrings into plain strings when performing a substitution, even if the resulting string is the same (`s/a/a/`).
- Prototype mismatch warnings no longer erroneously treat constant subs as having no prototype when they actually have "".
- Constant subroutines and forward declarations no longer prevent prototype mismatch warnings from omitting the sub name.
- `undef` on a subroutine now clears call checkers.
- The `ref` operator started leaking memory on blessed objects in Perl v5.16.0. This has been fixed [perl #114340].
- `use` no longer tries to parse its arguments as a statement, making `use constant { () };` a syntax error [perl #114222].
- On debugging builds, ‘‘uninitialized’’ warnings inside formats no longer cause assertion failures.
- On debugging builds, subroutines nested inside formats no longer cause assertion failures [perl #78550].
- Formats and `use` statements are now permitted inside formats.
- `print $x` and `sub { print $x }->()` now always produce the same output. It was possible for the latter to refuse to close over `$x` if the variable was not active; e.g., if it was defined outside a currently-running named subroutine.
- Similarly, `print $x` and `print eval '$x'` now produce the same output. This also allows ‘‘my `$x` if 0’’ variables to be seen in the debugger [perl #114018].
- Formats called recursively no longer stomp on their own lexical variables, but each recursive call has its own set of lexicals.
- Attempting to free an active format or the handle associated with it no longer results in a crash.
- Format parsing no longer gets confused by braces, semicolons and low-precedence operators. It used to be possible to use braces as format delimiters (instead of `=` and `.`), but only sometimes. Semicolons and low-precedence operators in format argument lines no longer confuse the parser into ignoring the line’s return value. In format argument lines, braces can now be used for anonymous hashes, instead of being treated always as `do` blocks.
- Formats can now be nested inside code blocks in regular expressions and other quoted constructs (`/(?{...})/` and `qq/${...}/`) [perl #114040].
- Formats are no longer created after compilation errors.
- Under debugging builds, the **-DA** command line option started crashing in Perl v5.16.0. It has been fixed [perl #114368].
- A potential deadlock scenario involving the premature termination of a pseudo-forked child in a Windows build with ithreads enabled has been fixed. This resolves the common problem of the *t/op/fork.t* test hanging on Windows [perl #88840].
- The code which generates errors from `require()` could potentially read one or two bytes before the start of the filename for filenames less than three bytes long and ending `/\.p?\z/`. This has now been fixed. Note that it could never have happened with module names given to `use()` or `require()` anyway.

- The handling of pathnames of modules given to `require()` has been made thread-safe on VMS.
- Non-blocking sockets have been fixed on VMS.
- Pod can now be nested in code inside a quoted construct outside of a string eval. This used to work only within string evals [perl #114040].
- `goto ''` now looks for an empty label, producing the ‘‘goto must have label’’ error message, instead of exiting the program [perl #111794].
- `goto "\0"` now dies with ‘‘Can’t find label’’ instead of ‘‘goto must have label’’.
- The C function `hv_store` used to result in crashes when used on `%^H` [perl #111000].
- A call checker attached to a closure prototype via `cv_set_call_checker` is now copied to closures cloned from it. So `cv_set_call_checker` now works inside an attribute handler for a closure.
- Writing to `$^N` used to have no effect. Now it croaks with ‘‘Modification of a read-only value’’ by default, but that can be overridden by a custom regular expression engine, as with `$1` [perl #112184].
- `undef` on a control character glob (`undef *^H`) no longer emits an erroneous warning about ambiguity [perl #112456].
- For efficiency’s sake, many operators and built-in functions return the same scalar each time. Lvalue subroutines and subroutines in the CORE:: namespace were allowing this implementation detail to leak through. `print &CORE::uc("a"), &CORE::uc("b")` used to print ‘‘BB’’. The same thing would happen with an lvalue subroutine returning the return value of `uc`. Now the value is copied in such cases.
- `method {}` syntax with an empty block or a block returning an empty list used to crash or use some random value left on the stack as its invocant. Now it produces an error.
- `vec` now works with extremely large offsets (>2 GB) [perl #111730].
- Changes to overload settings now take effect immediately, as do changes to inheritance that affect overloading. They used to take effect only after `bless`.

  Objects that were created before a class had any overloading used to remain non-overloaded even if the class gained overloading through `use overload` or `@ISA` changes, and even after `bless`. This has been fixed [perl #112708].
- Classes with overloading can now inherit fallback values.
- Overloading was not respecting a fallback value of 0 if there were overloaded objects on both sides of an assignment operator like `+=` [perl #111856].
- `pos` now croaks with hash and array arguments, instead of producing erroneous warnings.
- `while(each %h)` now implies `while(defined($_ = each %h))`, like `readline` and `readdir`.
- Subs in the CORE:: namespace no longer crash after `undef *_` when called with no argument list (`&CORE::time` with no parentheses).
- `unpack` no longer produces the ‘‘’/’ must follow a numeric type in unpack’’ error when it is the data that are at fault [perl #60204].
- `join` and `"@array"` now call FETCH only once on a tied `$"` [perl #8931].
- Some subroutine calls generated by compiling core ops affected by a `CORE::GLOBAL` override had op checking performed twice. The checking is always idempotent for pure Perl code, but the double checking can matter when custom call checkers are involved.
- A race condition used to exist around fork that could cause a signal sent to the parent to be handled by both parent and child. Signals are now blocked briefly around fork to prevent this from happening [perl #82580].
- The implementation of code blocks in regular expressions, such as `(?{})` and `(??{})`, has been heavily reworked to eliminate a whole slew of bugs. The main user-visible changes are:

- Code blocks within patterns are now parsed in the same pass as the surrounding code; in particular it is no longer necessary to have balanced braces: this now works:

```
/(?{  $x='{'  })/
```

  This means that this error message is no longer generated:

```
Sequence (?{...}) not terminated or not {}-balanced in regex
```

  but a new error may be seen:

```
Sequence (?{...}) not terminated with ')'
```

  In addition, literal code blocks within run-time patterns are only compiled once, at perl compile-time:

```
for my $p (...) {
    # this 'FOO' block of code is compiled once,
    # at the same time as the surrounding 'for' loop
    /$p{(?{FOO;})/;
}
```

- Lexical variables are now sane as regards scope, recursion and closure behavior. In particular, `/A(?{B})C/` behaves (from a closure viewpoint) exactly like `/A/ && do { B } && /C/`, while `qr/A(?{B})C/` is like `sub {/A/ && do { B } && /C/}`. So this code now works how you might expect, creating three regexes that match 0, 1, and 2:

```
for my $i (0..2) {
    push @r, qr/^(??{$i})$/;
}
"1" =~ $r[1]; # matches
```

- The `use re 'eval'` pragma is now only required for code blocks defined at runtime; in particular in the following, the text of the `$r` pattern is still interpolated into the new pattern and recompiled, but the individual compiled code-blocks within `$r` are reused rather than being recompiled, and `use re 'eval'` isn't needed any more:

```
my $r = qr/abc(?{....})def/;
/xyz$r/;
```

- Flow control operators no longer crash. Each code block runs in a new dynamic scope, so `next` etc. will not see any enclosing loops. `return` returns a value from the code block, not from any enclosing subroutine.

- Perl normally caches the compilation of run-time patterns, and doesn't recompile if the pattern hasn't changed, but this is now disabled if required for the correct behavior of closures. For example:

```
my $code = '(??{$x})';
for my $x (1..3) {
    # recompile to see fresh value of $x each time
    $x =~ /$code/;
}
```

- The `/msix` and `(?msix)` etc. flags are now propagated into the return value from `(??{})`; this now works:

```
"AB" =~ /a(??{'b'})/i;
```

- Warnings and errors will appear to come from the surrounding code (or for run-time code blocks, from an eval) rather than from an `re_eval`:

```
use re 'eval'; $c = '(?{ warn "foo" })'; /$c/;
/(?{ warn "foo" })/;
```

  formerly gave:

```
foo at (re_eval 1) line 1.
foo at (re_eval 2) line 1.
```

and now gives:

```
foo at (eval 1) line 1.
foo at /some/prog line 2.
```

- Perl now can be recompiled to use any Unicode version. In v5.16, it worked on Unicodes 6.0 and 6.1, but there were various bugs if earlier releases were used; the older the release the more problems.
- `vec` no longer produces ‘‘uninitialized’’ warnings in lvalue context [perl #9423].
- An optimization involving fixed strings in regular expressions could cause a severe performance penalty in edge cases. This has been fixed [perl #76546].
- In certain cases, including empty subpatterns within a regular expression (such as `(?:)` or `(?:|)`) could disable some optimizations. This has been fixed.
- The ‘‘Can’t find an opnumber’’ message that `prototype` produces when passed a string like ‘‘CORE::nonexistent_keyword’’ now passes UTF-8 and embedded NULs through unchanged [perl #97478].
- `prototype` now treats magical variables like `$1` the same way as non-magical variables when checking for the CORE:: prefix, instead of treating them as subroutine names.
- Under threaded perls, a runtime code block in a regular expression could corrupt the package name stored in the op tree, resulting in bad reads in `caller`, and possibly crashes [perl #113060].
- Referencing a closure prototype (`\&{$_[1]}` in an attribute handler for a closure) no longer results in a copy of the subroutine (or assertion failures on debugging builds).
- `eval '__PACKAGE__'` now returns the right answer on threaded builds if the current package has been assigned over (as in `*ThisPackage:: = *ThatPackage::`) [perl #78742].
- If a package is deleted by code that it calls, it is possible for `caller` to see a stack frame belonging to that deleted package. `caller` could crash if the stash’s memory address was reused for a scalar and a substitution was performed on the same scalar [perl #113486].
- `UNIVERSAL::can` no longer treats its first argument differently depending on whether it is a string or number internally.
- `open` with `<&` for the mode checks to see whether the third argument is a number, in determining whether to treat it as a file descriptor or a handle name. Magical variables like `$1` were always failing the numeric check and being treated as handle names.
- `warn`’s handling of magical variables (`$1`, ties) has undergone several fixes. `FETCH` is only called once now on a tied argument or a tied `$@` [perl #97480]. Tied variables returning objects that stringify as "" are no longer ignored. A tied `$@` that happened to return a reference the *previous* time it was used is no longer ignored.
- `warn ""` now treats `$@` with a number in it the same way, regardless of whether it happened via `$@=3` or `$@="3"`. It used to ignore the former. Now it appends ‘‘\t...caught’’, as it has always done with `$@="3"`.
- Numeric operators on magical variables (e.g., `$1 + 1`) used to use floating point operations even where integer operations were more appropriate, resulting in loss of accuracy on 64-bit platforms [perl #109542].
- Unary negation no longer treats a string as a number if the string happened to be used as a number at some point. So, if `$x` contains the string ‘‘dogs’’, `-$x` returns ‘‘-dogs’’ even if `$y=0+$x` has happened at some point.
- In Perl v5.14, `-'-10'` was fixed to return ‘‘10’’, not ‘‘+10’’. But magical variables (`$1`, ties) were not fixed till now [perl #57706].
- Unary negation now treats strings consistently, regardless of the internal `UTF8` flag.

- A regression introduced in Perl v5.16.0 involving `tr/SEARCHLIST/REPLACEMENTLIST/` has been fixed. Only the first instance is supposed to be meaningful if a character appears more than once in *SEARCHLIST*. Under some circumstances, the final instance was overriding all earlier ones. [perl #113584]
- Regular expressions like `qr/\87/` previously silently inserted a NUL character, thus matching as if it had been written `qr/\00087/`. Now it matches as if it had been written as `qr/87/`, with a message that the sequence `"\8"` is unrecognized.
- `__SUB__` now works in special blocks (`BEGIN`, `END`, etc.).
- Thread creation on Windows could theoretically result in a crash if done inside a `BEGIN` block. It still does not work properly, but it no longer crashes [perl #111610].
- `\&{''}` (with the empty string) now autovivifies a stub like any other sub name, and no longer produces the ‘‘Unable to create sub’’ error [perl #94476].
- A regression introduced in v5.14.0 has been fixed, in which some calls to the `re` module would clobber `$_` [perl #113750].
- `do FILE` now always either sets or clears `$@`, even when the file can’t be read. This ensures that testing `$@` first (as recommended by the documentation) always returns the correct result.
- The array iterator used for the `each @array` construct is now correctly reset when `@array` is cleared [perl #75596]. This happens, for example, when the array is globally assigned to, as in `@array = (...)`, but not when its **values** are assigned to. In terms of the XS API, it means that `av_clear()` will now reset the iterator.

  This mirrors the behaviour of the hash iterator when the hash is cleared.
- `$class->can`, `$class->isa`, and `$class->DOES` now return correct results, regardless of whether that package referred to by `$class` exists [perl #47113].
- Arriving signals no longer clear `$@` [perl #45173].
- Allow `my ()` declarations with an empty variable list [perl #113554].
- During parsing, subs declared after errors no longer leave stubs [perl #113712].
- Closures containing no string evals no longer hang on to their containing subroutines, allowing variables closed over by outer subroutines to be freed when the outer sub is freed, even if the inner sub still exists [perl #89544].
- Duplication of in-memory filehandles by opening with a ‘‘<&=’’ or ‘‘>&=’’ mode stopped working properly in v5.16.0. It was causing the new handle to reference a different scalar variable. This has been fixed [perl #113764].
- `qr//` expressions no longer crash with custom regular expression engines that do not set `offs` at regular expression compilation time [perl #112962].
- `delete local` no longer crashes with certain magical arrays and hashes [perl #112966].
- `local` on elements of certain magical arrays and hashes used not to arrange to have the element deleted on scope exit, even if the element did not exist before `local`.
- `scalar(write)` no longer returns multiple items [perl #73690].
- String to floating point conversions no longer misparse certain strings under `use locale` [perl #109318].
- `@INC` filters that die no longer leak memory [perl #92252].
- The implementations of overloaded operations are now called in the correct context. This allows, among other things, being able to properly override `<>` [perl #47119].
- Specifying only the `fallback` key when calling `use overload` now behaves properly [perl #113010].
- `sub foo { my $a = 0; while ($a) { ... } }` and `sub foo { while (0) { ... } }` now return the same thing [perl #73618].

- String negation now behaves the same under `use integer;` as it does without [perl #113012].
- `chr` now returns the Unicode replacement character (U+FFFD) for −1, regardless of the internal representation. −1 used to wrap if the argument was tied or a string internally.
- Using a `format` after its enclosing sub was freed could crash as of perl v5.12.0, if the format referenced lexical variables from the outer sub.
- Using a `format` after its enclosing sub was undefined could crash as of perl v5.10.0, if the format referenced lexical variables from the outer sub.
- Using a `format` defined inside a closure, which format references lexical variables from outside, never really worked unless the `write` call was directly inside the closure. In v5.10.0 it even started crashing. Now the copy of that closure nearest the top of the call stack is used to find those variables.
- Formats that close over variables in special blocks no longer crash if a stub exists with the same name as the special block before the special block is compiled.
- The parser no longer gets confused, treating `eval foo ()` as a syntax error if preceded by `print;` [perl #16249].
- The return value of `syscall` is no longer truncated on 64−bit platforms [perl #113980].
- Constant folding no longer causes `print 1 ? FOO : BAR` to print to the FOO handle [perl #78064].
- `do subname` now calls the named subroutine and uses the file name it returns, instead of opening a file named ‘‘subname’’.
- Subroutines looked up by rv2cv check hooks (registered by XS modules) are now taken into consideration when determining whether `foo bar` should be the sub call `foo(bar)` or the method call `"bar"->foo`.
- `CORE::foo::bar` is no longer treated specially, allowing global overrides to be called directly via `CORE::GLOBAL::uc(...)` [perl #113016].
- Calling an undefined sub whose typeglob has been undefined now produces the customary ‘‘Undefined subroutine called’’ error, instead of ‘‘Not a CODE reference’’.
- Two bugs involving `@ISA` have been fixed. `*ISA = *glob_without_array` and `undef *ISA; @{*ISA}` would prevent future modifications to `@ISA` from updating the internal caches used to look up methods. The *glob_without_array case was a regression from Perl v5.12.
- Regular expression optimisations sometimes caused `$` with `/m` to produce failed or incorrect matches [perl #114068].
- `__SUB__` now works in a `sort` block when the enclosing subroutine is predeclared with `sub foo;` syntax [perl #113710].
- Unicode properties only apply to Unicode code points, which leads to some subtleties when regular expressions are matched against above-Unicode code points. There is a warning generated to draw your attention to this. However, this warning was being generated inappropriately in some cases, such as when a program was being parsed. Non-Unicode matches such as `\w` and `[:word:]` should not generate the warning, as their definitions don’t limit them to apply to only Unicode code points. Now the message is only generated when matching against `\p{}` and `\P{}`. There remains a bug, [perl #114148], for the very few properties in Unicode that match just a single code point. The warning is not generated if they are matched against an above-Unicode code point.
- Uninitialized warnings mentioning hash elements would only mention the element name if it was not in the first bucket of the hash, due to an off-by-one error.
- A regular expression optimizer bug could cause multiline ‘‘^’’ to behave incorrectly in the presence of line breaks, such that `"/\n\n" =~ m#\A(?:^/$)#im` would not match [perl #115242].
- Failed `fork` in list context no longer corrupts the stack. `@a = (1, 2, fork, 3)` used to gobble up the 2 and assign `(1, undef, 3)` if the `fork` call failed.

- Numerous memory leaks have been fixed, mostly involving tied variables that die, regular expression character classes and code blocks, and syntax errors.
- Assigning a regular expression (`${qr//}`) to a variable that happens to hold a floating point number no longer causes assertion failures on debugging builds.
- Assigning a regular expression to a scalar containing a number no longer causes subsequent numification to produce random numbers.
- Assigning a regular expression to a magic variable no longer wipes away the magic. This was a regression from v5.10.
- Assigning a regular expression to a blessed scalar no longer results in crashes. This was also a regression from v5.10.
- Regular expression can now be assigned to tied hash and array elements with flattening into strings.
- Numifying a regular expression no longer results in an uninitialized warning.
- Negative array indices no longer cause EXISTS methods of tied variables to be ignored. This was a regression from v5.12.
- Negative array indices no longer result in crashes on arrays tied to non-objects.
- `$byte_overload .= $utf8` no longer results in doubly-encoded UTF-8 if the left-hand scalar happened to have produced a UTF-8 string the last time overloading was invoked.
- `goto &sub` now uses the current value of `@_`, instead of using the array the subroutine was originally called with. This means `local @_ = (...); goto &sub` now works [perl #43077].
- If a debugger is invoked recursively, it no longer stomps on its own lexical variables. Formerly under recursion all calls would share the same set of lexical variables [perl #115742].
- `*_{ARRAY}` returned from a subroutine no longer spontaneously becomes empty.
- When using `say` to print to a tied filehandle, the value of `$\` is correctly localized, even if it was previously undef. [perl #119927]

## Known Problems

- UTF8-flagged strings in `%ENV` on HP-UX 11.00 are buggy

  The interaction of UTF8-flagged strings and `%ENV` on HP-UX 11.00 is currently dodgy in some not-yet-fully-diagnosed way. Expect test failures in *t/op/magic.t*, followed by unknown behavior when storing wide characters in the environment.

## Obituary

Hojung Yoon (AMORETTE), 24, of Seoul, South Korea, went to his long rest on May 8, 2013 with llama figurine and autographed TIMTOADY card. He was a brilliant young Perl 5 & 6 hacker and a devoted member of Seoul.pm. He programmed Perl, talked Perl, ate Perl, and loved Perl. We believe that he is still programming in Perl with his broken IBM laptop somewhere. He will be missed.

## Acknowledgements

Perl v5.18.0 represents approximately 12 months of development since Perl v5.16.0 and contains approximately 400,000 lines of changes across 2,100 files from 113 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl v5.18.0:

Aaron Crane, Aaron Trevena, Abhijit Menon-Sen, Adrian M. Enache, Alan Haggai Alavi, Alexandr Ciornii, Andrew Tam, Andy Dougherty, Anton Nikishaev, Aristotle Pagaltzis, Augustina Blair, Bob Ernst, Brad Gilbert, Breno G. de Oliveira, Brian Carlson, Brian Fraser, Charlie Gonzalez, Chip Salzenberg, Chris 'BinGOs' Williams, Christian Hansen, Colin Kuskie, Craig A. Berry, Dagfinn Ilmari Mannsåker, Daniel Dragan, Daniel Perrett, Darin McBride, Dave Rolsky, David Golden, David Leadbeater, David Mitchell, David Nicol, Dominic Hargreaves, E. Choroba, Eric Brine, Evan Miller, Father Chrysostomos, Florian Ragwitz, François Perrad, George Greer, Goro Fuji, H.Merijn Brand, Herbert Breunung, Hugo van der Sanden, Igor Zaytsev, James E Keenan, Jan Dubois, Jasmine Ahuja, Jerry D. Hedden, Jess Robinson, Jesse Luehrs, Joaquin Ferrero, Joel Berger, John Goodyear, John Peacock, Karen Etheridge, Karl Williamson, Karthik Rajagopalan, Kent Fredric, Leon Timmermans,

Lucas Holt, Lukas Mai, Marcus Holland-Moritz, Markus Jansen, Martin Hasch, Matthew Horsfall, Max Maischein, Michael G Schwern, Michael Schroeder, Moritz Lenz, Nicholas Clark, Niko Tyni, Oleg Nesterov, Patrik Hägglund, Paul Green, Paul Johnson, Paul Marquess, Peter Martini, Rafael Garcia-Suarez, Reini Urban, Renee Baecker, Rhesa Rozendaal, Ricardo Signes, Robin Barker, Ronald J. Kimball, Ruslan Zakirov, Salvador Fandiño, Sawyer X, Scott Lanning, Sergey Alekseev, Shawn M Moore, Shirakata Kentaro, Shlomi Fish, Sisyphus, Smylers, Steffen Müller, Steve Hay, Steve Peters, Steven Schubiger, Sullivan Beck, Sven Strickroth, Sébastien Aperghis-Tramoni, Thomas Sibley, Tobias Leich, Tom Wyant, Tony Cook, Vadim Konovalov, Vincent Pit, Volker Schatz, Walt Mankowski, Yves Orton, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5181delta − what is new for perl v5.18.1

## DESCRIPTION

This document describes differences between the 5.18.0 release and the 5.18.1 release.

If you are upgrading from an earlier release such as 5.16.0, first read perl5180delta, which describes differences between 5.16.0 and 5.18.0.

## Incompatible Changes

There are no changes intentionally incompatible with 5.18.0 If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- B has been upgraded from 1.42 to 1.42_01, fixing bugs related to lexical subroutines.
- Digest::SHA has been upgraded from 5.84 to 5.84_01, fixing a crashing bug. [RT #118649]
- Module::CoreList has been upgraded from 2.89 to 2.96.

## Platform Support

### Platform-Specific Notes

AIX

A rarely-encountered configuration bug in the AIX hints file has been corrected.

MidnightBSD

After a patch to the relevant hints file, perl should now build correctly on MidnightBSD 0.4−RELEASE.

## Selected Bug Fixes

- Starting in v5.18.0, a construct like `/[#](?{})/x` would have its # incorrectly interpreted as a comment. The code block would be skipped, unparsed. This has been corrected.
- A number of memory leaks related to the new, experimental regexp bracketed character class feature have been plugged.
- The OP allocation code now returns correctly aligned memory in all cases for `struct pmop`. Previously it could return memory only aligned to a 4−byte boundary, which is not correct for an ithreads build with 64 bit IVs on some 32 bit platforms. Notably, this caused the build to fail completely on sparc GNU/Linux. [RT #118055]
- The debugger’s `man` command been fixed. It was broken in the v5.18.0 release. The `man` command is aliased to the names `doc` and `perldoc` − all now work again.
- `@_` is now correctly visible in the debugger, fixing a regression introduced in v5.18.0’s debugger. [RT #118169]
- Fixed a small number of regexp constructions that could either fail to match or crash perl when the string being matched against was allocated above the 2GB line on 32−bit systems. [RT #118175]
- Perl v5.16 inadvertently introduced a bug whereby calls to XSUBs that were not visible at compile time were treated as lvalues and could be assigned to, even when the subroutine was not an lvalue sub. This has been fixed. [perl #117947]
- Perl v5.18 inadvertently introduced a bug whereby dual-vars (i.e. variables with both string and numeric values, such as `$!` ) where the truthness of the variable was determined by the numeric value rather than the string value. [RT #118159]
- Perl v5.18 inadvertently introduced a bug whereby interpolating mixed up− and down-graded UTF−8 strings in a regex could result in malformed UTF−8 in the pattern: specifically if a downgraded character in the range `\x80..\xff` followed a UTF−8 string, e.g.

```
utf8::upgrade(  my $u = "\x{e5}");
utf8::downgrade(my $d = "\x{e5}");
/$u$d/
```

[perl #118297].

- Lexical constants (`my sub a() { 42 }`) no longer crash when inlined.
- Parameter prototypes attached to lexical subroutines are now respected when compiling sub calls without parentheses. Previously, the prototypes were honoured only for calls *with* parentheses. [RT #116735]
- Syntax errors in lexical subroutines in combination with calls to the same subroutines no longer cause crashes at compile time.
- The dtrace sub-entry probe now works with lexical subs, instead of crashing [perl #118305].
- Undefining an inlinable lexical subroutine (`my sub foo() { 42 } undef &foo`) would result in a crash if warnings were turned on.
- Deep recursion warnings no longer crash lexical subroutines. [RT #118521]

## Acknowledgements

Perl 5.18.1 represents approximately 2 months of development since Perl 5.18.0 and contains approximately 8,400 lines of changes across 60 files from 12 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.18.1:

Chris 'BinGOs' Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, David Mitchell, Father Chrysostomos, Karl Williamson, Lukas Mai, Nicholas Clark, Peter Martini, Ricardo Signes, Shlomi Fish, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5182delta − what is new for perl v5.18.2

## DESCRIPTION

This document describes differences between the 5.18.1 release and the 5.18.2 release.

If you are upgrading from an earlier release such as 5.18.0, first read perl5181delta, which describes differences between 5.18.0 and 5.18.1.

## Modules and Pragmata

### Updated Modules and Pragmata

- B has been upgraded from version 1.42_01 to 1.42_02.

  The fix for [perl #118525] introduced a regression in the behaviour of `B::CV::GV`, changing the return value from a `B::SPECIAL` object on a `NULL CvGV` to `undef`. `B::CV::GV` again returns a `B::SPECIAL` object in this case. [perl #119413]

- B::Concise has been upgraded from version 0.95 to 0.95_01.

  This fixes a bug in dumping unexpected SPECIALs.

- English has been upgraded from version 1.06 to 1.06_01. This fixes an error about the performance of `$``, `$&`, and `$'`.

- File::Glob has been upgraded from version 1.20 to 1.20_01.

## Documentation

### Changes to Existing Documentation

- perlrepository has been restored with a pointer to more useful pages.
- perlhack has been updated with the latest changes from blead.

## Selected Bug Fixes

- Perl 5.18.1 introduced a regression along with a bugfix for lexical subs. Some B::SPECIAL results from B::CV::GV became undefs instead. This broke Devel::Cover among other libraries. This has been fixed. [perl #119351]
- Perl 5.18.0 introduced a regression whereby `[:^ascii:]`, if used in the same character class as other qualifiers, would fail to match characters in the Latin−1 block. This has been fixed. [perl #120799]
- Perl 5.18.0 introduced a regression when using −>SUPER::method with AUTOLOAD by looking up AUTOLOAD from the current package, rather than the current packageXs superclass. This has been fixed. [perl #120694]
- Perl 5.18.0 introduced a regression whereby `-bareword` was no longer permitted under the `strict` and `integer` pragmata when used together. This has been fixed. [perl #120288]
- Previously PerlIOBase_dup didn't check if pushing the new layer succeeded before (optionally) setting the utf8 flag. This could cause segfaults-by-nullpointer. This has been fixed.
- A buffer overflow with very long identifiers has been fixed.
- A regression from 5.16 in the handling of padranges led to assertion failures if a keyword plugin declined to handle the second XmyX, but only after creating a padop.

  This affected, at least, Devel::CallParser under threaded builds.

  This has been fixed.

- The construct `$r=qr/.../; /$r/p` is now handled properly, an issue which had been worsened by changes 5.18.0. [perl #118213]

## Acknowledgements

Perl 5.18.2 represents approximately 3 months of development since Perl 5.18.1 and contains approximately 980 lines of changes across 39 files from 4 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.18.2:

Craig A. Berry, David Mitchell, Ricardo Signes, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control

history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5184delta − what is new for perl v5.18.4

## DESCRIPTION

This document describes differences between the 5.18.4 release and the 5.18.2 release. **Please note:** This document ignores perl 5.18.3, a broken release which existed for a few hours only.

If you are upgrading from an earlier release such as 5.18.1, first read perl5182delta, which describes differences between 5.18.1 and 5.18.2.

## Modules and Pragmata

### Updated Modules and Pragmata

- Digest::SHA has been upgraded from 5.84_01 to 5.84_02.
- perl5db.pl has been upgraded from version 1.39_10 to 1.39_11.

  This fixes a crash in tab completion, where available. [perl #120827] Also, filehandle information is properly reset after a pager is run. [perl #121456]

## Platform Support

### Platform-Specific Notes

Win32

- Introduced by [GH #12161] <https://github.com/Perl/perl5/issues/12161>, a memory leak on every call to `system` and backticks ( `` ), on most Win32 Perls starting from 5.18.0 has been fixed. The memory leak only occurred if you enabled pseudo-fork in your build of Win32 Perl, and were running that build on Server 2003 R2 or newer OS. The leak does not appear on WinXP SP3. [GH #13741] <https://github.com/Perl/perl5/issues/13741>

## Selected Bug Fixes

- The debugger now properly resets filehandles as needed. [perl #121456]
- A segfault in Digest::SHA has been addressed. [perl #121421]
- perl can again be built with USE_64_BIT_INT, with Visual C 2003, 32 bit. [perl #120925]
- A leading { (brace) in formats is properly parsed again. [perl #119973]
- Copy the values used to perturb hash iteration when cloning an interpreter. This was fairly harmless but caused `valgrind` to complain. [perl #121336]
- In Perl v5.18 `undef *_; goto &sub` and `local *_; goto &sub` started crashing. This has been fixed. [perl #119949]

## Acknowledgements

Perl 5.18.4 represents approximately 9 months of development since Perl 5.18.2 and contains approximately 2,000 lines of changes across 53 files from 13 authors.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.18.4:

Daniel Dragan, David Mitchell, Doug Bell, Father Chrysostomos, Hiroo Hayashi, James E Keenan, Karl Williamson, Mark Shelor, Ricardo Signes, Shlomi Fish, Smylers, Steve Hay, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5200delta − what is new for perl v5.20.0

## DESCRIPTION

This document describes differences between the 5.18.0 release and the 5.20.0 release.

If you are upgrading from an earlier release such as 5.16.0, first read perl5180delta, which describes differences between 5.16.0 and 5.18.0.

## Core Enhancements

### Experimental Subroutine signatures

Declarative syntax to unwrap argument list into lexical variables. `sub foo ($a,$b) {...}` checks the number of arguments and puts the arguments into lexical variables. Signatures are not equivalent to the existing idiom of `sub foo { my($a,$b) = @_; ... }`. Signatures are only available by enabling a non-default feature, and generate warnings about being experimental. The syntactic clash with prototypes is managed by disabling the short prototype syntax when signatures are enabled.

See ‘‘Signatures’’ in perlsub for details.

### `sub`s now take a `prototype` attribute

When declaring or defining a `sub`, the prototype can now be specified inside of a `prototype` attribute instead of in parens following the name.

For example, `sub foo($$){}` could be rewritten as `sub foo : prototype($$){}`.

### More consistent prototype parsing

Multiple semicolons in subroutine prototypes have long been tolerated and treated as a single semicolon. There was one case where this did not happen. A subroutine whose prototype begins with ‘‘*’’ or ‘‘;*’’ can affect whether a bareword is considered a method name or sub call. This now applies also to ‘‘;;;*’’.

Whitespace has long been allowed inside subroutine prototypes, so `sub( $ $ )` is equivalent to `sub($$)`, but until now it was stripped when the subroutine was parsed. Hence, whitespace was *not* allowed in prototypes set by `Scalar::Util::set_prototype`. Now it is permitted, and the parser no longer strips whitespace. This means `prototype &mysub` returns the original prototype, whitespace and all.

### `rand` now uses a consistent random number generator

Previously perl would use a platform specific random number generator, varying between the libc **rand()**, **random()** or **drand48()**.

This meant that the quality of perl’s random numbers would vary from platform to platform, from the 15 bits of **rand()** on Windows to 48−bits on POSIX platforms such as Linux with **drand48()**.

Perl now uses its own internal **drand48()** implementation on all platforms. This does not make perl’s `rand` cryptographically secure. [perl #115928]

### New slice syntax

The new `%hash{...}` and `%array[...]` syntax returns a list of key/value (or index/value) pairs. See ‘‘Key/Value Hash Slices’’ in perldata.

### Experimental Postfix Dereferencing

When the `postderef` feature is in effect, the following syntactical equivalencies are set up:

```
$sref->$*;  # same as ${ $sref }  # interpolates
$aref->@*;  # same as @{ $aref }  # interpolates
$href->%*;  # same as %{ $href }
$cref->&*;  # same as &{ $cref }
$gref->**;  # same as *{ $gref }

$aref->$#*; # same as $#{ $aref }

$gref->*{ $slot }; # same as *{ $gref }{ $slot }

$aref->@[ ... ];  # same as @$aref[ ... ]  # interpolates
$href->@{ ... };  # same as @$href{ ... }  # interpolates
```

```
$aref->%[ ... ];  # same as %$aref[ ... ]
$href->%{ ... };  # same as %$href{ ... }
```

Those marked as interpolating only interpolate if the associated `postderef_qq` feature is also enabled. This feature is **experimental** and will trigger `experimental::postderef`-category warnings when used, unless they are suppressed.

For more information, consult the Postfix Dereference Syntax section of perlref.

#### Unicode 6.3 now supported

Perl now supports and is shipped with Unicode 6.3 (though Perl may be recompiled with any previous Unicode release as well). A detailed list of Unicode 6.3 changes is at <http://www.unicode.org/versions/Unicode6.3.0/>.

#### New `\p{Unicode}` regular expression pattern property

This is a synonym for `\p{Any}` and matches the set of Unicode-defined code points 0 − 0x10FFFF.

#### Better 64-bit support

On 64-bit platforms, the internal array functions now use 64-bit offsets, allowing Perl arrays to hold more than 2**31 elements, if you have the memory available.

The regular expression engine now supports strings longer than 2**31 characters. [perl #112790, #116907]

The functions PerlIO_get_bufsiz, PerlIO_get_cnt, PerlIO_set_cnt and PerlIO_set_ptrcnt now have SSize_t, rather than int, return values and parameters.

#### `use locale` now works on UTF-8 locales

Until this release, only single-byte locales, such as the ISO 8859 series were supported. Now, the increasingly common multi-byte UTF-8 locales are also supported. A UTF-8 locale is one in which the character set is Unicode and the encoding is UTF-8. The POSIX `LC_CTYPE` category operations (case changing (like `lc()`, `"\U"`), and character classification (`\w`, `\D`, `qr/[[:punct:]]/`)) under such a locale work just as if not under locale, but instead as if under `use feature 'unicode_strings'`, except taint rules are followed. Sorting remains by code point order in this release. [perl #56820].

#### `use locale` now compiles on systems without locale ability

Previously doing this caused the program to not compile. Within its scope the program behaves as if in the "C" locale. Thus programs written for platforms that support locales can run on locale-less platforms without change. Attempts to change the locale away from the "C" locale will, of course, fail.

#### More locale initialization fallback options

If there was an error with locales during Perl start-up, it immediately gave up and tried to use the `"C"` locale. Now it first tries using other locales given by the environment variables, as detailed in "ENVIRONMENT" in perllocale. For example, if `LC_ALL` and `LANG` are both set, and using the `LC_ALL` locale fails, Perl will now try the `LANG` locale, and only if that fails, will it fall back to `"C"`. On Windows machines, Perl will try, ahead of using `"C"`, the system default locale if all the locales given by environment variables fail.

#### `-DL` runtime option now added for tracing locale setting

This is designed for Perl core developers to aid in field debugging bugs regarding locales.

#### **-F** now implies **-a** and **-a** implies **-n**

Previously **-F** without **-a** was a no-op, and **-a** without **-n** or **-p** was a no-op, with this change, if you supply **-F** then both **-a** and **-n** are implied and if you supply **-a** then **-n** is implied.

You can still use **-p** for its extra behaviour. [perl #116190]

#### `$a` and `$b` warnings exemption

The special variables `$a` and `$b`, used in `sort`, are now exempt from "used once" warnings, even where `sort` is not used. This makes it easier for CPAN modules to provide functions using `$a` and `$b` for similar purposes. [perl #120462]

### Security

#### Avoid possible read of free()d memory during parsing

It was possible that **free()**d memory could be read during parsing in the unusual circumstance of the Perl program ending with a heredoc and the last line of the file on disk having no terminating newline

character. This has now been fixed.

## Incompatible Changes

### `do` can no longer be used to call subroutines

The `do SUBROUTINE(LIST)` form has resulted in a deprecation warning since Perl v5.0.0, and is now a syntax error.

### Quote-like escape changes

The character after `\c` in a double-quoted string (“...” or qq(...)) or regular expression must now be a printable character and may not be `{`.

A literal `{` after `\B` or `\b` is now fatal.

These were deprecated in perl v5.14.0.

### Tainting happens under more circumstances; now conforms to documentation

This affects regular expression matching and changing the case of a string (`lc`, `"\U"`, *etc.*) within the scope of `use locale`. The result is now tainted based on the operation, no matter what the contents of the string were, as the documentation (perlsec, “SECURITY” in perllocale) indicates it should. Previously, for the case change operation, if the string contained no characters whose case change could be affected by the locale, the result would not be tainted. For example, the result of `uc()` on an empty string or one containing only above−Latin1 code points is now tainted, and wasn’t before. This leads to more consistent tainting results. Regular expression patterns taint their non-binary results (like `$&`, `$2`) if and only if the pattern contains elements whose matching depends on the current (potentially tainted) locale. Like the case changing functions, the actual contents of the string being matched now do not matter, whereas formerly it did. For example, if the pattern contains a `\w`, the results will be tainted even if the match did not have to use that portion of the pattern to succeed or fail, because what a `\w` matches depends on locale. However, for example, a `.` in a pattern will not enable tainting, because the dot matches any single character, and what the current locale is doesn’t change in any way what matches and what doesn’t.

### `\p{}`, `\P{}` matching has changed for non-Unicode code points.

`\p{}` and `\P{}` are defined by Unicode only on Unicode-defined code points (`U+0000` through `U+10FFFF`). Their behavior on matching these legal Unicode code points is unchanged, but there are changes for code points `0x110000` and above. Previously, Perl treated the result of matching `\p{}` and `\P{}` against these as `undef`, which translates into “false”. For `\P{}`, this was then complemented into “true”. A warning was supposed to be raised when this happened. However, various optimizations could prevent the warning, and the results were often counter-intuitive, with both a match and its seeming complement being false. Now all non-Unicode code points are treated as typical unassigned Unicode code points. This generally is more Do-What-I-Mean. A warning is raised only if the results are arguably different from a strict Unicode approach, and from what Perl used to do. Code that needs to be strictly Unicode compliant can make this warning fatal, and then Perl always raises the warning.

Details are in “Beyond Unicode code points” in perlunicode.

### `\p{All}` has been expanded to match all possible code points

The Perl-defined regular expression pattern element `\p{All}`, unused on CPAN, used to match just the Unicode code points; now it matches all possible code points; that is, it is equivalent to `qr/./s`. Thus `\p{All}` is no longer synonymous with `\p{Any}`, which continues to match just the Unicode code points, as Unicode says it should.

### Data::Dumper’s output may change

Depending on the data structures dumped and the settings set for Data::Dumper, the dumped output may have changed from previous versions.

If you have tests that depend on the exact output of Data::Dumper, they may fail.

To avoid this problem in your code, test against the data structure from evaluating the dumped structure, instead of the dump itself.

### Locale decimal point character no longer leaks outside of `use locale` scope

This is actually a bug fix, but some code has come to rely on the bug being present, so this change is listed here. The current locale that the program is running under is not supposed to be visible to Perl code except within the scope of a `use locale`. However, until now under certain circumstances, the character used for a decimal point (often a comma) leaked outside the scope. If your code is affected

by this change, simply add a `use locale`.

### Assignments of Windows sockets error codes to $! now prefer *errno.h* values over WSAGetLastError() values

In previous versions of Perl, Windows sockets error codes as returned by **WSAGetLastError()** were assigned to $!, and some constants such as ECONNABORTED, not in *errno.h* in VC++ (or the various Windows ports of gcc) were defined to corresponding WSAE* values to allow $! to be tested against the E* constants exported by Errno and POSIX.

This worked well until VC++ 2010 and later, which introduced new E* constants with values > 100 into *errno.h*, including some being (re)defined by perl to WSAE* values. That caused problems when linking XS code against other libraries which used the original definitions of *errno.h* constants.

To avoid this incompatibility, perl now maps WSAE* error codes to E* values where possible, and assigns those values to $!. The E* constants exported by Errno and POSIX are updated to match so that testing $! against them, wherever previously possible, will continue to work as expected, and all E* constants found in *errno.h* are now exported from those modules with their original *errno.h* values.

In order to avoid breakage in existing Perl code which assigns WSAE* values to $!, perl now intercepts the assignment and performs the same mapping to E* values as it uses internally when assigning to $! itself.

However, one backwards-incompatibility remains: existing Perl code which compares $! against the numeric values of the WSAE* error codes that were previously assigned to $! will now be broken in those cases where a corresponding E* value has been assigned instead. This is only an issue for those E* values < 100, which were always exported from Errno and POSIX with their original *errno.h* values, and therefore could not be used for WSAE* error code tests (e.g. WSAEINVAL is 10022, but the corresponding EINVAL is 22). (E* values > 100, if present, were redefined to WSAE* values anyway, so compatibility can be achieved by using the E* constants, which will work both before and after this change, albeit using different numeric values under the hood.)

### Functions `PerlIO_vsprintf` and `PerlIO_sprintf` have been removed

These two functions, undocumented, unused in CPAN, and problematic, have been removed.

## Deprecations

### The `/\C/` character class

The `/\C/` regular expression character class is deprecated. From perl 5.22 onwards it will generate a warning, and from perl 5.24 onwards it will be a regular expression compiler error. If you need to examine the individual bytes that make up a UTF8−encoded character, then use `utf8::encode()` on the string (or a copy) first.

### Literal control characters in variable names

This deprecation affects things like $\cT, where \cT is a literal control (such as a `NAK` or `NEGATIVE ACKNOWLEDGE` character) in the source code. Surprisingly, it appears that originally this was intended as the canonical way of accessing variables like $^T, with the caret form only being added as an alternative.

The literal control form is being deprecated for two main reasons. It has what are likely unfixable bugs, such as $\cI not working as an alias for $^I, and their usage not being portable to non-ASCII platforms: While $^T will work everywhere, \cT is whitespace in EBCDIC. [perl #119123]

### References to non-integers and non-positive integers in `$/`

Setting `$/` to a reference to zero or a reference to a negative integer is now deprecated, and will behave **exactly** as though it was set to `undef`. If you want slurp behavior set `$/` to `undef` explicitly.

Setting `$/` to a reference to a non integer is now forbidden and will throw an error. Perl has never documented what would happen in this context and while it used to behave the same as setting `$/` to the address of the references in future it may behave differently, so we have forbidden this usage.

### Character matching routines in POSIX

Use of any of these functions in the `POSIX` module is now deprecated: `isalnum`, `isalpha`, `iscntrl`, `isdigit`, `isgraph`, `islower`, `isprint`, `ispunct`, `isspace`, `isupper`, and `isxdigit`. The functions are buggy and don't work on UTF−8 encoded strings. See their entries in POSIX for more information.

A warning is raised on the first call to any of them from each place in the code that they are called.

(Hence a repeated statement in a loop will raise just the one warning.)

### Interpreter-based threads are now *discouraged*

The ‘‘interpreter-based threads’’ provided by Perl are not the fast, lightweight system for multitasking that one might expect or hope for. Threads are implemented in a way that make them easy to misuse. Few people know how to use them correctly or will be able to provide help.

The use of interpreter-based threads in perl is officially discouraged.

### Module removals

The following modules will be removed from the core distribution in a future release, and will at that time need to be installed from CPAN. Distributions on CPAN which require these modules will need to list them as prerequisites.

The core versions of these modules will now issue `"deprecated"`-category warnings to alert you to this fact. To silence these deprecation warnings, install the modules in question from CPAN.

Note that the planned removal of these modules from core does not reflect a judgement about the quality of the code and should not be taken as a suggestion that their use be halted. Their disinclusion from core primarily hinges on their necessity to bootstrapping a fully functional, CPAN-capable Perl installation, not on concerns over their design.

CGI and its associated CGI:: packages
inc::latest
Package::Constants
Module::Build and its associated Module::Build:: packages

### Utility removals

The following utilities will be removed from the core distribution in a future release, and will at that time need to be installed from CPAN.

find2perl
s2p
a2p

## Performance Enhancements

- Perl has a new copy-on-write mechanism that avoids the need to copy the internal string buffer when assigning from one scalar to another. This makes copying large strings appear much faster. Modifying one of the two (or more) strings after an assignment will force a copy internally. This makes it unnecessary to pass strings by reference for efficiency.

  This feature was already available in 5.18.0, but wasn't enabled by default. It is the default now, and so you no longer need build perl with the *Configure* argument:

  ```
  -Accflags=-DPERL_NEW_COPY_ON_WRITE
  ```

  It can be disabled (for now) in a perl build with:

  ```
  -Accflags=-DPERL_NO_COW
  ```

  On some operating systems Perl can be compiled in such a way that any attempt to modify string buffers shared by multiple SVs will crash. This way XS authors can test that their modules handle copy-on-write scalars correctly. See ‘‘Copy on Write’’ in perlguts for detail.

- Perl has an optimizer for regular expression patterns. It analyzes the pattern to find things such as the minimum length a string has to be to match, etc. It now better handles code points that are above the Latin1 range.

- Executing a regex that contains the `^` anchor (or its variant under the `/m` flag) has been made much faster in several situations.

- Precomputed hash values are now used in more places during method lookup.

- Constant hash key lookups (`$hash{key}` as opposed to `$hash{$key}`) have long had the internal hash value computed at compile time, to speed up lookup. This optimisation has only now been applied to hash slices as well.

- Combined `and` and `or` operators in void context, like those generated for `unless ($a && $b)` and `if ($a || b)` now short circuit directly to the end of the statement. [perl #120128]

- In certain situations, when `return` is the last statement in a subroutine's main scope, it will be optimized out. This means code like:

  ```
  sub baz { return $cat; }
  ```

  will now behave like:

  ```
  sub baz { $cat; }
  ```

  which is notably faster.

  [perl #120765]

- Code like:

  ```
  my $x; # or @x, %x
  my $y;
  ```

  is now optimized to:

  ```
  my ($x, $y);
  ```

  In combination with the padrange optimization introduced in v5.18.0, this means longer uninitialized my variable statements are also optimized, so:

  ```
  my $x; my @y; my %z;
  ```

  becomes:

  ```
  my ($x, @y, %z);
  ```

  [perl #121077]

- The creation of certain sorts of lists, including array and hash slices, is now faster.
- The optimisation for arrays indexed with a small constant integer is now applied for integers in the range −128..127, rather than 0..255. This should speed up Perl code using expressions like `$x[-1]`, at the expense of (presumably much rarer) code using expressions like `$x[200]`.
- The first iteration over a large hash (using `keys` or `each`) is now faster. This is achieved by preallocating the hash's internal iterator state, rather than lazily creating it when the hash is first iterated. (For small hashes, the iterator is still created only when first needed. The assumption is that small hashes are more likely to be used as objects, and therefore never allocated. For large hashes, that's less likely to be true, and the cost of allocating the iterator is swamped by the cost of allocating space for the hash itself.)
- When doing a global regex match on a string that came from the `readline` or `<>` operator, the data is no longer copied unnecessarily. [perl #121259]
- Dereferencing (as in `$obj->[0]` or `$obj->{k}`) is now faster when `$obj` is an instance of a class that has overloaded methods, but doesn't overload any of the dereferencing methods `@{}`, `%{}`, and so on.
- Perl's optimiser no longer skips optimising code that follows certain `eval {}` expressions (including those with an apparent infinite loop).
- The implementation now does a better job of avoiding meaningless work at runtime. Internal effect-free ‘‘null’’ operations (created as a side-effect of parsing Perl programs) are normally deleted during compilation. That deletion is now applied in some situations that weren't previously handled.
- Perl now does less disk I/O when dealing with Unicode properties that cover up to three ranges of consecutive code points.

## Modules and Pragmata

### New Modules and Pragmata

- experimental 0.007 has been added to the Perl core.
- IO::Socket::IP 0.29 has been added to the Perl core.

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 1.90 to 1.96.
- arybase has been upgraded from version 0.06 to 0.07.
- Attribute::Handlers has been upgraded from version 0.94 to 0.96.
- attributes has been upgraded from version 0.21 to 0.22.
- autodie has been upgraded from version 2.13 to 2.23.
- AutoLoader has been upgraded from version 5.73 to 5.74.
- autouse has been upgraded from version 1.07 to 1.08.
- B has been upgraded from version 1.42 to 1.48.
- B::Concise has been upgraded from version 0.95 to 0.992.
- B::Debug has been upgraded from version 1.18 to 1.19.
- B::Deparse has been upgraded from version 1.20 to 1.26.
- base has been upgraded from version 2.18 to 2.22.
- Benchmark has been upgraded from version 1.15 to 1.18.
- bignum has been upgraded from version 0.33 to 0.37.
- Carp has been upgraded from version 1.29 to 1.3301.
- CGI has been upgraded from version 3.63 to 3.65. NOTE: CGI is deprecated and may be removed from a future version of Perl.
- charnames has been upgraded from version 1.36 to 1.40.
- Class::Struct has been upgraded from version 0.64 to 0.65.
- Compress::Raw::Bzip2 has been upgraded from version 2.060 to 2.064.
- Compress::Raw::Zlib has been upgraded from version 2.060 to 2.065.
- Config::Perl::V has been upgraded from version 0.17 to 0.20.
- constant has been upgraded from version 1.27 to 1.31.
- CPAN has been upgraded from version 2.00 to 2.05.
- CPAN::Meta has been upgraded from version 2.120921 to 2.140640.
- CPAN::Meta::Requirements has been upgraded from version 2.122 to 2.125.
- CPAN::Meta::YAML has been upgraded from version 0.008 to 0.012.
- Data::Dumper has been upgraded from version 2.145 to 2.151.
- DB has been upgraded from version 1.04 to 1.07.
- DB_File has been upgraded from version 1.827 to 1.831.
- DBM_Filter has been upgraded from version 0.05 to 0.06.
- deprecate has been upgraded from version 0.02 to 0.03.
- Devel::Peek has been upgraded from version 1.11 to 1.16.
- Devel::PPPort has been upgraded from version 3.20 to 3.21.
- diagnostics has been upgraded from version 1.31 to 1.34.
- Digest::MD5 has been upgraded from version 2.52 to 2.53.
- Digest::SHA has been upgraded from version 5.84 to 5.88.
- DynaLoader has been upgraded from version 1.18 to 1.25.
- Encode has been upgraded from version 2.49 to 2.60.
- encoding has been upgraded from version 2.6_01 to 2.12.

- English has been upgraded from version 1.06 to 1.09.

  `$OLD_PERL_VERSION` was added as an alias of `$]`.
- Errno has been upgraded from version 1.18 to 1.20_03.
- Exporter has been upgraded from version 5.68 to 5.70.
- ExtUtils::CBuilder has been upgraded from version 0.280210 to 0.280216.
- ExtUtils::Command has been upgraded from version 1.17 to 1.18.
- ExtUtils::Embed has been upgraded from version 1.30 to 1.32.
- ExtUtils::Install has been upgraded from version 1.59 to 1.67.
- ExtUtils::MakeMaker has been upgraded from version 6.66 to 6.98.
- ExtUtils::Miniperl has been upgraded from version  to 1.01.
- ExtUtils::ParseXS has been upgraded from version 3.18 to 3.24.
- ExtUtils::Typemaps has been upgraded from version 3.19 to 3.24.
- ExtUtils::XSSymSet has been upgraded from version 1.2 to 1.3.
- feature has been upgraded from version 1.32 to 1.36.
- fields has been upgraded from version 2.16 to 2.17.
- File::Basename has been upgraded from version 2.84 to 2.85.
- File::Copy has been upgraded from version 2.26 to 2.29.
- File::DosGlob has been upgraded from version 1.10 to 1.12.
- File::Fetch has been upgraded from version 0.38 to 0.48.
- File::Find has been upgraded from version 1.23 to 1.27.
- File::Glob has been upgraded from version 1.20 to 1.23.
- File::Spec has been upgraded from version 3.40 to 3.47.
- File::Temp has been upgraded from version 0.23 to 0.2304.
- FileCache has been upgraded from version 1.08 to 1.09.
- Filter::Simple has been upgraded from version 0.89 to 0.91.
- Filter::Util::Call has been upgraded from version 1.45 to 1.49.
- Getopt::Long has been upgraded from version 2.39 to 2.42.
- Getopt::Std has been upgraded from version 1.07 to 1.10.
- Hash::Util::FieldHash has been upgraded from version 1.10 to 1.15.
- HTTP::Tiny has been upgraded from version 0.025 to 0.043.
- I18N::Langinfo has been upgraded from version 0.10 to 0.11.
- I18N::LangTags has been upgraded from version 0.39 to 0.40.
- if has been upgraded from version 0.0602 to 0.0603.
- inc::latest has been upgraded from version 0.4003 to 0.4205. NOTE: inc::latest is deprecated and may be removed from a future version of Perl.
- integer has been upgraded from version 1.00 to 1.01.
- IO has been upgraded from version 1.28 to 1.31.
- IO::Compress::Gzip and friends have been upgraded from version 2.060 to 2.064.
- IPC::Cmd has been upgraded from version 0.80 to 0.92.
- IPC::Open3 has been upgraded from version 1.13 to 1.16.
- IPC::SysV has been upgraded from version 2.03 to 2.04.

- JSON::PP has been upgraded from version 2.27202 to 2.27203.
- List::Util has been upgraded from version 1.27 to 1.38.
- locale has been upgraded from version 1.02 to 1.03.
- Locale::Codes has been upgraded from version 3.25 to 3.30.
- Locale::Maketext has been upgraded from version 1.23 to 1.25.
- Math::BigInt has been upgraded from version 1.9991 to 1.9993.
- Math::BigInt::FastCalc has been upgraded from version 0.30 to 0.31.
- Math::BigRat has been upgraded from version 0.2604 to 0.2606.
- MIME::Base64 has been upgraded from version 3.13 to 3.14.
- Module::Build has been upgraded from version 0.4003 to 0.4205. NOTE: Module::Build is deprecated and may be removed from a future version of Perl.
- Module::CoreList has been upgraded from version 2.89 to 3.10.
- Module::Load has been upgraded from version 0.24 to 0.32.
- Module::Load::Conditional has been upgraded from version 0.54 to 0.62.
- Module::Metadata has been upgraded from version 1.000011 to 1.000019.
- mro has been upgraded from version 1.11 to 1.16.
- Net::Ping has been upgraded from version 2.41 to 2.43.
- Opcode has been upgraded from version 1.25 to 1.27.
- Package::Constants has been upgraded from version 0.02 to 0.04. NOTE: Package::Constants is deprecated and may be removed from a future version of Perl.
- Params::Check has been upgraded from version 0.36 to 0.38.
- parent has been upgraded from version 0.225 to 0.228.
- Parse::CPAN::Meta has been upgraded from version 1.4404 to 1.4414.
- Perl::OSType has been upgraded from version 1.003 to 1.007.
- perlfaq has been upgraded from version 5.0150042 to 5.0150044.
- PerlIO has been upgraded from version 1.07 to 1.09.
- PerlIO::encoding has been upgraded from version 0.16 to 0.18.
- PerlIO::scalar has been upgraded from version 0.16 to 0.18.
- PerlIO::via has been upgraded from version 0.12 to 0.14.
- Pod::Escapes has been upgraded from version 1.04 to 1.06.
- Pod::Functions has been upgraded from version 1.06 to 1.08.
- Pod::Html has been upgraded from version 1.18 to 1.21.
- Pod::Parser has been upgraded from version 1.60 to 1.62.
- Pod::Perldoc has been upgraded from version 3.19 to 3.23.
- Pod::Usage has been upgraded from version 1.61 to 1.63.
- POSIX has been upgraded from version 1.32 to 1.38_03.
- re has been upgraded from version 0.23 to 0.26.
- Safe has been upgraded from version 2.35 to 2.37.
- Scalar::Util has been upgraded from version 1.27 to 1.38.
- SDBM_File has been upgraded from version 1.09 to 1.11.
- Socket has been upgraded from version 2.009 to 2.013.

- Storable has been upgraded from version 2.41 to 2.49.
- strict has been upgraded from version 1.07 to 1.08.
- subs has been upgraded from version 1.01 to 1.02.
- Sys::Hostname has been upgraded from version 1.17 to 1.18.
- Sys::Syslog has been upgraded from version 0.32 to 0.33.
- Term::Cap has been upgraded from version 1.13 to 1.15.
- Term::ReadLine has been upgraded from version 1.12 to 1.14.
- Test::Harness has been upgraded from version 3.26 to 3.30.
- Test::Simple has been upgraded from version 0.98 to 1.001002.
- Text::ParseWords has been upgraded from version 3.28 to 3.29.
- Text::Tabs has been upgraded from version 2012.0818 to 2013.0523.
- Text::Wrap has been upgraded from version 2012.0818 to 2013.0523.
- Thread has been upgraded from version 3.02 to 3.04.
- Thread::Queue has been upgraded from version 3.02 to 3.05.
- threads has been upgraded from version 1.86 to 1.93.
- threads::shared has been upgraded from version 1.43 to 1.46.
- Tie::Array has been upgraded from version 1.05 to 1.06.
- Tie::File has been upgraded from version 0.99 to 1.00.
- Tie::Hash has been upgraded from version 1.04 to 1.05.
- Tie::Scalar has been upgraded from version 1.02 to 1.03.
- Tie::StdHandle has been upgraded from version 4.3 to 4.4.
- Time::HiRes has been upgraded from version 1.9725 to 1.9726.
- Time::Piece has been upgraded from version 1.20_01 to 1.27.
- Unicode::Collate has been upgraded from version 0.97 to 1.04.
- Unicode::Normalize has been upgraded from version 1.16 to 1.17.
- Unicode::UCD has been upgraded from version 0.51 to 0.57.
- utf8 has been upgraded from version 1.10 to 1.13.
- version has been upgraded from version 0.9902 to 0.9908.
- vmsish has been upgraded from version 1.03 to 1.04.
- warnings has been upgraded from version 1.18 to 1.23.
- Win32 has been upgraded from version 0.47 to 0.49.
- XS::Typemap has been upgraded from version 0.10 to 0.13.
- XSLoader has been upgraded from version 0.16 to 0.17.

## Documentation

### New Documentation

*perlrepository*

This document was removed (actually, renamed perlgit and given a major overhaul) in Perl v5.14, causing Perl documentation websites to show the now out of date version in Perl v5.12 as the latest version. It has now been restored in stub form, directing readers to current information.

### Changes to Existing Documentation

*perldata*

- New sections have been added to document the new index/value array slice and key/value hash slice syntax.

*perldebguts*

- The `DB::goto` and `DB::lsub` debugger subroutines are now documented. [perl #77680]

*perlexperiment*

- `\s` matching `\cK` is marked experimental.
- ithreads were accepted in v5.8.0 (but are discouraged as of v5.20.0).
- Long doubles are not considered experimental.
- Code in regular expressions, regular expression backtracking verbs, and lvalue subroutines are no longer listed as experimental. (This also affects perlre and perlsub.)

*perlfunc*

- `chop` and `chomp` now note that they can reset the hash iterator.
- `exec`'s handling of arguments is now more clearly documented.
- `eval EXPR` now has caveats about expanding floating point numbers in some locales.
- `goto EXPR` is now documented to handle an expression that evalutes to a code reference as if it was `goto &$coderef`. This behavior is at least ten years old.
- Since Perl v5.10, it has been possible for subroutines in `@INC` to return a reference to a scalar holding initial source code to prepend to the file. This is now documented.
- The documentation of `ref` has been updated to recommend the use of `blessed`, `isa` and `reftype` when dealing with references to blessed objects.

*perlguts*

- Numerous minor changes have been made to reflect changes made to the perl internals in this release.
- New sections on Read-Only Values and Copy on Write have been added.

*perlhack*

- The Super Quick Patch Guide section has been updated.

*perlhacktips*

- The documentation has been updated to include some more examples of `gdb` usage.

*perllexwarn*

- The perllexwarn documentation used to describe the hierarchy of warning categories understood by the warnings pragma. That description has now been moved to the warnings documentation itself, leaving perllexwarn as a stub that points to it. This change consolidates all documentation for lexical warnings in a single place.

*perllocale*

- The documentation now mentions ***fc()*** and `\F`, and includes many clarifications and corrections in general.

*perlop*

- The language design of Perl has always called for monomorphic operators. This is now mentioned explicitly.

*perlopentut*

- The `open` tutorial has been completely rewritten by Tom Christiansen, and now focuses on covering only the basics, rather than providing a comprehensive reference to all things openable. This rewrite came as the result of a vigorous discussion on perl5−porters kicked off by a set of improvements written by Alexander Hartmaier to the existing perlopentut. A "more than you ever wanted to know about `open`" document may follow in subsequent versions of perl.

*perlre*

- The fact that the regexp engine makes no effort to call (?{}) and (??{}) constructs any specified number of times (although it will basically DWIM in case of a successful match) has been documented.

- The `/r` modifier (for non-destructive substitution) is now documented. [perl #119151]
- The documentation for `/x` and `(?# comment)` has been expanded and clarified.

*perlreguts*

- The documentation has been updated in the light of recent changes to *regcomp.c*.

*perlsub*

- The need to predeclare recursive functions with prototypes in order for the prototype to be honoured in the recursive call is now documented. [perl #2726]
- A list of subroutine names used by the perl implementation is now included. [perl #77680]

*perltrap*

- There is now a JavaScript section.

*perlunicode*

- The documentation has been updated to reflect `Bidi_Class` changes in Unicode 6.3.

*perlvar*

- A new section explaining the performance issues of $‘, $& and $’, including workarounds and changes in different versions of Perl, has been added.
- Three English variable names which have long been documented but do not actually exist have been removed from the documentation. These were `$OLD_PERL_VERSION`, `$OFMT`, and `$ARRAY_BASE`.

  (Actually, `OLD_PERL_VERSION` *does* exist, starting with this revision, but remained undocumented until perl 5.22.0.)

*perlxs*

- Several problems in the `MY_CXT` example have been fixed.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- delete argument is index/value array slice, use array slice

  (F) You used index/value array slice syntax (`%array[...]`) as the argument to `delete`. You probably meant `@array[...]` with an @ symbol instead.

- delete argument is key/value hash slice, use hash slice

  (F) You used key/value hash slice syntax (`%hash{...}`) as the argument to `delete`. You probably meant `@hash{...}` with an @ symbol instead.

- Magical list constants are not supported

  (F) You assigned a magical array to a stash element, and then tried to use the subroutine from the same slot. You are asking Perl to do something it cannot do, details subject to change between Perl versions.

- Added Setting $/ to a `%s` reference is forbidden

*New Warnings*

- `%s` on reference is experimental:

  The ‘‘auto-deref’’ feature is experimental.

  Starting in v5.14.0, it was possible to use push, pop, keys, and other built-in functions not only on aggregate types, but on references to them. The feature was not deployed to its original intended specification, and now may become redundant to postfix dereferencing. It has always been categorized as an experimental feature, and in v5.20.0 is carries a warning as such.

  Warnings will now be issued at compile time when these operations are detected.

```
no if $] >= 5.01908, warnings => "experimental::autoderef";
```

Consider, though, replacing the use of these features, as they may change behavior again before becoming stable.

- A sequence of multiple spaces in a charnames alias definition is deprecated

  Trailing white-space in a charnames alias definition is deprecated

  These two deprecation warnings involving `\N{...}` were incorrectly implemented. They did not warn by default (now they do) and could not be made fatal via `use warnings FATAL => 'deprecated'` (now they can).

- Attribute prototype(%s) discards earlier prototype attribute in same sub

  (W misc) A sub was declared as `sub foo : prototype(A) : prototype(B) {}`, for example. Since each sub can only have one prototype, the earlier declaration(s) are discarded while the last one is applied.

- Invalid \0 character in %s for `%s: %s`\0%s

  (W syscalls) Embedded \0 characters in pathnames or other system call arguments produce a warning as of 5.20. The parts after the \0 were formerly ignored by system calls.

- Matched non-Unicode code point 0x%X against Unicode property; may not be portable.

  This replaces the message ‘‘Code point 0x%X is not Unicode, all \p{} matches fail; all \P{} matches succeed’’.

- Missing ’]’ in prototype for %s : %s

  (W illegalproto) A grouping was started with `[` but never closed with `]`.

- Possible precedence issue with control flow operator

  (W syntax) There is a possible problem with the mixing of a control flow operator (e.g. `return`) and a low-precedence operator like `or`. Consider:

  ```
  sub { return $a or $b; }
  ```

  This is parsed as:

  ```
  sub { (return $a) or $b; }
  ```

  Which is effectively just:

  ```
  sub { return $a; }
  ```

  Either use parentheses or the high-precedence variant of the operator.

  Note this may be also triggered for constructs like:

  ```
  sub { 1 if die; }
  ```

- Postfix dereference is experimental

  (S experimental::postderef) This warning is emitted if you use the experimental postfix dereference syntax. Simply suppress the warning if you want to use the feature, but know that in doing so you are taking the risk of using an experimental feature which may change or be removed in a future Perl version:

  ```
  no warnings "experimental::postderef";
  use feature "postderef", "postderef_qq";
  $ref->$*;
  $aref->@*;
  $aref->@[@indices];
  ... etc ...
  ```

- Prototype ’%s’ overridden by attribute ’prototype(%s)’ in %s

  (W prototype) A prototype was declared in both the parentheses after the sub name and via the prototype attribute. The prototype in parentheses is useless, since it will be replaced by the prototype from the attribute before it’s ever used.

- Scalar value @%s[%s] better written as $%s[%s]

  (W syntax) In scalar context, you've used an array index/value slice (indicated by %) to select a single element of an array. Generally it's better to ask for a scalar value (indicated by $). The difference is that `$foo[&bar]` always behaves like a scalar, both in the value it returns and when evaluating its argument, while `%foo[&bar]` provides a list context to its subscript, which can do weird things if you're expecting only one subscript. When called in list context, it also returns the index (what `&bar` returns) in addition to the value.

- Scalar value @%s{%s} better written as $%s{%s}

  (W syntax) In scalar context, you've used a hash key/value slice (indicated by %) to select a single element of a hash. Generally it's better to ask for a scalar value (indicated by $). The difference is that `$foo{&bar}` always behaves like a scalar, both in the value it returns and when evaluating its argument, while `@foo{&bar}` and provides a list context to its subscript, which can do weird things if you're expecting only one subscript. When called in list context, it also returns the key in addition to the value.

- Setting $/ to a reference to `%s` as a form of slurp is deprecated, treating as undef

- Unexpected exit `%u`

  (S) **exit()** was called or the script otherwise finished gracefully when `PERL_EXIT_WARN` was set in `PL_exit_flags`.

- Unexpected exit failure `%d`

  (S) An uncaught **die()** was called when `PERL_EXIT_WARN` was set in `PL_exit_flags`.

- Use of literal control characters in variable names is deprecated

  (D deprecated) Using literal control characters in the source to refer to the ^FOO variables, like $^X and ${^GLOBAL_PHASE} is now deprecated. This only affects code like $\cT, where \cT is a control (like a `SOH`) in the source code: ${"\cT"} and $^T remain valid.

- Useless use of greediness modifier

  This fixes [Perl #42957].

### Changes to Existing Diagnostics

- Warnings and errors from the regexp engine are now UTF-8 clean.

- The "Unknown switch condition" error message has some slight changes. This error triggers when there is an unknown condition in a `(?(foo))` conditional. The error message used to read:

  ```
  Unknown switch condition (?(%s in regex;
  ```

  But what `%s` could be was mostly up to luck. For `(?(foobar))`, you might have seen "fo" or "f". For Unicode characters, you would generally get a corrupted string. The message has been changed to read:

  ```
  Unknown switch condition (?(...)) in regex;
  ```

  Additionally, the `'<-- HERE'` marker in the error will now point to the correct spot in the regex.

- The "%s "\x%X" does not map to Unicode" warning is now correctly listed as a severe warning rather than as a fatal error.

- Under rare circumstances, one could get a "Can't coerce readonly REF to string" instead of the customary "Modification of a read-only value". This alternate error message has been removed.

- "Ambiguous use of * resolved as operator *": This and similar warnings about "%" and "&" used to occur in some circumstances where there was no operator of the type cited, so the warning was completely wrong. This has been fixed [perl #117535, #76910].

- Warnings about malformed subroutine prototypes are now more consistent in how the prototypes are rendered. Some of these warnings would truncate prototypes containing nulls. In other cases one warning would suppress another. The warning about illegal characters in prototypes no longer says "after '_'" if the bad character came before the underscore.

- Perl folding rules are not up-to-date for 0x%X; please use the perlbug utility to report; in regex; marked by <-- HERE in m/%s/

  This message is now only in the regexp category, and not in the deprecated category. It is still a default (i.e., severe) warning [perl #89648].

- %%s[%s] in scalar context better written as $%s[%s]

  This warning now occurs for any `%array[$index]` or `%hash{key}` known to be in scalar context at compile time. Previously it was worded ‘‘Scalar value %%s[%s] better written as $%s[%s]’’.

- Switch condition not recognized in regex; marked by <-- HERE in m/%s/:

  The description for this diagnostic has been extended to cover all cases where the warning may occur. Issues with the positioning of the arrow indicator have also been resolved.

- The error messages for `my($a?$b$c)` and `my(do{})` now mention ‘‘conditional expression’’ and ‘‘do block’’, respectively, instead of reading ’Can’t declare null operation in ‘‘my’’’.

- When `use re "debug"` executes a regex containing a backreference, the debugging output now shows what string is being matched.

- The now fatal error message `Character following "\c" must be ASCII` has been reworded as `Character following "\c" must be printable ASCII` to emphasize that in `\c`*X*, *X* must be a *printable (non-control)* ASCII character.

## Utility Changes

*a2p*

- A possible crash from an off-by-one error when trying to access before the beginning of a buffer has been fixed. [perl #120244]

*bisect.pl*

The git bisection tool *Porting/bisect.pl* has had many enhancements.

It is provided as part of the source distribution but not installed because it is not self-contained as it relies on being run from within a git checkout. Note also that it makes no attempt to fix tests, correct runtime bugs or make something useful to install − its purpose is to make minimal changes to get any historical revision of interest to build and run as close as possible to ‘‘as-was’’, and thereby make `git bisect` easy to use.

- Can optionally run the test case with a timeout.
- Can now run in-place in a clean git checkout.
- Can run the test case under `valgrind`.
- Can apply user supplied patches and fixes to the source checkout before building.
- Now has fixups to enable building several more historical ranges of bleadperl, which can be useful for pinpointing the origins of bugs or behaviour changes.

*find2perl*

- find2perl now handles `?` wildcards correctly. [perl #113054]

*perlbug*

- *perlbug* now has a `-p` option for attaching patches with a bug report.
- perlbug has been modified to supply the report template with CRLF line endings on Windows. [GH #13612] <https://github.com/Perl/perl5/issues/13612>
- perlbug now makes as few assumptions as possible about the encoding of the report. This will likely change in the future to assume UTF−8 by default but allow a user override.

## Configuration and Compilation

- The *Makefile.PL* for SDBM_File now generates a better *Makefile*, which avoids a race condition during parallel makes, which could cause the build to fail. This is the last known parallel make problem (on *nix platforms), and therefore we believe that a parallel make should now always be error free.

- *installperl* and *installman*'s option handling has been refactored to use Getopt::Long. Both are used by the *Makefile* `install` targets, and are not installed, so these changes are only likely to affect custom installation scripts.

  - Single letter options now also have long names.
  - Invalid options are now rejected.
  - Command line arguments that are not options are now rejected.
  - Each now has a `--help` option to display the usage message.

  The behaviour for all valid documented invocations is unchanged.

- Where possible, the build now avoids recursive invocations of *make* when building pure-Perl extensions, without removing any parallelism from the build. Currently around 80 extensions can be processed directly by the *make_ext.pl* tool, meaning that 80 invocations of *make* and 160 invocations of *miniperl* are no longer made.

- The build system now works correctly when compiling under GCC or Clang with link-time optimization enabled (the `-flto` option). [perl #113022]

- Distinct library basenames with `d_libname_unique`.

  When compiling perl with this option, the library files for XS modules are named something ''unique'' — for example, Hash/Util/Util.so becomes Hash/Util/PL_Hash__Util.so. This behavior is similar to what currently happens on VMS, and serves as groundwork for the Android port.

- `sysroot` option to indicate the logical root directory under gcc and clang.

  When building with this option set, both Configure and the compilers search for all headers and libraries under this new sysroot, instead of /.

  This is a huge time saver if cross-compiling, but can also help on native builds if your toolchain's files have non-standard locations.

- The cross-compilation model has been renovated. There's several new options, and some backwards-incompatible changes:

  We now build binaries for miniperl and generate_uudmap to be used on the host, rather than running every miniperl call on the target; this means that, short of 'make test', we no longer need access to the target system once Configure is done. You can provide already-built binaries through the `hostperl` and `hostgenerate` options to Configure.

  Additionally, if targeting an EBCDIC platform from an ASCII host, or viceversa, you'll need to run Configure with `-Uhostgenerate`, to indicate that generate_uudmap should be run on the target.

  Finally, there's also a way of having Configure end early, right after building the host binaries, by cross-compiling without specifying a `targethost`.

  The incompatible changes include no longer using xconfig.h, xlib, or Cross.pm, so canned config files and Makefiles will have to be updated.

- Related to the above, there is now a way of specifying the location of sh (or equivalent) on the target system: `targetsh`.

  For example, Android has its sh in /system/bin/sh, so if cross-compiling from a more normal Unixy system with sh in /bin/sh, ''targetsh'' would end up as /system/bin/sh, and ''sh'' as /bin/sh.

- By default, **gcc** 4.9 does some optimizations that break perl. The **-fwrapv** option disables those optimizations (and probably others), so for **gcc** 4.3 and later (since the there might be similar problems lurking on older versions too, but **-fwrapv** was broken before 4.3, and the optimizations probably won't go away), *Configure* now adds **-fwrapv** unless the user requests **-fno-wrapv**, which disables **-fwrapv**, or **-fsanitize=undefined**, which turns the overflows **-fwrapv** ignores into runtime errors. [GH #13690] <https://github.com/Perl/perl5/issues/13690>

## Testing

- The `test.valgrind` make target now allows tests to be run in parallel. This target allows Perl's test suite to be run under Valgrind, which detects certain sorts of C programming errors, though at significant cost in running time. On suitable hardware, allowing parallel execution claws back a lot of that additional cost. [perl #121431]
- Various tests in *t/porting/* are no longer skipped when the perl *.git* directory is outside the perl tree and pointed to by `$GIT_DIR`. [perl #120505]
- The test suite no longer fails when the user's interactive shell maintains a `$PWD` environment variable, but the */bin/sh* used for running tests doesn't.

## Platform Support

### New Platforms

Android

Perl can now be built for Android, either natively or through cross-compilation, for all three currently available architectures (ARM, MIPS, and x86), on a wide range of versions.

Bitrig

Compile support has been added for Bitrig, a fork of OpenBSD.

FreeMiNT

Support has been added for FreeMiNT, a free open-source OS for the Atari ST system and its successors, based on the original MiNT that was officially adopted by Atari.

Synology

Synology ships its NAS boxes with a lean Linux distribution (DSM) on relative cheap CPU's (like the Marvell Kirkwood mv6282 − ARMv5tel or Freescale QorIQ P1022 ppc − e500v2) not meant for workstations or development. These boxes should build now. The basic problems are the non-standard location for tools.

### Discontinued Platforms

`sfio`

Code related to supporting the `sfio` I/O system has been removed.

Perl 5.004 added support to use the native API of `sfio`, AT&T's Safe/Fast I/O library. This code still built with v5.8.0, albeit with many regression tests failing, but was inadvertently broken before the v5.8.1 release, meaning that it has not worked on any version of Perl released since then. In over a decade we have received no bug reports about this, hence it is clear that no-one is using this functionality on any version of Perl that is still supported to any degree.

AT&T 3b1

Configure support for the 3b1, also known as the AT&T Unix PC (and the similar AT&T 7300), has been removed.

DG/UX

DG/UX was a Unix sold by Data General. The last release was in April 2001. It only runs on Data General's own hardware.

EBCDIC

In the absence of a regular source of smoke reports, code intended to support native EBCDIC platforms will be removed from perl before 5.22.0.

### Platform-Specific Notes

Cygwin

- **recv()** on a connected handle would populate the returned sender address with whatever happened to be in the working buffer. **recv()** now uses a workaround similar to the Win32 **recv()** wrapper and returns an empty string when **recvfrom**(2) doesn't modify the supplied address length. [perl #118843]
- Fixed a build error in cygwin.c on Cygwin 1.7.28.

  Tests now handle the errors that occur when `cygserver` isn't running.

GNU/Hurd

The BSD compatibility library `libbsd` is no longer required for builds.

Linux

The hints file now looks for `libgdbm_compat` only if `libgdbm` itself is also wanted. The former is never useful without the latter, and in some circumstances, including it could actually prevent building.

Mac OS

The build system now honors an `ld` setting supplied by the user running *Configure*.

MidnightBSD

`objformat` was removed from version 0.4−RELEASE of MidnightBSD and had been deprecated on earlier versions. This caused the build environment to be erroneously configured for `a.out` rather than `elf`. This has been now been corrected.

Mixed-endian platforms

The code supporting `pack` and `unpack` operations on mixed endian platforms has been removed. We believe that Perl has long been unable to build on mixed endian architectures (such as PDP−11s), so we don't think that this change will affect any platforms which were able to build v5.18.0.

VMS

- The `PERL_ENV_TABLES` feature to control the population of `%ENV` at perl start-up was broken in Perl 5.16.0 but has now been fixed.
- Skip access checks on remotes in **opendir()**. [perl #121002]
- A check for glob metacharacters in a path returned by the `glob()` operator has been replaced with a check for VMS wildcard characters. This saves a significant number of unnecessary `lstat()` calls such that some simple glob operations become 60−80% faster.

Win32

- `rename` and `link` on Win32 now set $! to ENOSPC and EDQUOT when appropriate. [perl #119857]
- The BUILD_STATIC and ALL_STATIC makefile options for linking some or (nearly) all extensions statically (into perl520.dll, and into a separate perl−static.exe too) were broken for MinGW builds. This has now been fixed.

  The ALL_STATIC option has also been improved to include the Encode and Win32 extensions (for both VC++ and MinGW builds).
- Support for building with Visual C++ 2013 has been added. There are currently two possible test failures (see ''Testing Perl on Windows'' in perlwin32) which will hopefully be resolved soon.
- Experimental support for building with Intel C++ Compiler has been added. The nmake makefile (win32/Makefile) and the dmake makefile (win32/makefile.mk) can be used. A ''nmake test'' will not pass at this time due to *cpan/CGI/t/url.t*.
- Killing a process tree with ''kill'' in perlfunc and a negative signal, was broken starting in 5.18.0. In this bug, `kill` always returned 0 for a negative signal even for valid PIDs, and no processes were terminated. This has been fixed [perl #121230].
- The time taken to build perl on Windows has been reduced quite significantly (time savings in the region of 30−40% are typically seen) by reducing the number of, usually failing, I/O calls for each `require()` (for **miniperl.exe** only). [GH #13566] <https://github.com/Perl/perl5/issues/13566>
- About 15 minutes of idle sleeping was removed from running `make test` due to a bug in which the timeout monitor used for tests could not be cancelled once the test completes, and the full timeout period elapsed before running the next test file. [GH #13647] <https://github.com/Perl/perl5/issues/13647>
- On a perl built without pseudo-fork (pseudo-fork builds were not affected by this bug), killing a process tree with `kill()` and a negative signal resulted in `kill()` inverting the returned value. For example, if `kill()` killed 1 process tree PID then it returned 0 instead of 1, and if `kill()` was passed 2 invalid PIDs then it returned 2 instead of 0. This has probably been the case since the process tree kill feature was implemented on Win32. It has now been corrected to follow the documented behaviour. [GH #13595]

<https://github.com/Perl/perl5/issues/13595>

- When building a 64−bit perl, an uninitialized memory read in **miniperl.exe**, used during the build process, could lead to a 4GB **wperl.exe** being created. This has now been fixed. (Note that **perl.exe** itself was unaffected, but obviously **wperl.exe** would have been completely broken.) [GH #13677] <https://github.com/Perl/perl5/issues/13677>
- Perl can now be built with **gcc** version 4.8.1 from <http://www.mingw.org>. This was previously broken due to an incorrect definition of **DllMain()** in one of perl's source files. Earlier **gcc** versions were also affected when using version 4 of the w32api package. Versions of **gcc** available from <http://mingw−w64.sourceforge.net/> were not affected. [GH #13733] <https://github.com/Perl/perl5/issues/13733>
- The test harness now has no failures when perl is built on a FAT drive with the Windows OS on an NTFS drive. [GH #6348] <https://github.com/Perl/perl5/issues/6348>
- When cloning the context stack in **fork()** emulation, **Perl_cx_dup()** would crash accessing parameter information for context stack entries that included no parameters, as with `&foo;`. [GH #13763] <https://github.com/Perl/perl5/issues/13763>
- Introduced by [GH #12161] <https://github.com/Perl/perl5/issues/12161>, a memory leak on every call to `system` and backticks ( `` ), on most Win32 Perls starting from 5.18.0 has been fixed. The memory leak only occurred if you enabled pseudo-fork in your build of Win32 Perl, and were running that build on Server 2003 R2 or newer OS. The leak does not appear on WinXP SP3. [GH #13741] <https://github.com/Perl/perl5/issues/13741>

WinCE

- The building of XS modules has largely been restored. Several still cannot (yet) be built but it is now possible to build Perl on WinCE with only a couple of further patches (to Socket and ExtUtils::MakeMaker), hopefully to be incorporated soon.
- Perl can now be built in one shot with no user intervention on WinCE by running `nmake −f Makefile.ce all`.

  Support for building with EVC (Embedded Visual C++) 4 has been restored. Perl can also be built using Smart Devices for Visual C++ 2005 or 2008.

## Internal Changes

- The internal representation has changed for the match variables `$1`, `$2` etc., $‘, $&, $’, ${^PREMATCH}, ${^MATCH} and ${^POSTMATCH}. It uses slightly less memory, avoids string comparisons and numeric conversions during lookup, and uses 23 fewer lines of C. This change should not affect any external code.
- Arrays now use NULL internally to represent unused slots, instead of &PL_sv_undef. &PL_sv_undef is no longer treated as a special value, so av_store(av, 0, &PL_sv_undef) will cause element 0 of that array to hold a read-only undefined scalar. `$array[0] = anything` will croak and `\$array[0]` will compare equal to `\undef`.
- The SV returned by **HeSVKEY_force()** now correctly reflects the UTF8ness of the underlying hash key when that key is not stored as a SV. [perl #79074]
- Certain rarely used functions and macros available to XS code are now deprecated. These are: `utf8_to_uvuni_buf` (use `utf8_to_uvchr_buf` instead), `valid_utf8_to_uvuni` (use `utf8_to_uvchr_buf` instead), `NATIVE_TO_NEED` (this did not work properly anyway), and `ASCII_TO_NEED` (this did not work properly anyway).

  Starting in this release, almost never does application code need to distinguish between the platform's character set and Latin1, on which the lowest 256 characters of Unicode are based. New code should not use `utf8n_to_uvuni` (use `utf8_to_uvchr_buf` instead), nor `uvuni_to_utf8` (use `uvchr_to_utf8` instead),
- The Makefile shortcut targets for many rarely (or never) used testing and profiling targets have been removed, or merged into the only other Makefile target that uses them. Specifically, these targets are gone, along with documentation that referenced them or explained how to use them:

```
check.third check.utf16 check.utf8 coretest minitest.prep
minitest.utf16 perl.config.dashg perl.config.dashpg
perl.config.gcov perl.gcov perl.gprof perl.gprof.config
perl.pixie perl.pixie.atom perl.pixie.config perl.pixie.irix
perl.third perl.third.config perl.valgrind.config purecovperl
pureperl quantperl test.deparse test.taintwarn test.third
test.torture test.utf16 test.utf8 test_notty.deparse
test_notty.third test_notty.valgrind test_prep.third
test_prep.valgrind torturetest ucheck ucheck.third ucheck.utf16
ucheck.valgrind utest utest.third utest.utf16 utest.valgrind
```

It's still possible to run the relevant commands by ''hand'' − no underlying functionality has been removed.

- It is now possible to keep Perl from initializing locale handling. For the most part, Perl doesn't pay attention to locale. (See perllocale.) Nonetheless, until now, on startup, it has always initialized locale handling to the system default, just in case the program being executed ends up using locales. (This is one of the first things a locale-aware program should do, long before Perl knows if it will actually be needed or not.) This works well except when Perl is embedded in another application which wants a locale that isn't the system default. Now, if the environment variable `PERL_SKIP_LOCALE_INIT` is set at the time Perl is started, this initialization step is skipped. Prior to this, on Windows platforms, the only workaround for this deficiency was to use a hacked-up copy of internal Perl code. Applications that need to use older Perls can discover if the embedded Perl they are using needs the workaround by testing that the C preprocessor symbol `HAS_SKIP_LOCALE_INIT` is not defined. [RT #38193]
- `BmRARE` and `BmPREVIOUS` have been removed. They were not used anywhere and are not part of the API. For XS modules, they are now #defined as 0.
- `sv_force_normal`, which usually croaks on read-only values, used to allow read-only values to be modified at compile time. This has been changed to croak on read-only values regardless. This change uncovered several core bugs.
- Perl's new copy-on-write mechanism (which is now enabled by default), allows any `SvPOK` scalar to be automatically upgraded to a copy-on-write scalar when copied. A reference count on the string buffer is stored in the string buffer itself.

  For example:

```
$ perl -MDevel::Peek -e'$a="abc"; $b = $a; Dump $a; Dump $b'
SV = PV(0x260cd80) at 0x2620ad8
  REFCNT = 1
  FLAGS = (POK,IsCOW,pPOK)
  PV = 0x2619bc0 "abc"\0
  CUR = 3
  LEN = 16
  COW_REFCNT = 1
SV = PV(0x260ce30) at 0x2620b20
  REFCNT = 1
  FLAGS = (POK,IsCOW,pPOK)
  PV = 0x2619bc0 "abc"\0
  CUR = 3
  LEN = 16
  COW_REFCNT = 1
```

  Note that both scalars share the same PV buffer and have a COW_REFCNT greater than zero.

  This means that XS code which wishes to modify the `SvPVX()` buffer of an SV should call `SvPV_force()` or similar first, to ensure a valid (and unshared) buffer, and to call `SvSETMAGIC()` afterwards. This in fact has always been the case (for example hash keys were already copy-on-write); this change just spreads the COW behaviour to a wider variety of SVs.

  One important difference is that before 5.18.0, shared hash-key scalars used to have the `SvREADONLY` flag set; this is no longer the case.

This new behaviour can still be disabled by running *Configure* with **−Accflags=−DPERL_NO_COW**. This option will probably be removed in Perl 5.22.

- `PL_sawampersand` is now a constant. The switch this variable provided (to enable/disable the pre-match copy depending on whether `$&` had been seen) has been removed and replaced with copy-on-write, eliminating a few bugs.

  The previous behaviour can still be enabled by running *Configure* with **−Accflags=−DPERL_SAWAMPERSAND**.

- The functions `my_swap`, `my_htonl` and `my_ntohl` have been removed. It is unclear why these functions were ever marked as *A*, part of the API. XS code can't call them directly, as it can't rely on them being compiled. Unsurprisingly, no code on CPAN references them.

- The signature of the `Perl_re_intuit_start()` regex function has changed; the function pointer `intuit` in the regex engine plugin structure has also changed accordingly. A new parameter, `strbeg` has been added; this has the same meaning as the same-named parameter in `Perl_regexec_flags`. Previously intuit would try to guess the start of the string from the passed SV (if any), and would sometimes get it wrong (e.g. with an overloaded SV).

- The signature of the `Perl_regexec_flags()` regex function has changed; the function pointer `exec` in the regex engine plugin structure has also changed to match. The `minend` parameter now has type `SSize_t` to better support 64−bit systems.

- XS code may use various macros to change the case of a character or code point (for example `toLOWER_utf8()`). Only a couple of these were documented until now; and now they should be used in preference to calling the underlying functions. See ''Character case changing'' in perlapi.

- The code dealt rather inconsistently with uids and gids. Some places assumed that they could be safely stored in UVs, others in IVs, others in ints. Four new macros are introduced: **SvUID()**, **sv_setuid()**, **SvGID()**, and **sv_setgid()**

- `sv_pos_b2u_flags` has been added to the API. It is similar to `sv_pos_b2u`, but supports long strings on 64−bit platforms.

- `PL_exit_flags` can now be used by perl embedders or other XS code to have perl `warn` or `abort` on an attempted exit. [perl #52000]

- Compiling with `−Accflags=−PERL_BOOL_AS_CHAR` now allows C99 and C++ compilers to emulate the aliasing of `bool` to `char` that perl does for C89 compilers. [perl #120314]

- The `sv` argument in ''sv_2pv_flags'' in perlapi, ''sv_2iv_flags'' in perlapi, ''sv_2uv_flags'' in perlapi, and ''sv_2nv_flags'' in perlapi and their older wrappers sv_2pv, sv_2iv, sv_2uv, sv_2nv, is now non-NULL. Passing NULL now will crash. When the non-NULL marker was introduced en masse in 5.9.3 the functions were marked non-NULL, but since the creation of the SV API in 5.0 alpha 2, if NULL was passed, the functions returned 0 or false-type values. The code that supports `sv` argument being non-NULL dates to 5.0 alpha 2 directly, and indirectly to Perl 1.0 (pre 5.0 api). The lack of documentation that the functions accepted a NULL `sv` was corrected in 5.11.0 and between 5.11.0 and 5.19.5 the functions were marked NULLOK. As an optimization the NULLOK code has now been removed, and the functions became non-NULL marked again, because core getter-type macros never pass NULL to these functions and would crash before ever passing NULL.

  The only way a NULL `sv` can be passed to sv_2*v* functions is if XS code directly calls sv_2*v*. This is unlikely as XS code uses Sv*V* macros to get the underlying value out of the SV. One possible situation which leads to a NULL `sv` being passed to sv_2*v* functions, is if XS code defines its own getter type Sv*V* macros, which check for NULL **before** dereferencing and checking the SV's flags through public API Sv*OK* macros or directly using private API `SvFLAGS`, and if `sv` is NULL, then calling the sv_2*v functions with a NULL literal or passing the `sv` containing a NULL value.

- newATTRSUB is now a macro

  The public API newATTRSUB was previously a macro to the private function Perl_newATTRSUB. Function Perl_newATTRSUB has been removed. newATTRSUB is now macro to a different internal function.

- Changes in warnings raised by `utf8n_to_uvchr()`

  This bottom level function decodes the first character of a UTF−8 string into a code point. It is accessible to XS level code, but it's discouraged from using it directly. There are higher level functions that call this that should be used instead, such as ''utf8_to_uvchr_buf'' in perlapi. For completeness though, this documents some changes to it. Now, tests for malformations are done before any tests for other potential issues. One of those issues involves code points so large that they have never appeared in any official standard (the current standard has scaled back the highest acceptable code point from earlier versions). It is possible (though not done in CPAN) to warn and/or forbid these code points, while accepting smaller code points that are still above the legal Unicode maximum. The warning message for this now includes the code point if representable on the machine. Previously it always displayed raw bytes, which is what it still does for non-representable code points.

- Regexp engine changes that affect the pluggable regex engine interface

  Many flags that used to be exposed via regexp.h and used to populate the extflags member of struct regexp have been removed. These fields were technically private to Perl's own regexp engine and should not have been exposed there in the first place.

  The affected flags are:

  ```
  RXf_NOSCAN
  RXf_CANY_SEEN
  RXf_GPOS_SEEN
  RXf_GPOS_FLOAT
  RXf_ANCH_BOL
  RXf_ANCH_MBOL
  RXf_ANCH_SBOL
  RXf_ANCH_GPOS
  ```

  As well as the follow flag masks:

  ```
  RXf_ANCH_SINGLE
  RXf_ANCH
  ```

  All have been renamed to PREGf_ equivalents and moved to regcomp.h.

  The behavior previously achieved by setting one or more of the RXf_ANCH_ flags (via the RXf_ANCH mask) have now been replaced by a *single* flag bit in extflags:

  ```
  RXf_IS_ANCHORED
  ```

  pluggable regex engines which previously used to set these flags should now set this flag ALONE.

- The Perl core now consistently uses `av_tindex()` (''the top index of an array'') as a more clearly-named synonym for `av_len()`.

- The obscure interpreter variable `PL_timesbuf` is expected to be removed early in the 5.21.x development series, so that Perl 5.22.0 will not provide it to XS authors. While the variable still exists in 5.20.0, we hope that this advance warning of the deprecation will help anyone who is using that variable.

## Selected Bug Fixes

### Regular Expressions

- Fixed a small number of regexp constructions that could either fail to match or crash perl when the string being matched against was allocated above the 2GB line on 32−bit systems. [RT #118175]

- Various memory leaks involving the parsing of the `(?[...])` regular expression construct have been fixed.

- `(?[...])` now allows interpolation of precompiled patterns consisting of `(?[...])` with bracketed character classes inside (`$pat = qr/(?[ [a] ])/; /(?[ $pat ])/`). Formerly, the brackets would confuse the regular expression parser.

- The ''Quantifier unexpected on zero-length expression'' warning message could appear twice starting in Perl v5.10 for a regular expression also containing alternations (e.g., ''a|b'') triggering the trie optimisation.

- Perl v5.18 inadvertently introduced a bug whereby interpolating mixed up- and down-graded UTF-8 strings in a regex could result in malformed UTF-8 in the pattern: specifically if a downgraded character in the range `\x80..\xff` followed a UTF-8 string, e.g.

```
utf8::upgrade(  my $u = "\x{e5}");
utf8::downgrade(my $d = "\x{e5}");
/$u$d/
```

  [RT #118297]

- In regular expressions containing multiple code blocks, the values of `$1`, `$2`, etc., set by nested regular expression calls would leak from one block to the next. Now these variables always refer to the outer regular expression at the start of an embedded block [perl #117917].
- `/$qr/p` was broken in Perl 5.18.0; the `/p` flag was ignored. This has been fixed. [perl #118213]
- Starting in Perl 5.18.0, a construct like `/[#](?{})/x` would have its # incorrectly interpreted as a comment. The code block would be skipped, unparsed. This has been corrected.
- Starting in Perl 5.001, a regular expression like `/[#$a]/x` or `/[#]$a/x` would have its # incorrectly interpreted as a comment, so the variable would not interpolate. This has been corrected. [perl #45667]
- Perl 5.18.0 inadvertently made dereferenced regular expressions (`${ qr// }`) false as booleans. This has been fixed.
- The use of `\G` in regular expressions, where it's not at the start of the pattern, is now slightly less buggy (although it is still somewhat problematic).
- Where a regular expression included code blocks (`/(?{...})/`), and where the use of constant overloading triggered a re-compilation of the code block, the second compilation didn't see its outer lexical scope. This was a regression in Perl 5.18.0.
- The string position set by `pos` could shift if the string changed representation internally to or from utf8. This could happen, e.g., with references to objects with string overloading.
- Taking references to the return values of two `pos` calls with the same argument, and then assigning a reference to one and `undef` to the other, could result in assertion failures or memory leaks.
- Elements of @- and @+ now update correctly when they refer to non-existent captures. Previously, a referenced element (`$ref = \$-[1]`) could refer to the wrong match after subsequent matches.
- The code that parses regex backrefs (or ambiguous backref/octals) such as \123 did a simple **atoi()**, which could wrap round to negative values on long digit strings and cause segmentation faults. This has now been fixed. [perl #119505]
- Assigning another typeglob to `*^R` no longer makes the regular expression engine crash.
- The `\N` regular expression escape, when used without the curly braces (to mean `[^\n]`), was ignoring a following `*` if followed by whitespace under /x. It had been this way since `\N` to mean `[^\n]` was introduced in 5.12.0.
- `s///`, `tr///` and `y///` now work when a wide character is used as the delimiter. [perl #120463]
- Some cases of unterminated (?...) sequences in regular expressions (e.g., `/(?</`) have been fixed to produce the proper error message instead of ''panic: memory wrap''. Other cases (e.g., `/(?(/`) have yet to be fixed.
- When a reference to a reference to an overloaded object was returned from a regular expression `(??{...})` code block, an incorrect implicit dereference could take place if the inner reference had been returned by a code block previously.
- A tied variable returned from `(??{...})` sees the inner values of match variables (i.e., the `$1` etc. from any matches inside the block) in its FETCH method. This was not the case if a reference to an overloaded object was the last thing assigned to the tied variable. Instead, the match variables referred to the outer pattern during the FETCH call.

- Fix unexpected tainting via regexp using locale. Previously, under certain conditions, the use of character classes could cause tainting when it shouldn't. Some character classes are locale-dependent, but before this patch, sometimes tainting was happening even for character classes that don't depend on the locale. [perl #120675]
- Under certain conditions, Perl would throw an error if in a lookbehind assertion in a regexp, the assertion referred to a named subpattern, complaining the lookbehind was variable when it wasn't. This has been fixed. [perl #120600], [perl #120618]. The current fix may be improved on in the future.
- `$^R` wasn't available outside of the regular expression that initialized it. [perl #121070]
- A large set of fixes and refactoring for **re_intuit_start()** was merged, the highlights are:
  - Fixed a panic when compiling the regular expression `/\x{100}[xy]\x{100}{2}/`.
  - Fixed a performance regression when performing a global pattern match against a UTF-8 string. [perl #120692]
  - Fixed another performance issue where matching a regular expression like `/ab.{1,2}x/` against a long UTF-8 string would unnecessarily calculate byte offsets for a large portion of the string. [perl #120692]
- Fixed an alignment error when compiling regular expressions when built with GCC on HP-UX 64-bit.
- On 64-bit platforms `pos` can now be set to a value higher than 2**31-1. [perl #72766]

### Perl 5 Debugger and -d

- The debugger's `man` command been fixed. It was broken in the v5.18.0 release. The `man` command is aliased to the names `doc` and `perldoc` - all now work again.
- `@_` is now correctly visible in the debugger, fixing a regression introduced in v5.18.0's debugger. [RT #118169]
- Under copy-on-write builds (the default as of 5.20.0) `${'_<-e'}[0]` no longer gets mangled. This is the first line of input saved for the debugger's use for one-liners [perl #118627].
- On non-threaded builds, setting `${"_<filename"}` to a reference or typeglob no longer causes `__FILE__` and some error messages to produce a corrupt string, and no longer prevents `#line` directives in string evals from providing the source lines to the debugger. Threaded builds were unaffected.
- Starting with Perl 5.12, line numbers were off by one if the **-d** switch was used on the #! line. Now they are correct.
- `*DB::DB = sub {} if 0` no longer stops Perl's debugging mode from finding `DB::DB` subs declared thereafter.
- `%{'_<...'}` hashes now set breakpoints on the corresponding `@{'_<...'}` rather than whichever array `@DB::dbline` is aliased to. [perl #119799]
- Call set-magic when setting `$DB::sub`. [perl #121255]
- The debugger's ''n'' command now respects lvalue subroutines and steps over them [perl #118839].

### Lexical Subroutines

- Lexical constants (`my sub a() { 42 }`) no longer crash when inlined.
- Parameter prototypes attached to lexical subroutines are now respected when compiling sub calls without parentheses. Previously, the prototypes were honoured only for calls *with* parentheses. [RT #116735]
- Syntax errors in lexical subroutines in combination with calls to the same subroutines no longer cause crashes at compile time.
- Deep recursion warnings no longer crash lexical subroutines. [RT #118521]
- The dtrace sub-entry probe now works with lexical subs, instead of crashing [perl #118305].

- Undefining an inlinable lexical subroutine (`my sub foo() { 42 } undef &foo`) would result in a crash if warnings were turned on.
- An undefined lexical sub used as an inherited method no longer crashes.
- The presence of a lexical sub named ‘‘CORE’’ no longer stops the CORE:: prefix from working.

### Everything Else

- The OP allocation code now returns correctly aligned memory in all cases for `struct pmop`. Previously it could return memory only aligned to a 4−byte boundary, which is not correct for an ithreads build with 64 bit IVs on some 32 bit platforms. Notably, this caused the build to fail completely on sparc GNU/Linux. [RT #118055]
- Evaluating large hashes in scalar context is now much faster, as the number of used chains in the hash is now cached for larger hashes. Smaller hashes continue not to store it and calculate it when needed, as this saves one IV. That would be 1 IV overhead for every object built from a hash. [RT #114576]
- Perl v5.16 inadvertently introduced a bug whereby calls to XSUBs that were not visible at compile time were treated as lvalues and could be assigned to, even when the subroutine was not an lvalue sub. This has been fixed. [RT #117947]
- In Perl v5.18.0 dualvars that had an empty string for the string part but a non-zero number for the number part starting being treated as true. In previous versions they were treated as false, the string representation taking precedence. The old behaviour has been restored. [RT #118159]
- Since Perl v5.12, inlining of constants that override built-in keywords of the same name had countermanded `use subs`, causing subsequent mentions of the constant to use the built-in keyword instead. This has been fixed.
- The warning produced by `−l $handle` now applies to IO refs and globs, not just to glob refs. That warning is also now UTF8−clean. [RT #117595]
- `delete local $ENV{nonexistent_env_var}` no longer leaks memory.
- `sort` and `require` followed by a keyword prefixed with `CORE::` now treat it as a keyword, and not as a subroutine or module name. [RT #24482]
- Through certain conundrums, it is possible to cause the current package to be freed. Certain operators (`bless`, `reset`, `open`, `eval`) could not cope and would crash. They have been made more resilient. [RT #117941]
- Aliasing filehandles through glob-to-glob assignment would not update internal method caches properly if a package of the same name as the filehandle existed, resulting in filehandle method calls going to the package instead. This has been fixed.
- `./Configure −de −Dusevendorprefix` didn’t default. [RT #64126]
- The `Statement unlikely to be reached` warning was listed in perldiag as an `exec`−category warning, but was enabled and disabled by the `syntax` category. On the other hand, the `exec` category controlled its fatal-ness. It is now entirely handled by the `exec` category.
- The ‘‘Replacement list is longer that search list’’ warning for `tr///` and `y///` no longer occurs in the presence of the `/c` flag. [RT #118047]
- Stringification of NVs are not cached so that the lexical locale controls stringification of the decimal point. [perl #108378] [perl #115800]
- There have been several fixes related to Perl’s handling of locales. perl #38193 was described above in ‘‘Internal Changes’’. Also fixed is #118197, where the radix (decimal point) character had to be an ASCII character (which doesn’t work for some non-Western languages); and #115808, in which `POSIX::setlocale()` on failure returned an `undef` which didn’t warn about not being defined even if those warnings were enabled.
- Compiling a `split` operator whose third argument is a named constant evaluating to 0 no longer causes the constant’s value to change.

- A named constant used as the second argument to `index` no longer gets coerced to a string if it is a reference, regular expression, dualvar, etc.
- A named constant evaluating to the undefined value used as the second argument to `index` no longer produces ''uninitialized'' warnings at compile time. It will still produce them at run time.
- When a scalar was returned from a subroutine in `@INC`, the referenced scalar was magically converted into an IO thingy, possibly resulting in ''Bizarre copy'' errors if that scalar continued to be used elsewhere. Now Perl uses an internal copy of the scalar instead.
- Certain uses of the `sort` operator are optimised to modify an array in place, such as `@a = sort @a`. During the sorting, the array is made read-only. If a sort block should happen to die, then the array remained read-only even outside the `sort`. This has been fixed.
- `$a` and `$b` inside a sort block are aliased to the actual arguments to `sort`, so they can be modified through those two variables. This did not always work, e.g., for lvalue subs and `$#ary`, and probably many other operators. It works now.
- The arguments to `sort` are now all in list context. If the `sort` itself were called in void or scalar context, then *some*, but not all, of the arguments used to be in void or scalar context.
- Subroutine prototypes with Unicode characters above U+00FF were getting mangled during closure cloning. This would happen with subroutines closing over lexical variables declared outside, and with lexical subs.
- `UNIVERSAL::can` now treats its first argument the same way that method calls do: Typeglobs and glob references with non-empty IO slots are treated as handles, and strings are treated as filehandles, rather than packages, if a handle with that name exists [perl #113932].
- Method calls on typeglobs (e.g., `*ARGV->getline`) used to stringify the typeglob and then look it up again. Combined with changes in Perl 5.18.0, this allowed `*foo->bar` to call methods on the ''foo'' package (like `foo->bar`). In some cases it could cause the method to be called on the wrong handle. Now a typeglob argument is treated as a handle (just like `(\*foo)->bar`), or, if its IO slot is empty, an error is raised.
- Assigning a vstring to a tied variable or to a subroutine argument aliased to a nonexistent hash or array element now works, without flattening the vstring into a regular string.
- `pos`, `tie`, `tied` and `untie` did not work properly on subroutine arguments aliased to nonexistent hash and array elements [perl #77814, #27010].
- The `=>` fat arrow operator can now quote built-in keywords even if it occurs on the next line, making it consistent with how it treats other barewords.
- Autovivifying a subroutine stub via `\&$glob` started causing crashes in Perl 5.18.0 if the `$glob` was merely a copy of a real glob, i.e., a scalar that had had a glob assigned to it. This has been fixed. [perl #119051]
- Perl used to leak an implementation detail when it came to referencing the return values of certain operators. `for ($a+$b) { warn \$_; warn \$_ }` used to display two different memory addresses, because the `\` operator was copying the variable. Under threaded builds, it would also happen for constants (`for(1) { ... }`). This has been fixed. [perl #21979, #78194, #89188, #109746, #114838, #115388]
- The range operator `..` was returning the same modifiable scalars with each call, unless it was the only thing in a `foreach` loop header. This meant that changes to values within the list returned would be visible the next time the operator was executed. [perl #3105]
- Constant folding and subroutine inlining no longer cause operations that would normally return new modifiable scalars to return read-only values instead.
- Closures of the form `sub () { $some_variable }` are no longer inlined, causing changes to the variable to be ignored by callers of the subroutine. [perl #79908]
- Return values of certain operators such as `ref` would sometimes be shared between recursive calls to the same subroutine, causing the inner call to modify the value returned by `ref` in the outer call. This has been fixed.

- `__PACKAGE__` and constants returning a package name or hash key are now consistently read-only. In various previous Perl releases, they have become mutable under certain circumstances.
- Enabling ‘‘used once’’ warnings no longer causes crashes on stash circularities created at compile time (`*Foo::Bar::Foo:: = *Foo::`).
- Undef constants used in hash keys (`use constant u => undef; $h{+u}`) no longer produce ‘‘uninitialized’’ warnings at compile time.
- Modifying a substitution target inside the substitution replacement no longer causes crashes.
- The first statement inside a string eval used to use the wrong pragma setting sometimes during constant folding. `eval 'uc chr 0xe0'` would randomly choose between Unicode, byte, and locale semantics. This has been fixed.
- The handling of return values of `@INC` filters (subroutines returned by subroutines in `@INC`) has been fixed in various ways. Previously tied variables were mishandled, and setting `$_` to a reference or typeglob could result in crashes.
- The `SvPVbyte` XS function has been fixed to work with tied scalars returning something other than a string. It used to return utf8 in those cases where `SvPV` would.
- Perl 5.18.0 inadvertently made `--` and `++` crash on dereferenced regular expressions, and stopped `++` from flattening vstrings.
- `bless` no longer dies with ‘‘Can’t bless non-reference value’’ if its first argument is a tied reference.
- `reset` with an argument no longer skips copy-on-write scalars, regular expressions, typeglob copies, and vstrings. Also, when encountering those or read-only values, it no longer skips any array or hash with the same name.
- `reset` with an argument now skips scalars aliased to typeglobs (`for $z (*foo) { reset "z" }`). Previously it would corrupt memory or crash.
- `ucfirst` and `lcfirst` were not respecting the bytes pragma. This was a regression from Perl 5.12. [perl #117355]
- Changes to `UNIVERSAL::DESTROY` now update DESTROY caches in all classes, instead of causing classes that have already had objects destroyed to continue using the old sub. This was a regression in Perl 5.18. [perl #114864]
- All known false-positive occurrences of the deprecation warning ‘‘Useless use of ’\’; doesn’t escape metacharacter ’%c’’’, added in Perl 5.18.0, have been removed. [perl #119101]
- The value of $^E is now saved across signal handlers on Windows. [perl #85104]
- A lexical filehandle (as in `open my $fh...`) is usually given a name based on the current package and the name of the variable, e.g. ‘‘main::$fh’’. Under recursion, the filehandle was losing the ‘‘$fh’’ part of the name. This has been fixed.
- Uninitialized values returned by XSUBs are no longer exempt from uninitialized warnings. [perl #118693]
- `elsif ("")` no longer erroneously produces a warning about void context. [perl #118753]
- Passing `undef` to a subroutine now causes `@_` to contain the same read-only undefined scalar that `undef` returns. Furthermore, `exists $_[0]` will now return true if `undef` was the first argument. [perl #7508, #109726]
- Passing a non-existent array element to a subroutine does not usually autovivify it unless the subroutine modifies its argument. This did not work correctly with negative indices and with non-existent elements within the array. The element would be vivified immediately. The delayed vivification has been extended to work with those. [perl #118691]
- Assigning references or globs to the scalar returned by $#foo after the `@foo` array has been freed no longer causes assertion failures on debugging builds and memory leaks on regular builds.
- On 64-bit platforms, large ranges like 1..1000000000000 no longer crash, but eat up all your memory instead. [perl #119161]

- `__DATA__` now puts the `DATA` handle in the right package, even if the current package has been renamed through glob assignment.
- When `die`, `last`, `next`, `redo`, `goto` and `exit` unwind the scope, it is possible for `DESTROY` recursively to call a subroutine or format that is currently being exited. It that case, sometimes the lexical variables inside the sub would start out having values from the outer call, instead of being undefined as they should. This has been fixed. [perl #119311]
- ${^MPEN} is no longer treated as a synonym for ${^MATCH}.
- Perl now tries a little harder to return the correct line number in `(caller)[2]`. [perl #115768]
- Line numbers inside multiline quote-like operators are now reported correctly. [perl #3643]
- `#line` directives inside code embedded in quote-like operators are now respected.
- Line numbers are now correct inside the second here-doc when two here-doc markers occur on the same line.
- An optimization in Perl 5.18 made incorrect assumptions causing a bad interaction with the Devel::CallParser CPAN module. If the module was loaded then lexical variables declared in separate statements following a `my(...)` list might fail to be cleared on scope exit.
- `&xsub` and `goto &xsub` calls now allow the called subroutine to autovivify elements of `@_`.
- `&xsub` and `goto &xsub` no longer crash if *_ has been undefined and has no ARRAY entry (i.e. `@_` does not exist).
- `&xsub` and `goto &xsub` now work with tied `@_`.
- Overlong identifiers no longer cause a buffer overflow (and a crash). They started doing so in Perl 5.18.
- The warning ‘‘Scalar value `@hash`{foo} better written as `$hash`{foo}’’ now produces far fewer false positives. In particular, `@hash{+function_returning_a_list}` and `@hash{ qw "foo bar baz" }` no longer warn. The same applies to array slices. [perl #28380, #114024]
- `$! = EINVAL; waitpid(0, WNOHANG);` no longer goes into an internal infinite loop. [perl #85228]
- A possible segmentation fault in filehandle duplication has been fixed.
- A subroutine in `@INC` can return a reference to a scalar containing the initial contents of the file. However, that scalar was freed prematurely if not referenced elsewhere, giving random results.
- `last` no longer returns values that the same statement has accumulated so far, fixing amongst other things the long-standing bug that `push @a, last` would try to return the `@a`, copying it like a scalar in the process and resulting in the error, ‘‘Bizarre copy of ARRAY in last.’’ [perl #3112]
- In some cases, closing file handles opened to pipe to or from a process, which had been duplicated into a standard handle, would call perl’s internal waitpid wrapper with a pid of zero. With the fix for [perl #85228] this zero pid was passed to `waitpid`, possibly blocking the process. This wait for process zero no longer occurs. [perl #119893]
- `select` used to ignore magic on the fourth (timeout) argument, leading to effects such as `select` blocking indefinitely rather than the expected sleep time. This has now been fixed. [perl #120102]
- The class name in `for my class $foo` is now parsed correctly. In the case of the second character of the class name being followed by a digit (e.g. ’a1b’) this used to give the error ‘‘Missing $ on loop variable’’. [perl #120112]
- Perl 5.18.0 accidentally disallowed `-bareword` under `use strict` and `use integer`. This has been fixed. [perl #120288]
- `-a` at the start of a line (or a hyphen with any single letter that is not a filetest operator) no longer produces an erroneous ’Use of ‘‘-a’’ without parentheses is ambiguous’ warning. [perl #120288]
- Lvalue context is now properly propagated into bare blocks and `if` and `else` blocks in lvalue subroutines. Previously, arrays and hashes would sometimes incorrectly be flattened when returned in lvalue list context, or ‘‘Bizarre copy’’ errors could occur. [perl #119797]

- Lvalue context is now propagated to the branches of `||` and `&&` (and their alphabetic equivalents, `or` and `and`). This means `foreach (pos $x || pos $y) {...}` now allows `pos` to be modified through `$_`.
- `stat` and `readline` remember the last handle used; the former for the special _ filehandle, the latter for `${^LAST_FH}`. `eval "*foo if 0"` where *foo was the last handle passed to `stat` or `readline` could cause that handle to be forgotten if the handle were not opened yet. This has been fixed.
- Various cases of `delete $::{a}`, `delete $::{ENV}` etc. causing a crash have been fixed. [perl #54044]
- Setting `$!` to EACCESS before calling `require` could affect `require`'s behaviour. This has been fixed.
- The ''Can't use \1 to mean `$1` in expression'' warning message now only occurs on the right-hand (replacement) part of a substitution. Formerly it could happen in code embedded in the left-hand side, or in any other quote-like operator.
- Blessing into a reference (`bless $thisref, $thatref`) has long been disallowed, but magical scalars for the second like `$/` and those tied were exempt. They no longer are. [perl #119809]
- Blessing into a reference was accidentally allowed in 5.18 if the class argument were a blessed reference with stale method caches (i.e., whose class had had subs defined since the last method call). They are disallowed once more, as in 5.16.
- `$x->{key}` where `$x` was declared as `my Class $x` no longer crashes if a Class::FIELDS subroutine stub has been declared.
- `@$obj{'key'}` and `${$obj}{key}` used to be exempt from compile-time field checking (''No such class field''; see fields) but no longer are.
- A nonexistent array element with a large index passed to a subroutine that ties the array and then tries to access the element no longer results in a crash.
- Declaring a subroutine stub named NEGATIVE_INDICES no longer makes negative array indices crash when the current package is a tied array class.
- Declaring a `require`, `glob`, or `do` subroutine stub in the CORE::GLOBAL:: package no longer makes compilation of calls to the corresponding functions crash.
- Aliasing CORE::GLOBAL:: functions to constants stopped working in Perl 5.10 but has now been fixed.
- When `` `...` `` or `qx/.../` calls a `readpipe` override, double-quotish interpolation now happens, as is the case when there is no override. Previously, the presence of an override would make these quote-like operators act like `q{}`, suppressing interpolation. [perl #115330]
- `<<<`...`` here-docs (with backticks as the delimiters) now call `readpipe` overrides. [perl #119827]
- `&CORE::exit()` and `&CORE::die()` now respect vmsish hints.
- Undefining a glob that triggers a DESTROY method that undefines the same glob is now safe. It used to produce ''Attempt to free unreferenced glob pointer'' warnings and leak memory.
- If subroutine redefinition (`eval 'sub foo{}'` or `newXS` for XS code) triggers a DESTROY method on the sub that is being redefined, and that method assigns a subroutine to the same slot (`*foo = sub {}`), `$_[0]` is no longer left pointing to a freed scalar. Now DESTROY is delayed until the new subroutine has been installed.
- On Windows, perl no longer calls **CloseHandle()** on a socket handle. This makes debugging easier on Windows by removing certain irrelevant bad handle exceptions. It also fixes a race condition that made socket functions randomly fail in a Perl process with multiple OS threads, and possible test failures in *dist/IO/t/cachepropagate-tcp.t*. [perl #120091/118059]
- Formats involving UTF-8 encoded strings, or strange vars like ties, overloads, or stringified refs (and in recent perls, pure NOK vars) would generally do the wrong thing in formats when the var is treated as a string and repeatedly chopped, as in `^<<<~~` and similar. This has now been

resolved. [perl #33832/45325/113868/119847/119849/119851]

- `semctl(..., SETVAL, ...)` would set the semaphore to the top 32−bits of the supplied integer instead of the bottom 32−bits on 64−bit big-endian systems. [perl #120635]
- `readdir()` now only sets `$!` on error. `$!` is no longer set to `EBADF` when then terminating `undef` is read from the directory unless the system call sets `$!`. [perl #118651]
- `&CORE::glob` no longer causes an intermittent crash due to perl's stack getting corrupted. [perl #119993]
- `open` with layers that load modules (e.g., "<:encoding(utf8)") no longer runs the risk of crashing due to stack corruption.
- Perl 5.18 broke autoloading via `->SUPER::foo` method calls by looking up AUTOLOAD from the current package rather than the current package's superclass. This has been fixed. [perl #120694]
- A longstanding bug causing `do {} until CONSTANT`, where the constant holds a true value, to read unallocated memory has been resolved. This would usually happen after a syntax error. In past versions of Perl it has crashed intermittently. [perl #72406]
- Fix HP-UX `$!` failure. HP-UX **strerror()** returns an empty string for an unknown error code. This caused an assertion to fail under DEBUGGING builds. Now instead, the returned string for `"$!"` contains text indicating the code is for an unknown error.
- Individually-tied elements of `@INC` (as in `tie $INC[0]...`) are now handled correctly. Formerly, whether a sub returned by such a tied element would be treated as a sub depended on whether a FETCH had occurred previously.
- `getc` on a byte-sized handle after the same `getc` operator had been used on a utf8 handle used to treat the bytes as utf8, resulting in erratic behavior (e.g., malformed UTF−8 warnings).
- An initial `{` at the beginning of a format argument line was always interpreted as the beginning of a block prior to v5.18. In Perl v5.18, it started being treated as an ambiguous token. The parser would guess whether it was supposed to be an anonymous hash constructor or a block based on the contents. Now the previous behaviour has been restored. [perl #119973]
- In Perl v5.18 `undef *_; goto &sub` and `local *_; goto &sub` started crashing. This has been fixed. [perl #119949]
- Backticks ( `` ` `` or `qx//` ) combined with multiple threads on Win32 could result in output sent to stdout on one thread being captured by backticks of an external command in another thread.

  This could occur for pseudo-forked processes too, as Win32's pseudo-fork is implemented in terms of threads. [perl #77672]
- `open $fh, ">+", undef` no longer leaks memory when TMPDIR is set but points to a directory a temporary file cannot be created in. [perl #120951]
- `for ( $h{k} || '' )` no longer auto-vivifies `$h{k}`. [perl #120374]
- On Windows machines, Perl now emulates the POSIX use of the environment for locale initialization. Previously, the environment was ignored. See "ENVIRONMENT" in perllocale.
- Fixed a crash when destroying a self-referencing GLOB. [perl #121242]

## Known Problems

- IO::Socket is known to fail tests on AIX 5.3. There is a patch <https://github.com/Perl/perl5/issues/13484> in the request tracker, #120835, which may be applied to future releases.
- The following modules are known to have test failures with this version of Perl. Patches have been submitted, so there will hopefully be new releases soon:
  - Data::Structure::Util version 0.15
  - HTML::StripScripts version 1.05

- List::Gather version 0.08.

## Obituary

Diana Rosa, 27, of Rio de Janeiro, went to her long rest on May 10, 2014, along with the plush camel she kept hanging on her computer screen all the time. She was a passionate Perl hacker who loved the language and its community, and who never missed a Rio.pm event. She was a true artist, an enthusiast about writing code, singing arias and graffiting walls. We'll never forget you.

Greg McCarroll died on August 28, 2013.

Greg was well known for many good reasons. He was one of the organisers of the first YAPC::Europe, which concluded with an unscheduled auction where he frantically tried to raise extra money to avoid the conference making a loss. It was Greg who mistakenly arrived for a london.pm meeting a week late; some years later he was the one who sold the choice of official meeting date at a YAPC::Europe auction, and eventually as glorious leader of london.pm he got to inherit the irreverent confusion that he had created.

Always helpful, friendly and cheerfully optimistic, you will be missed, but never forgotten.

## Acknowledgements

Perl 5.20.0 represents approximately 12 months of development since Perl 5.18.0 and contains approximately 470,000 lines of changes across 2,900 files from 124 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 280,000 lines of changes to 1,800 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.20.0:

Aaron Crane, Abhijit Menon-Sen, Abigail, Abir Viqar, Alan Haggai Alavi, Alan Hourihane, Alexander Voronov, Alexandr Ciornii, Andy Dougherty, Anno Siegel, Aristotle Pagaltzis, Arthur Axel 'fREW' Schmidt, Brad Gilbert, Brendan Byrd, Brian Childs, Brian Fraser, Brian Gottreu, Chris 'BinGOs' Williams, Christian Millour, Colin Kuskie, Craig A. Berry, Dabrien 'Dabe' Murphy, Dagfinn Ilmari Mannså, Daniel Dragan, Darin McBride, David Golden, David Leadbeater, David Mitchell, David Nicol, David Steinbrunner, Dennis Kaarsemaker, Dominic Hargreaves, Ed Avis, Eric Brine, Evan Zacks, Father Chrysostomos, Florian Ragwitz, François Perrad, Gavin Shelley, Gideon Israel Dsouza, Gisle Aas, Graham Knop, H.Merijn Brand, Hauke D, Heiko Eissfeldt, Hiroo Hayashi, Hojung Youn, James E Keenan, Jarkko Hietaniemi, Jerry D. Hedden, Jess Robinson, Jesse Luehrs, Johan Vromans, John Gardiner Myers, John Goodyear, John P. Linderman, John Peacock, kafka, Kang-min Liu, Karen Etheridge, Karl Williamson, Keedi Kim, Kent Fredric, kevin dawson, Kevin Falcone, Kevin Ryde, Leon Timmermans, Lukas Mai, Marc Simpson, Marcel Grünauer, Marco Peereboom, Marcus Holland-Moritz, Mark Jason Dominus, Martin McGrath, Matthew Horsfall, Max Maischein, Mike Doherty, Moritz Lenz, Nathan Glenn, Nathan Trapuzzano, Neil Bowers, Neil Williams, Nicholas Clark, Niels Thykier, Niko Tyni, Olivier Mengué, Owain G. Ainsworth, Paul Green, Paul Johnson, Peter John Acklam, Peter Martini, Peter Rabbitson, Petr PísaX, Philip Boulain, Philip Guenther, Piotr Roszatycki, Rafael Garcia-Suarez, Reini Urban, Reuben Thomas, Ricardo Signes, Ruslan Zakirov, Sergey Alekseev, Shirakata Kentaro, Shlomi Fish, Slaven Rezic, Smylers, Steffen Müller, Steve Hay, Sullivan Beck, Thomas Sibley, Tobias Leich, Toby Inkster, Tokuhiro Matsuno, Tom Christiansen, Tom Hukins, Tony Cook, Victor Efimov, Viktor Turskyi, Vladimir Timofeev, YAMASHINA Hio, Yves Orton, Zefram, Zsbán Ambrus, ÆEvar Arnfjörð Bjarmason.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5201delta − what is new for perl v5.20.1

## DESCRIPTION

This document describes differences between the 5.20.0 release and the 5.20.1 release.

If you are upgrading from an earlier release such as 5.18.0, first read perl5200delta, which describes differences between 5.18.0 and 5.20.0.

## Incompatible Changes

There are no changes intentionally incompatible with 5.20.0. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Performance Enhancements

- An optimization to avoid problems with COW and deliberately overallocated PVs has been disabled because it interfered with another, more important, optimization, causing a slowdown on some platforms. [GH #13878] <https://github.com/Perl/perl5/issues/13878>
- Returning a string from a lexical variable could be slow in some cases. This has now been fixed. [GH #13880] <https://github.com/Perl/perl5/issues/13880>

## Modules and Pragmata

### Updated Modules and Pragmata

- Config::Perl::V has been upgraded from version 0.20 to 0.22.

  The list of Perl versions covered has been updated and some flaws in the parsing have been fixed.

- Exporter has been upgraded from version 5.70 to 5.71.

  Illegal POD syntax in the documentation has been corrected.

- ExtUtils::CBuilder has been upgraded from version 0.280216 to 0.280217.

  Android builds now link to both **−lperl** and `$Config::Config{perllibs}`.

- File::Copy has been upgraded from version 2.29 to 2.30.

  The documentation now notes that `copy` will not overwrite read-only files.

- Module::CoreList has been upgraded from version 3.11 to 5.020001.

  The list of Perl versions covered has been updated.

- The PathTools module collection has been upgraded from version 3.47 to 3.48.

  Fallbacks are now in place when cross-compiling for Android and `$Config::Config{sh}` is not yet defined. [GH #13872] <https://github.com/Perl/perl5/issues/13872>

- PerlIO::via has been upgraded from version 0.14 to 0.15.

  A minor portability improvement has been made to the XS implementation.

- Unicode::UCD has been upgraded from version 0.57 to 0.58.

  The documentation includes many clarifications and fixes.

- utf8 has been upgraded from version 1.13 to 1.13_01.

  The documentation has some minor formatting improvements.

- version has been upgraded from version 0.9908 to 0.9909.

  External libraries and Perl may have different ideas of what the locale is. This is problematic when parsing version strings if the locale's numeric separator has been changed. Version parsing has been patched to ensure it handles the locales correctly. [GH #13863] <https://github.com/Perl/perl5/issues/13863>

## Documentation

### Changes to Existing Documentation

*perlapi*

- `av_len` − Emphasize that this returns the highest index in the array, not the size of the array. [GH #13377] <https://github.com/Perl/perl5/issues/13377>

- Note that `SvSetSV` doesn't do set magic.
- `sv_usepvn_flags` − Fix documentation to mention the use of `NewX` instead of `malloc`. [GH #13835] <https://github.com/Perl/perl5/issues/13835>
- Clarify where `NUL` may be embedded or is required to terminate a string.

*perlfunc*

- Clarify the meaning of `-B` and `-T`.
- `-l` now notes that it will return false if symlinks aren't supported by the file system. [GH #13695] <https://github.com/Perl/perl5/issues/13695>
- Note that `each`, `keys` and `values` may produce different orderings for tied hashes compared to other perl hashes. [GH #13650] <https://github.com/Perl/perl5/issues/13650>
- Note that `exec LIST` and `system LIST` may fall back to the shell on Win32. Only `exec PROGRAM LIST` and `system PROGRAM LIST` indirect object syntax will reliably avoid using the shell. This has also been noted in perlport. [GH #13907] <https://github.com/Perl/perl5/issues/13907>
- Clarify the meaning of `our`. [GH #13938] <https://github.com/Perl/perl5/issues/13938>

*perlguts*

- Explain various ways of modifying an existing SV's buffer. [GH #12813] <https://github.com/Perl/perl5/issues/12813>

*perlpolicy*

- We now have a code of conduct for the *p5p* mailing list, as documented in "STANDARDS OF CONDUCT" in perlpolicy.

*perlre*

- The `/x` modifier has been clarified to note that comments cannot be continued onto the next line by escaping them.

*perlsyn*

- Mention the use of empty conditionals in `for`/`while` loops for infinite loops.

*perlxs*

- Added a discussion of locale issues in XS code.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### Changes to Existing Diagnostics

- Variable length lookbehind not implemented in regex m/%s/

  Information about Unicode behaviour has been added.

## Configuration and Compilation

- Building Perl no longer writes to the source tree when configured with *Configure*'s **-Dmksymlinks** option. [GH #13712] <https://github.com/Perl/perl5/issues/13712>

## Platform Support

### Platform-Specific Notes

Android
: Build support has been improved for cross-compiling in general and for Android in particular.

OpenBSD
: Corrected architectures and version numbers used in configuration hints when building Perl.

Solaris
: **c99** options have been cleaned up, hints look for **solstudio** as well as **SUNWspro**, and support for native `setenv` has been added.

VMS
: An old bug in feature checking, mainly affecting pre−7.3 systems, has been fixed.

Windows
: `%I64d` is now being used instead of `%lld` for MinGW.

## Internal Changes

- Added ‘‘sync_locale’’ in perlapi. Changing the program’s locale should be avoided by XS code. Nevertheless, certain non-Perl libraries called from XS, such as `Gtk` do so. When this happens, Perl needs to be told that the locale has changed. Use this function to do so, before returning to Perl.

## Selected Bug Fixes

- A bug has been fixed where zero-length assertions and code blocks inside of a regex could cause `pos` to see an incorrect value. [GH #14016] <https://github.com/Perl/perl5/issues/14016>
- Using `s///e` on tainted utf8 strings could issue bogus ‘‘Malformed UTF−8 character (unexpected end of string)’’ warnings. This has now been fixed. [GH #13948] <https://github.com/Perl/perl5/issues/13948>
- `system` and friends should now work properly on more Android builds.

  Due to an oversight, the value specified through **−Dtargetsh** to *Configure* would end up being ignored by some of the build process. This caused perls cross-compiled for Android to end up with defective versions of `system`, `exec` and backticks: the commands would end up looking for */bin/sh* instead of */system/bin/sh*, and so would fail for the vast majority of devices, leaving `$!` as `ENOENT`.
- Many issues have been detected by Coverity <http://www.coverity.com/> and fixed.

## Acknowledgements

Perl 5.20.1 represents approximately 4 months of development since Perl 5.20.0 and contains approximately 12,000 lines of changes across 170 files from 36 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 2,600 lines of changes to 110 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.20.1:

Aaron Crane, Abigail, Alberto Simões, Alexandr Ciornii, Alexandre (Midnite) Jousset, Andrew Fresh, Andy Dougherty, Brian Fraser, Chris ’BinGOs’ Williams, Craig A. Berry, Daniel Dragan, David Golden, David Mitchell, H.Merijn Brand, James E Keenan, Jan Dubois, Jarkko Hietaniemi, John Peacock, kafka, Karen Etheridge, Karl Williamson, Lukas Mai, Matthew Horsfall, Michael Bunk, Peter Martini, Rafael Garcia-Suarez, Reini Urban, Ricardo Signes, Shirakata Kentaro, Smylers, Steve Hay, Thomas Sibley, Todd Rinaldo, Tony Cook, Vladimir Marek, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help

assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5202delta − what is new for perl v5.20.2

## DESCRIPTION

This document describes differences between the 5.20.1 release and the 5.20.2 release.

If you are upgrading from an earlier release such as 5.20.0, first read perl5201delta, which describes differences between 5.20.0 and 5.20.1.

## Incompatible Changes

There are no changes intentionally incompatible with 5.20.1. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- attributes has been upgraded from version 0.22 to 0.23.

  The usage of `memEQs` in the XS has been corrected. [GH #14072] <https://github.com/Perl/perl5/issues/14072>

- Data::Dumper has been upgraded from version 2.151 to 2.151_01.

  Fixes CVE−2014−4330 by adding a configuration variable/option to limit recursion when dumping deep data structures.

- Errno has been upgraded from version 1.20_03 to 1.20_05.

  Warnings when building the XS on Windows with the Visual C++ compiler are now avoided.

- feature has been upgraded from version 1.36 to 1.36_01.

  The `postderef` feature has now been documented. This feature was actually added in Perl 5.20.0 but was accidentally omitted from the feature documentation until now.

- IO::Socket has been upgraded from version 1.37 to 1.38.

  Document the limitations of the **connected()** method. [GH #14199] <https://github.com/Perl/perl5/issues/14199>

- Module::CoreList has been upgraded from version 5.020001 to 5.20150214.

  The list of Perl versions covered has been updated.

- PathTools has been upgraded from version 3.48 to 3.48_01.

  A warning from the **gcc** compiler is now avoided when building the XS.

- PerlIO::scalar has been upgraded from version 0.18 to 0.18_01.

  Reading from a position well past the end of the scalar now correctly returns end of file. [GH #14342] <https://github.com/Perl/perl5/issues/14342>

  Seeking to a negative position still fails, but no longer leaves the file position set to a negation location.

  `eof()` on a `PerlIO::scalar` handle now properly returns true when the file position is past the 2GB mark on 32−bit systems.

- Storable has been upgraded from version 2.49 to 2.49_01.

  Minor grammatical change to the documentation only.

- VMS::DCLsym has been upgraded from version 1.05 to 1.05_01.

  Minor formatting change to the documentation only.

- VMS::Stdio has been upgraded from version 2.4 to 2.41.

  Minor formatting change to the documentation only.

## Documentation

### New Documentation

*perlunicook*

This document, by Tom Christiansen, provides examples of handling Unicode in Perl.

### Changes to Existing Documentation

*perlexperiment*

- Added reference to subroutine signatures. This feature was actually added in Perl 5.20.0 but was accidentally omitted from the experimental feature documentation until now.

*perlpolicy*

- The process whereby features may graduate from experimental status has now been formally documented.

*perlsyn*

- An ambiguity in the documentation of the ellipsis statement has been corrected. [GH #14054] <https://github.com/Perl/perl5/issues/14054>

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### Changes to Existing Diagnostics

- Bad symbol for scalar is now documented. This error is not new, but was not previously documented here.
- Missing right brace on \N{} is now documented. This error is not new, but was not previously documented here.

## Testing

- The test script *re/rt122747.t* has been added to verify that [GH #14081] <https://github.com/Perl/perl5/issues/14081> remains fixed.

## Platform Support

### Regained Platforms

IRIX and Tru64 platforms are working again. (Some `make test` failures remain.)

## Selected Bug Fixes

- AIX now sets the length in `getsockopt` correctly. [GH #13484] <https://github.com/Perl/perl5/issues/13484>, [cpan #91183] <https://rt.cpan.org/Ticket/Display.html?id=91183>, [cpan #85570] <https://rt.cpan.org/Ticket/Display.html?id=85570>
- In Perl 5.20.0, `$^N` accidentally had the internal UTF8 flag turned off if accessed from a code block within a regular expression, effectively UTF8-encoding the value. This has been fixed. [GH #14211] <https://github.com/Perl/perl5/issues/14211>
- Various cases where the name of a sub is used (autoload, overloading, error messages) used to crash for lexical subs, but have been fixed.
- An assertion failure when parsing `sort` with debugging enabled has been fixed. [GH #14087] <https://github.com/Perl/perl5/issues/14087>
- Loading UTF8 tables during a regular expression match could cause assertion failures under debugging builds if the previous match used the very same regular expression. [GH #14081] <https://github.com/Perl/perl5/issues/14081>
- Due to a mistake in the string-copying logic, copying the value of a state variable could instead steal the value and undefine the variable. This bug, introduced in Perl 5.20, would happen mostly for long strings (1250 chars or more), but could happen for any strings under builds with copy-on-write disabled. [GH #14175] <https://github.com/Perl/perl5/issues/14175>
- Fixed a bug that could cause perl to execute an infinite loop during compilation. [GH #14165] <https://github.com/Perl/perl5/issues/14165>
- On Win32, restoring in a child pseudo-process a variable that was `local()`ed in a parent pseudo-process before the `fork` happened caused memory corruption and a crash in the child pseudo-process (and therefore OS process). [GH #8641] <https://github.com/Perl/perl5/issues/8641>

- Tainted constants evaluated at compile time no longer cause unrelated statements to become tainted. [GH #14059] <https://github.com/Perl/perl5/issues/14059>
- Calling `write` on a format with a `^**` field could produce a panic in **sv_chop()** if there were insufficient arguments or if the variable used to fill the field was empty. [GH #14255] <https://github.com/Perl/perl5/issues/14255>
- In Perl 5.20.0, `sort CORE::fake` where 'fake' is anything other than a keyword started chopping of the last 6 characters and treating the result as a sort sub name. The previous behaviour of treating "CORE::fake" as a sort sub name has been restored. [GH #14323] <https://github.com/Perl/perl5/issues/14323>
- A bug in regular expression patterns that could lead to segfaults and other crashes has been fixed. This occurred only in patterns compiled with `"/i"`, while taking into account the current POSIX locale (this usually means they have to be compiled within the scope of `"use locale"`), and there must be a string of at least 128 consecutive bytes to match. [GH #14389] <https://github.com/Perl/perl5/issues/14389>
- `qr/@array(?{block})/` no longer dies with "Bizarre copy of ARRAY". [GH #14292] <https://github.com/Perl/perl5/issues/14292>
- `gmtime` no longer crashes with not-a-number values. [GH #14365] <https://github.com/Perl/perl5/issues/14365>
- Certain syntax errors in substitutions, such as `s/${<>{})//`, would crash, and had done so since Perl 5.10. (In some cases the crash did not start happening until Perl 5.16.) The crash has, of course, been fixed. [GH #14391] <https://github.com/Perl/perl5/issues/14391>
- A memory leak in some regular expressions, introduced in Perl 5.20.1, has been fixed. [GH #14236] <https://github.com/Perl/perl5/issues/14236>
- `formline("@...", "a");` would crash. The `FF_CHECKNL` case in **pp_formline()** didn't set the pointer used to mark the chop position, which led to the `FF_MORE` case crashing with a segmentation fault. This has been fixed. [GH #14388] <https://github.com/Perl/perl5/issues/14388> [GH #14425] <https://github.com/Perl/perl5/issues/14425>
- A possible buffer overrun and crash when parsing a literal pattern during regular expression compilation has been fixed. [GH #14416] <https://github.com/Perl/perl5/issues/14416>

## Known Problems

- It is a known bug that lexical subroutines cannot be used as the `SUBNAME` argument to `sort`. This will be fixed in a future version of Perl.

## Errata From Previous Releases

- A regression has been fixed that was introduced in Perl 5.20.0 (fixed in Perl 5.20.1 as well as here) in which a UTF-8 encoded regular expression pattern that contains a single ASCII lowercase letter does not match its uppercase counterpart. [GH #14051] <https://github.com/Perl/perl5/issues/14051>

## Acknowledgements

Perl 5.20.2 represents approximately 5 months of development since Perl 5.20.1 and contains approximately 6,300 lines of changes across 170 files from 34 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,900 lines of changes to 80 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.20.2:

Aaron Crane, Abigail, Andreas Voegele, Andy Dougherty, Anthony Heading, Aristotle Pagaltzis, Chris 'BinGOs' Williams, Craig A. Berry, Daniel Dragan, Doug Bell, Ed J, Father Chrysostomos, Glenn D. Golden, H.Merijn Brand, Hugo van der Sanden, James E Keenan, Jarkko Hietaniemi, Jim Cromie, Karen Etheridge, Karl Williamson, kmx, Matthew Horsfall, Max Maischein, Peter Martini, Rafael Garcia-Suarez, Ricardo Signes, Shlomi Fish, Slaven Rezic, Steffen Müller, Steve Hay, Tadeusz SoXnierz, Tony Cook, Yves Orton, Ævar Arnfjörð Bjarmason.

The list above is almost certainly incomplete as it is automatically generated from version control

history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5203delta − what is new for perl v5.20.3

## DESCRIPTION

This document describes differences between the 5.20.2 release and the 5.20.3 release.

If you are upgrading from an earlier release such as 5.20.1, first read perl5202delta, which describes differences between 5.20.1 and 5.20.2.

## Incompatible Changes

There are no changes intentionally incompatible with 5.20.2. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Errno has been upgraded from version 1.20_05 to 1.20_06.

  Add **−P** to the pre-processor command-line on GCC 5. GCC added extra line directives, breaking parsing of error code definitions. [GH #14491] <https://github.com/Perl/perl5/issues/14491>

- Module::CoreList has been upgraded from version 5.20150214 to 5.20150822.

  Updated to cover the latest releases of Perl.

- perl5db.pl has been upgraded from 1.44 to 1.44_01.

  The debugger would cause an assertion failure. [GH #14605] <https://github.com/Perl/perl5/issues/14605>

## Documentation

### Changes to Existing Documentation

*perlfunc*

- Mention that `study()` is currently a no-op.

*perlguts*

- The OOK example has been updated to account for COW changes and a change in the storage of the offset.

*perlhacktips*

- Documentation has been added illustrating the perils of assuming the contents of static memory pointed to by the return values of Perl wrappers for C library functions doesn't change.

*perlpodspec*

- The specification of the POD language is changing so that the default encoding of PODs that aren't in UTF−8 (unless otherwise indicated) is CP1252 instead of ISO−8859−1 (Latin1).

## Utility Changes

### h2ph

- **h2ph** now handles hexadecimal constants in the compiler's predefined macro definitions, as visible in `$Config{cppsymbols}`. [GH #14491] <https://github.com/Perl/perl5/issues/14491>

## Testing

- *t/perf/taint.t* has been added to see if optimisations with taint issues are keeping things fast.
- *t/porting/re_context.t* has been added to test that utf8 and its dependencies only use the subset of the `$1..$n` capture vars that **Perl_save_re_context()** is hard-coded to localize, because that function has no efficient way of determining at runtime what vars to localize.

## Platform Support

### Platform-Specific Notes

Win32

- Previously, when compiling with a 64−bit Visual C++, every Perl XS module (including CPAN ones) and Perl aware C file would unconditionally have around a dozen warnings from *hv_func.h*. These warnings have been silenced. GCC (all bitness) and 32−bit Visual C++ were not affected.

- **miniperl.exe** is now built with **-fno-strict-aliasing**, allowing 64-bit builds to complete with GCC 4.8. [GH #14556] <https://github.com/Perl/perl5/issues/14556>

## Selected Bug Fixes

- Repeated global pattern matches in scalar context on large tainted strings were exponentially slow depending on the current match position in the string. [GH #14238] <https://github.com/Perl/perl5/issues/14238>
- The original visible value of `$/` is now preserved when it is set to an invalid value. Previously if you set `$/` to a reference to an array, for example, perl would produce a runtime error and not set PL_rs, but Perl code that checked `$/` would see the array reference. [GH #14245] <https://github.com/Perl/perl5/issues/14245>
- Perl 5.14.0 introduced a bug whereby `eval { LABEL: }` would crash. This has been fixed. [GH #14438] <https://github.com/Perl/perl5/issues/14438>
- Extending an array cloned from a parent thread could result in ''Modification of a read-only value attempted'' errors when attempting to modify the new elements. [GH #14605] <https://github.com/Perl/perl5/issues/14605>
- Several cases of data used to store environment variable contents in core C code being potentially overwritten before being used have been fixed. [GH #14476] <https://github.com/Perl/perl5/issues/14476>
- UTF-8 variable names used in array indexes, unquoted UTF-8 HERE-document terminators and UTF-8 function names all now work correctly. [GH #14601] <https://github.com/Perl/perl5/issues/14601>
- A subtle bug introduced in Perl 5.20.2 involving UTF-8 in regular expressions and sometimes causing a crash has been fixed. A new test script has been added to test this fix; see under ''Testing''. [GH #14600] <https://github.com/Perl/perl5/issues/14600>
- Some patterns starting with `/.*..../` matched against long strings have been slow since Perl 5.8, and some of the form `/.*..../i` have been slow since Perl 5.18. They are now all fast again. [GH #14475] <https://github.com/Perl/perl5/issues/14475>
- Warning fatality is now ignored when rewinding the stack. This prevents infinite recursion when the now fatal error also causes rewinding of the stack. [GH #14319] <https://github.com/Perl/perl5/issues/14319>
- `setpgrp($nonzero)` (with one argument) was accidentally changed in Perl 5.16 to mean `setpgrp(0)`. This has been fixed.
- A crash with `%::=(); J->${\"::"}` has been fixed. [GH #14790] <https://github.com/Perl/perl5/issues/14790>
- Regular expression possessive quantifier Perl 5.20 regression now fixed. `qr/`*PAT*`{`*min*`,`*max*`}+/` is supposed to behave identically to `qr/(?>`*PAT*`{`*min*`,`*max*`})/`. Since Perl 5.20, this didn't work if *min* and *max* were equal. [GH #14857] <https://github.com/Perl/perl5/issues/14857>
- Code like `/$a[/` used to read the next line of input and treat it as though it came immediately after the opening bracket. Some invalid code consequently would parse and run, but some code caused crashes, so this is now disallowed. [GH #14462] <https://github.com/Perl/perl5/issues/14462>

## Acknowledgements

Perl 5.20.3 represents approximately 7 months of development since Perl 5.20.2 and contains approximately 3,200 lines of changes across 99 files from 26 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,500 lines of changes to 43 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.20.3:

Alex Vandiver, Andy Dougherty, Aristotle Pagaltzis, Chris 'BinGOs' Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, Daniel Dragan, David Mitchell, Father Chrysostomos, H.Merijn Brand, James E Keenan, James McCoy, Jarkko Hietaniemi, Karen Etheridge, Karl Williamson, kmx, Lajos Veres, Lukas Mai, Matthew Horsfall, Petr PísaX, Randy Stauner, Ricardo Signes, Sawyer X, Steve Hay,

Tony Cook, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5220delta − what is new for perl v5.22.0

## DESCRIPTION

This document describes differences between the 5.20.0 release and the 5.22.0 release.

If you are upgrading from an earlier release such as 5.18.0, first read perl5200delta, which describes differences between 5.18.0 and 5.20.0.

## Core Enhancements

### New bitwise operators

A new experimental facility has been added that makes the four standard bitwise operators (`& | ^ ~`) treat their operands consistently as numbers, and introduces four new dotted operators (`&. |. ^. ~.`) that treat their operands consistently as strings. The same applies to the assignment variants (`&= |= ^= &.= |.= ^.=`).

To use this, enable the ''bitwise'' feature and disable the ''experimental::bitwise'' warnings category. See ''Bitwise String Operators'' in perlop for details. [GH #14348] <https://github.com/Perl/perl5/issues/14348>.

### New double-diamond operator

`<<>>` is like `<>` but uses three-argument `open` to open each file in `@ARGV`. This means that each element of `@ARGV` will be treated as an actual file name, and `"|foo"` won't be treated as a pipe open.

### New `\b` boundaries in regular expressions

*`qr/\b{gcb}/`*

`gcb` stands for Grapheme Cluster Boundary. It is a Unicode property that finds the boundary between sequences of characters that look like a single character to a native speaker of a language. Perl has long had the ability to deal with these through the `\X` regular escape sequence. Now, there is an alternative way of handling these. See ''\b{}, \b, \B{}, \B'' in perlrebackslash for details.

*`qr/\b{wb}/`*

`wb` stands for Word Boundary. It is a Unicode property that finds the boundary between words. This is similar to the plain `\b` (without braces) but is more suitable for natural language processing. It knows, for example, that apostrophes can occur in the middle of words. See ''\b{}, \b, \B{}, \B'' in perlrebackslash for details.

*`qr/\b{sb}/`*

`sb` stands for Sentence Boundary. It is a Unicode property to aid in parsing natural language sentences. See ''\b{}, \b, \B{}, \B'' in perlrebackslash for details.

### Non-Capturing Regular Expression Flag

Regular expressions now support a `/n` flag that disables capturing and filling in `$1`, `$2`, etc inside of groups:

```
"hello" =~ /(hi|hello)/n; # $1 is not set
```

This is equivalent to putting `?:` at the beginning of every capturing group.

See ''n'' in perlre for more information.

### `use re 'strict'`

This applies stricter syntax rules to regular expression patterns compiled within its scope. This will hopefully alert you to typos and other unintentional behavior that backwards-compatibility issues prevent us from reporting in normal regular expression compilations. Because the behavior of this is subject to change in future Perl releases as we gain experience, using this pragma will raise a warning of category `experimental::re_strict`. See 'strict' in re.

### Unicode 7.0 (with correction) is now supported

For details on what is in this release, see <http://www.unicode.org/versions/Unicode7.0.0/>. The version of Unicode 7.0 that comes with Perl includes a correction dealing with glyph shaping in Arabic (see <http://www.unicode.org/errata/#current_errata>).

### `use locale` can restrict which locale categories are affected

It is now possible to pass a parameter to `use locale` to specify a subset of locale categories to be locale-aware, with the remaining ones unaffected. See ''The ''use locale'' pragma'' in perllocale for

details.

### Perl now supports POSIX 2008 locale currency additions

On platforms that are able to handle POSIX.1−2008, the hash returned by `POSIX::localeconv()` includes the international currency fields added by that version of the POSIX standard. These are `int_n_cs_precedes`, `int_n_sep_by_space`, `int_n_sign_posn`, `int_p_cs_precedes`, `int_p_sep_by_space`, and `int_p_sign_posn`.

### Better heuristics on older platforms for determining locale UTF−8ness

On platforms that implement neither the C99 standard nor the POSIX 2001 standard, determining if the current locale is UTF−8 or not depends on heuristics. These are improved in this release.

### Aliasing via reference

Variables and subroutines can now be aliased by assigning to a reference:

```
\$c = \$d;
\&x = \&y;
```

Aliasing can also be accomplished by using a backslash before a `foreach` iterator variable; this is perhaps the most useful idiom this feature provides:

```
foreach \%hash (@array_of_hash_refs) { ... }
```

This feature is experimental and must be enabled via `use feature 'refaliasing'`. It will warn unless the `experimental::refaliasing` warnings category is disabled.

See ‘‘Assigning to References’’ in perlref

### `prototype` with no arguments

`prototype()` with no arguments now infers `$_`. [GH #14376] <https://github.com/Perl/perl5/issues/14376>.

### New `:const` subroutine attribute

The `const` attribute can be applied to an anonymous subroutine. It causes the new sub to be executed immediately whenever one is created (*i.e.* when the `sub` expression is evaluated). Its value is captured and used to create a new constant subroutine that is returned. This feature is experimental. See ‘‘Constant Functions’’ in perlsub.

### `fileno` now works on directory handles

When the relevant support is available in the operating system, the `fileno` builtin now works on directory handles, yielding the underlying file descriptor in the same way as for filehandles. On operating systems without such support, `fileno` on a directory handle continues to return the undefined value, as before, but also sets `$!` to indicate that the operation is not supported.

Currently, this uses either a `dd_fd` member in the OS `DIR` structure, or a `dirfd(3)` function as specified by POSIX.1−2008.

### List form of pipe open implemented for Win32

The list form of pipe:

```
  open my $fh, "−|", "program", @arguments;
```

is now implemented on Win32. It has the same limitations as `system LIST` on Win32, since the Win32 API doesn’t accept program arguments as a list.

### Assignment to list repetition

`(...) x ...` can now be used within a list that is assigned to, as long as the left-hand side is a valid lvalue. This allows `(undef,undef,$foo) = that_function()` to be written as `((undef)x2, $foo) = that_function()`.

### Infinity and NaN (not-a-number) handling improved

Floating point values are able to hold the special values infinity, negative infinity, and NaN (not-a-number). Now we more robustly recognize and propagate the value in computations, and on output normalize them to the strings `Inf`, `−Inf`, and `NaN`.

See also the POSIX enhancements.

### Floating point parsing has been improved

Parsing and printing of floating point values has been improved.

As a completely new feature, hexadecimal floating point literals (like `0x1.23p−4`) are now

supported, and they can be output with `printf "%a"`. See ‘‘Scalar value constructors’’ in perldata for more details.

### Packing infinity or not-a-number into a character is now fatal

Before, when trying to pack infinity or not-a-number into a (signed) character, Perl would warn, and assumed you tried to pack `0xFF`; if you gave it as an argument to `chr`, `U+FFFD` was returned.

But now, all such actions (`pack`, `chr`, and `print '%c'`) result in a fatal error.

### Experimental C Backtrace API

Perl now supports (via a C level API) retrieving the C level backtrace (similar to what symbolic debuggers like gdb do).

The backtrace returns the stack trace of the C call frames, with the symbol names (function names), the object names (like ‘‘perl’’), and if it can, also the source code locations (file:line).

The supported platforms are Linux and OS X (some *BSD might work at least partly, but they have not yet been tested).

The feature needs to be enabled with `Configure -Dusecbacktrace`.

See ‘‘C backtrace’’ in perlhacktips for more information.

## Security

### Perl is now compiled with `-fstack-protector-strong` if available

Perl has been compiled with the anti-stack-smashing option `-fstack-protector` since 5.10.1. Now Perl uses the newer variant called `-fstack-protector-strong`, if available.

### The Safe module could allow outside packages to be replaced

Critical bugfix: outside packages could be replaced. Safe has been patched to 2.38 to address this.

### Perl is now always compiled with `-D_FORTIFY_SOURCE=2` if available

The ’code hardening’ option called `_FORTIFY_SOURCE`, available in gcc 4.*, is now always used for compiling Perl, if available.

Note that this isn’t necessarily a huge step since in many platforms the step had already been taken several years ago: many Linux distributions (like Fedora) have been using this option for Perl, and OS X has enforced the same for many years.

## Incompatible Changes

### Subroutine signatures moved before attributes

The experimental sub signatures feature, as introduced in 5.20, parsed signatures after attributes. In this release, following feedback from users of the experimental feature, the positioning has been moved such that signatures occur after the subroutine name (if any) and before the attribute list (if any).

### `&` and `\&` prototypes accepts only subs

The `&` prototype character now accepts only anonymous subs (`sub {...}`), things beginning with `\&`, or an explicit `undef`. Formerly it erroneously also allowed references to arrays, hashes, and lists. [GH #2776] <https://github.com/Perl/perl5/issues/2776>. [GH #14186] <https://github.com/Perl/perl5/issues/14186>. [GH #14353] <https://github.com/Perl/perl5/issues/14353>.

In addition, the `\&` prototype was allowing subroutine calls, whereas now it only allows subroutines: `&foo` is still permitted as an argument, while `&foo()` and `foo()` no longer are. [GH #10633] <https://github.com/Perl/perl5/issues/10633>.

### `use encoding` is now lexical

The encoding pragma’s effect is now limited to lexical scope. This pragma is deprecated, but in the meantime, it could adversely affect unrelated modules that are included in the same program; this change fixes that.

### List slices returning empty lists

List slices now return an empty list only if the original list was empty (or if there are no indices). Formerly, a list slice would return an empty list if all indices fell outside the original list; now it returns a list of `undef` values in that case. [GH #12335] <https://github.com/Perl/perl5/issues/12335>.

### `\N{}` with a sequence of multiple spaces is now a fatal error

E.g. `\N{TOO  MANY SPACES}` or `\N{TRAILING SPACE }`. This has been deprecated since v5.18.

**`use UNIVERSAL '...'` is now a fatal error**

Importing functions from `UNIVERSAL` has been deprecated since v5.12, and is now a fatal error. `use UNIVERSAL` without any arguments is still allowed.

**In double-quotish `\c`*X*, *X* must now be a printable ASCII character**

In prior releases, failure to do this raised a deprecation warning.

**Splitting the tokens `(?` and `(*` in regular expressions is now a fatal compilation error.**

These had been deprecated since v5.18.

**`qr/foo/x` now ignores all Unicode pattern white space**

The `/x` regular expression modifier allows the pattern to contain white space and comments (both of which are ignored) for improved readability. Until now, not all the white space characters that Unicode designates for this purpose were handled. The additional ones now recognized are:

```
U+0085 NEXT LINE
U+200E LEFT-TO-RIGHT MARK
U+200F RIGHT-TO-LEFT MARK
U+2028 LINE SEPARATOR
U+2029 PARAGRAPH SEPARATOR
```

The use of these characters with `/x` outside bracketed character classes and when not preceded by a backslash has raised a deprecation warning since v5.18. Now they will be ignored.

**Comment lines within `(?[ ])` are now ended only by a `\n`**

`(?[ ])` is an experimental feature, introduced in v5.18. It operates as if `/x` is always enabled. But there was a difference: comment lines (following a # character) were terminated by anything matching `\R` which includes all vertical whitespace, such as form feeds. For consistency, this is now changed to match what terminates comment lines outside `(?[ ])`, namely a `\n` (even if escaped), which is the same as what terminates a heredoc string and formats.

**`(?[...])` operators now follow standard Perl precedence**

This experimental feature allows set operations in regular expression patterns. Prior to this, the intersection operator had the same precedence as the other binary operators. Now it has higher precedence. This could lead to different outcomes than existing code expects (though the documentation has always noted that this change might happen, recommending fully parenthesizing the expressions). See ''Extended Bracketed Character Classes'' in perlrecharclass.

**Omitting `%` and `@` on hash and array names is no longer permitted**

Really old Perl let you omit the `@` on array names and the `%` on hash names in some spots. This has issued a deprecation warning since Perl 5.000, and is no longer permitted.

**''`$!`'' text is now in English outside the scope of `use locale`**

Previously, the text, unlike almost everything else, always came out based on the current underlying locale of the program. (Also affected on some systems is `"$^E"`.) For programs that are unprepared to handle locale differences, this can cause garbage text to be displayed. It's better to display text that is translatable via some tool than garbage text which is much harder to figure out.

**''`$!`'' text will be returned in UTF-8 when appropriate**

The stringification of `$!` and `$^E` will have the UTF-8 flag set when the text is actually non-ASCII UTF-8. This will enable programs that are set up to be locale-aware to properly output messages in the user's native language. Code that needs to continue the 5.20 and earlier behavior can do the stringification within the scopes of both `use bytes` and `use locale ":messages"`. Within these two scopes, no other Perl operations will be affected by locale; only `$!` and `$^E` stringification. The `bytes` pragma causes the UTF-8 flag to not be set, just as in previous Perl releases. This resolves [GH #12035] <https://github.com/Perl/perl5/issues/12035>.

**Support for `?PATTERN?` without explicit operator has been removed**

The `m?PATTERN?` construct, which allows matching a regex only once, previously had an alternative form that was written directly with a question mark delimiter, omitting the explicit `m` operator. This usage has produced a deprecation warning since 5.14.0. It is now a syntax error, so that the question mark can be available for use in new operators.

**`defined(@array)` and `defined(%hash)` are now fatal errors**

These have been deprecated since v5.6.1 and have raised deprecation warnings since v5.16.

### Using a hash or an array as a reference are now fatal errors

For example, `%foo->{"bar"}` now causes a fatal compilation error. These have been deprecated since before v5.8, and have raised deprecation warnings since then.

### Changes to the `*` prototype

The `*` character in a subroutine's prototype used to allow barewords to take precedence over most, but not all, subroutine names. It was never consistent and exhibited buggy behavior.

Now it has been changed, so subroutines always take precedence over barewords, which brings it into conformity with similarly prototyped built-in functions:

```
sub splat(*) { ... }
sub foo { ... }
splat(foo); # now always splat(foo())
splat(bar); # still splat('bar') as before
close(foo); # close(foo())
close(bar); # close('bar')
```

## Deprecations

### Setting `${^ENCODING}` to anything but `undef`

This variable allows Perl scripts to be written in an encoding other than ASCII or UTF-8. However, it affects all modules globally, leading to wrong answers and segmentation faults. New scripts should be written in UTF-8; old scripts should be converted to UTF-8, which is easily done with the piconv utility.

### Use of non-graphic characters in single-character variable names

The syntax for single-character variable names is more lenient than for longer variable names, allowing the one-character name to be a punctuation character or even invisible (a non-graphic). Perl v5.20 deprecated the ASCII-range controls as such a name. Now, all non-graphic characters that formerly were allowed are deprecated. The practical effect of this occurs only when not under `use utf8`, and affects just the C1 controls (code points 0x80 through 0xFF), NO-BREAK SPACE, and SOFT HYPHEN.

### Inlining of `sub () { $var }` with observable side-effects

In many cases Perl makes `sub () { $var }` into an inlinable constant subroutine, capturing the value of `$var` at the time the `sub` expression is evaluated. This can break the closure behavior in those cases where `$var` is subsequently modified, since the subroutine won't return the changed value. (Note that this all only applies to anonymous subroutines with an empty prototype (`sub ()`).)

This usage is now deprecated in those cases where the variable could be modified elsewhere. Perl detects those cases and emits a deprecation warning. Such code will likely change in the future and stop producing a constant.

If your variable is only modified in the place where it is declared, then Perl will continue to make the sub inlinable with no warnings.

```
sub make_constant {
    my $var = shift;
    return sub () { $var }; # fine
}

sub make_constant_deprecated {
    my $var;
    $var = shift;
    return sub () { $var }; # deprecated
}

sub make_constant_deprecated2 {
    my $var = shift;
    log_that_value($var); # could modify $var
    return sub () { $var }; # deprecated
}
```

In the second example above, detecting that `$var` is assigned to only once is too hard to detect. That it happens in a spot other than the `my` declaration is enough for Perl to find it suspicious.

This deprecation warning happens only for a simple variable for the body of the sub. (A `BEGIN` block

or `use` statement inside the sub is ignored, because it does not become part of the sub's body.) For more complex cases, such as `sub () { do_something() if 0; $var }` the behavior has changed such that inlining does not happen if the variable is modifiable elsewhere. Such cases should be rare.

#### Use of multiple `/x` regexp modifiers

It is now deprecated to say something like any of the following:

```
qr/foo/xx;
/(?xax:foo)/;
use re qw(/amxx);
```

That is, now `x` should only occur once in any string of contiguous regular expression pattern modifiers. We do not believe there are any occurrences of this in all of CPAN. This is in preparation for a future Perl release having `/xx` permit white-space for readability in bracketed character classes (those enclosed in square brackets: `[...]`).

#### Using a NO-BREAK space in a character alias for `\N{...}` is now deprecated

This non-graphic character is essentially indistinguishable from a regular space, and so should not be allowed. See "CUSTOM ALIASES" in charnames.

#### A literal `''{''` should now be escaped in a pattern

If you want a literal left curly bracket (also called a left brace) in a regular expression pattern, you should now escape it by either preceding it with a backslash (`"\{"`) or enclosing it within square brackets `"[{]"`, or by using `\Q`; otherwise a deprecation warning will be raised. This was first announced as forthcoming in the v5.16 release; it will allow future extensions to the language to happen.

#### Making all warnings fatal is discouraged

The documentation for fatal warnings notes that `use warnings FATAL => 'all'` is discouraged, and provides stronger language about the risks of fatal warnings in general.

## Performance Enhancements

- If a method or class name is known at compile time, a hash is precomputed to speed up run-time method lookup. Also, compound method names like `SUPER::new` are parsed at compile time, to save having to parse them at run time.
- Array and hash lookups (especially nested ones) that use only constants or simple variables as keys, are now considerably faster. See "Internal Changes" for more details.
- `(...)x1`, `("constant")x0` and `($scalar)x0` are now optimised in list context. If the right-hand argument is a constant 1, the repetition operator disappears. If the right-hand argument is a constant 0, the whole expression is optimised to the empty list, so long as the left-hand argument is a simple scalar or constant. (That is, `(foo())x0` is not subject to this optimisation.)
- `substr` assignment is now optimised into 4−argument `substr` at the end of a subroutine (or as the argument to `return`). Previously, this optimisation only happened in void context.
- In `"\L..."`, `"\Q..."`, etc., the extra "stringify" op is now optimised away, making these just as fast as `lcfirst`, `quotemeta`, etc.
- Assignment to an empty list is now sometimes faster. In particular, it never calls `FETCH` on tied arguments on the right-hand side, whereas it used to sometimes.
- There is a performance improvement of up to 20% when `length` is applied to a non-magical, non-tied string, and either `use bytes` is in scope or the string doesn't use UTF−8 internally.
- On most perl builds with 64−bit integers, memory usage for non-magical, non-tied scalars containing only a floating point value has been reduced by between 8 and 32 bytes, depending on OS.
- In `@array = split`, the assignment can be optimized away, so that `split` writes directly to the array. This optimisation was happening only for package arrays other than `@_`, and only sometimes. Now this optimisation happens almost all the time.
- `join` is now subject to constant folding. So for example `join "−", "a", "b"` is converted at compile-time to `"a−b"`. Moreover, `join` with a scalar or constant for the separator and a single-item list to join is simplified to a stringification, and the separator doesn't even get

evaluated.

- `qq(@array)` is implemented using two ops: a stringify op and a join op. If the `qq` contains nothing but a single array, the stringification is optimized away.
- `our $var` and `our($s,@a,%h)` in void context are no longer evaluated at run time. Even a whole sequence of `our $foo;` statements will simply be skipped over. The same applies to `state` variables.
- Many internal functions have been refactored to improve performance and reduce their memory footprints. [GH #13659] <https://github.com/Perl/perl5/issues/13659> [GH #13856] <https://github.com/Perl/perl5/issues/13856> [GH #13874] <https://github.com/Perl/perl5/issues/13874>
- `-T` and `-B` filetests will return sooner when an empty file is detected. [GH #13686] <https://github.com/Perl/perl5/issues/13686>
- Hash lookups where the key is a constant are faster.
- Subroutines with an empty prototype and a body containing just `undef` are now eligible for inlining. [GH #14077] <https://github.com/Perl/perl5/issues/14077>
- Subroutines in packages no longer need to be stored in typeglobs: declaring a subroutine will now put a simple sub reference directly in the stash if possible, saving memory. The typeglob still notionally exists, so accessing it will cause the stash entry to be upgraded to a typeglob (*i.e.* this is just an internal implementation detail). This optimization does not currently apply to XSUBs or exported subroutines, and method calls will undo it, since they cache things in typeglobs. [GH #13392] <https://github.com/Perl/perl5/issues/13392>
- The functions `utf8::native_to_unicode()` and `utf8::unicode_to_native()` (see utf8) are now optimized out on ASCII platforms. There is now not even a minimal performance hit in writing code portable between ASCII and EBCDIC platforms.
- Win32 Perl uses 8 KB less of per-process memory than before for every perl process, because some data is now memory mapped from disk and shared between processes from the same perl binary.

## Modules and Pragmata

### Updated Modules and Pragmata

Many of the libraries distributed with perl have been upgraded since v5.20.0. For a complete list of changes, run:

```
corelist --diff 5.20.0 5.22.0
```

You can substitute your favorite version in place of 5.20.0, too.

Some notable changes include:

- Archive::Tar has been upgraded to version 2.04.

  Tests can now be run in parallel.

- attributes has been upgraded to version 0.27.

  The usage of `memEQs` in the XS has been corrected. [GH #14072] <https://github.com/Perl/perl5/issues/14072>

  Avoid reading beyond the end of a buffer. [perl #122629]

- B has been upgraded to version 1.58.

  It provides a new `B::safename` function, based on the existing `B::GV->SAFENAME`, that converts `\cOPEN` to `^OPEN`.

  Nulled COPs are now of class `B::COP`, rather than `B::OP`.

  `B::REGEXP` objects now provide a `qr_anoncv` method for accessing the implicit CV associated with `qr//` things containing code blocks, and a `compflags` method that returns the pertinent flags originating from the `qr//blahblah` op.

  `B::PMOP` now provides a `pmregexp` method returning a `B::REGEXP` object. Two new classes, `B::PADNAME` and `B::PADNAMELIST`, have been introduced.

A bug where, after an ithread creation or pseudofork, special/immortal SVs in the child ithread/pseudoprocess did not have the correct class of `B::SPECIAL`, has been fixed. The `id` and `outid` PADLIST methods have been added.

- B::Concise has been upgraded to version 0.996.

  Null ops that are part of the execution chain are now given sequence numbers.

  Private flags for nulled ops are now dumped with mnemonics as they would be for the non-nulled counterparts.

- B::Deparse has been upgraded to version 1.35.

  It now deparses `+sub : attr { ... }` correctly at the start of a statement. Without the initial `+`, `sub` would be a statement label.

  `BEGIN` blocks are now emitted in the right place most of the time, but the change unfortunately introduced a regression, in that `BEGIN` blocks occurring just before the end of the enclosing block may appear below it instead.

  `B::Deparse` no longer puts erroneous `local` here and there, such as for `LIST = tr/a//d`. [perl #119815]

  Adjacent `use` statements are no longer accidentally nested if one contains a `do` block. [perl #115066]

  Parenthesised arrays in lists passed to `\` are now correctly deparsed with parentheses (*e.g.*, `\(@a, (@b), @c)` now retains the parentheses around `@b`), thus preserving the flattening behavior of referenced parenthesised arrays. Formerly, it only worked for one array: `\(@a)`.

  `local our` is now deparsed correctly, with the `our` included.

  `for($foo; !$bar; $baz) {...}` was deparsed without the `!` (or `not`). This has been fixed.

  Core keywords that conflict with lexical subroutines are now deparsed with the `CORE::` prefix.

  `foreach state $x (...) {...}` now deparses correctly with `state` and not `my`.

  `our @array = split(...)` now deparses correctly with `our` in those cases where the assignment is optimized away.

  It now deparses `our(`*`LIST`*`)` and typed lexical (`my Dog $spot`) correctly.

  Deparse `$#_` as that instead of as `$#{_}`. [GH #14545] <https://github.com/Perl/perl5/issues/14545>

  BEGIN blocks at the end of the enclosing scope are now deparsed in the right place. [perl #77452]

  BEGIN blocks were sometimes deparsed as `__ANON__`, but are now always called BEGIN.

  Lexical subroutines are now fully deparsed. [perl #116553]

  `Anything =~ y///r` with `/r` no longer omits the left-hand operand.

  The op trees that make up regexp code blocks are now deparsed for real. Formerly, the original string that made up the regular expression was used. That caused problems with `qr/(?{<<heredoc})/` and multiline code blocks, which were deparsed incorrectly. [perl #123217] [perl #115256]

  `$;` at the end of a statement no longer loses its semicolon. [perl #123357]

  Some cases of subroutine declarations stored in the stash in shorthand form were being omitted.

  Non-ASCII characters are now consistently escaped in strings, instead of some of the time. (There are still outstanding problems with regular expressions and identifiers that have not been fixed.)

  When prototype sub calls are deparsed with `&` (*e.g.*, under the **-P** option), `scalar` is now added where appropriate, to force the scalar context implied by the prototype.

  `require(foo())`, `do(foo())`, `goto(foo())` and similar constructs with loop controls are

now deparsed correctly. The outer parentheses are not optional.

Whitespace is no longer escaped in regular expressions, because it was getting erroneously escaped within `(?x:...)` sections.

`sub foo { foo() }` is now deparsed with those mandatory parentheses.

`/@array/` is now deparsed as a regular expression, and not just `@array`.

`/@{-}/`, `/@{+}/` and `$#{1}` are now deparsed with the braces, which are mandatory in these cases.

In deparsing feature bundles, `B::Deparse` was emitting `no feature;` first instead of `no feature ':all';`. This has been fixed.

`chdir FH` is now deparsed without quotation marks.

`\my @a` is now deparsed without parentheses. (Parenthese would flatten the array.)

`system` and `exec` followed by a block are now deparsed correctly. Formerly there was an erroneous `do` before the block.

`use constant QR => qr/.../flags` followed by `"" =~ QR` is no longer without the flags.

Deparsing `BEGIN { undef &foo }` with the **-w** switch enabled started to emit 'uninitialized' warnings in Perl 5.14. This has been fixed.

Deparsing calls to subs with a `(;+)` prototype resulted in an infinite loop. The `(;$)` `(_)` and `(;_)` prototypes were given the wrong precedence, causing `foo($a<$b)` to be deparsed without the parentheses.

Deparse now provides a defined state sub in inner subs.

- B::Op_private has been added.

  B::Op_private provides detailed information about the flags used in the `op_private` field of perl opcodes.

- bigint, bignum, bigrat have been upgraded to version 0.39.

  Document in CAVEATS that using strings as numbers won't always invoke the big number overloading, and how to invoke it. [rt.perl.org #123064]

- Carp has been upgraded to version 1.36.

  `Carp::Heavy` now ignores version mismatches with Carp if Carp is newer than 1.12, since `Carp::Heavy`'s guts were merged into Carp at that point. [GH #13708] <https://github.com/Perl/perl5/issues/13708>

  Carp now handles non-ASCII platforms better.

  Off-by-one error fix for Perl < 5.14.

- constant has been upgraded to version 1.33.

  It now accepts fully-qualified constant names, allowing constants to be defined in packages other than the caller.

- CPAN has been upgraded to version 2.11.

  Add support for `Cwd::getdcwd()` and introduce workaround for a misbehavior seen on Strawberry Perl 5.20.1.

  Fix `chdir()` after building dependencies bug.

  Introduce experimental support for plugins/hooks.

  Integrate the `App::Cpan` sources.

  Do not check recursion on optional dependencies.

  Sanity check *META.yml* to contain a hash. [cpan #95271] <https://rt.cpan.org/Ticket/Display.html?id=95271>

- CPAN::Meta::Requirements has been upgraded to version 2.132.

  Works around limitations in `version::vpp` detecting v−string magic and adds support for forthcoming ExtUtils::MakeMaker bootstrap *version.pm* for Perls older than 5.10.0.

- Data::Dumper has been upgraded to version 2.158.

  Fixes CVE−2014−4330 by adding a configuration variable/option to limit recursion when dumping deep data structures.

  Changes to resolve Coverity issues. XS dumps incorrectly stored the name of code references stored in a GLOB. [GH #13911] <https://github.com/Perl/perl5/issues/13911>

- DynaLoader has been upgraded to version 1.32.

  Remove `dl_nonlazy` global if unused in Dynaloader. [perl #122926]

- Encode has been upgraded to version 2.72.

  `piconv` now has better error handling when the encoding name is nonexistent, and a build breakage when upgrading Encode in perl−5.8.2 and earlier has been fixed.

  Building in C++ mode on Windows now works.

- Errno has been upgraded to version 1.23.

  Add `-P` to the preprocessor command-line on GCC 5. GCC added extra line directives, breaking parsing of error code definitions. [rt.perl.org #123784]

- experimental has been upgraded to version 0.013.

  Hardcodes features for Perls older than 5.15.7.

- ExtUtils::CBuilder has been upgraded to version 0.280221.

  Fixes a regression on Android. [GH #14064] <https://github.com/Perl/perl5/issues/14064>

- ExtUtils::Manifest has been upgraded to version 1.70.

  Fixes a bug with `maniread()`'s handling of quoted filenames and improves `manifind()` to follow symlinks. [GH #14003] <https://github.com/Perl/perl5/issues/14003>

- ExtUtils::ParseXS has been upgraded to version 3.28.

  Only declare `file` unused if we actually define it. Improve generated `RETVAL` code generation to avoid repeated references to `ST(0)`. [perl #123278] Broaden and document the `/OBJ$/` to `/REF$/` typemap optimization for the `DESTROY` method. [perl #123418]

- Fcntl has been upgraded to version 1.13.

  Add support for the Linux pipe buffer size `fcntl()` commands.

- File::Find has been upgraded to version 1.29.

  `find()` and `finddepth()` will now warn if passed inappropriate or misspelled options.

- File::Glob has been upgraded to version 1.24.

  Avoid `SvIV()` expanding to call `get_sv()` three times in a few places. [perl #123606]

- HTTP::Tiny has been upgraded to version 0.054.

  `keep_alive` is now fork-safe and thread-safe.

- IO has been upgraded to version 1.35.

  The XS implementation has been fixed for the sake of older Perls.

- IO::Socket has been upgraded to version 1.38.

  Document the limitations of the `connected()` method. [perl #123096]

- IO::Socket::IP has been upgraded to version 0.37.

  A better fix for subclassing `connect()`. [cpan #95983] <https://rt.cpan.org/Ticket/Display.html?id=95983> [cpan #97050] <https://rt.cpan.org/Ticket/Display.html?id=97050>

Implements Timeout for `connect()`. [cpan #92075] <https://rt.cpan.org/Ticket/Display.html?id=92075>

- The libnet collection of modules has been upgraded to version 3.05.

  Support for IPv6 and SSL to `Net::FTP`, `Net::NNTP`, `Net::POP3` and `Net::SMTP`. Improvements in `Net::SMTP` authentication.

- Locale::Codes has been upgraded to version 3.34.

  Fixed a bug in the scripts used to extract data from spreadsheets that prevented the SHP currency code from being found. [cpan #94229] <https://rt.cpan.org/Ticket/Display.html?id=94229>

  New codes have been added.

- Math::BigInt has been upgraded to version 1.9997.

  Synchronize POD changes from the CPAN release. `Math::BigFloat->blog(x)` would sometimes return `blog(2*x)` when the accuracy was greater than 70 digits. The result of `Math::BigFloat->bdiv()` in list context now satisfies `x = quotient * divisor + remainder`.

  Correct handling of subclasses. [cpan #96254] <https://rt.cpan.org/Ticket/Display.html?id=96254> [cpan #96329] <https://rt.cpan.org/Ticket/Display.html?id=96329>

- Module::Metadata has been upgraded to version 1.000026.

  Support installations on older perls with an ExtUtils::MakeMaker earlier than 6.63_03

- overload has been upgraded to version 1.26.

  A redundant `ref $sub` check has been removed.

- The PathTools module collection has been upgraded to version 3.56.

  A warning from the **gcc** compiler is now avoided when building the XS.

  Don't turn leading `//` into `/` on Cygwin. [perl #122635]

- perl5db.pl has been upgraded to version 1.49.

  The debugger would cause an assertion failure. [GH #14605] <https://github.com/Perl/perl5/issues/14605>

  `fork()` in the debugger under `tmux` will now create a new window for the forked process. [GH #13602] <https://github.com/Perl/perl5/issues/13602>

  The debugger now saves the current working directory on startup and restores it when you restart your program with `R` or `rerun`. [GH #13691] <https://github.com/Perl/perl5/issues/13691>

- PerlIO::scalar has been upgraded to version 0.22.

  Reading from a position well past the end of the scalar now correctly returns end of file. [perl #123443]

  Seeking to a negative position still fails, but no longer leaves the file position set to a negation location.

  `eof()` on a `PerlIO::scalar` handle now properly returns true when the file position is past the 2GB mark on 32−bit systems.

  Attempting to write at file positions impossible for the platform now fail early rather than wrapping at 4GB.

- Pod::Perldoc has been upgraded to version 3.25.

  Filehandles opened for reading or writing now have `:encoding(UTF-8)` set. [cpan #98019] <https://rt.cpan.org/Ticket/Display.html?id=98019>

- POSIX has been upgraded to version 1.53.

  The C99 math functions and constants (for example `acosh`, `isinf`, `isnan`, `round`, `trunc`; `M_E`, `M_SQRT2`, `M_PI`) have been added.

`POSIX::tmpnam()` now produces a deprecation warning. [perl #122005]

- Safe has been upgraded to version 2.39.

  `reval` was not propagating void context properly.

- Scalar-List-Utils has been upgraded to version 1.41.

  A new module, Sub::Util, has been added, containing functions related to CODE refs, including `subname` (inspired by `Sub::Identity`) and `set_subname` (copied and renamed from `Sub::Name`). The use of `GetMagic` in `List::Util::reduce()` has also been fixed. [cpan #63211] <https://rt.cpan.org/Ticket/Display.html?id=63211>

- SDBM_File has been upgraded to version 1.13.

  Simplified the build process. [perl #123413]

- Time::Piece has been upgraded to version 1.29.

  When pretty printing negative `Time::Seconds`, the ''minus'' is no longer lost.

- Unicode::Collate has been upgraded to version 1.12.

  Version 0.67's improved discontiguous contractions is invalidated by default and is supported as a parameter `long_contraction`.

- Unicode::Normalize has been upgraded to version 1.18.

  The XSUB implementation has been removed in favor of pure Perl.

- Unicode::UCD has been upgraded to version 0.61.

  A new function **property_values()** has been added to return a given property's possible values.

  A new function **charprop()** has been added to return the value of a given property for a given code point.

  A new function **charprops_all()** has been added to return the values of all Unicode properties for a given code point.

  A bug has been fixed so that **propaliases()** returns the correct short and long names for the Perl extensions where it was incorrect.

  A bug has been fixed so that **prop_value_aliases()** returns `undef` instead of a wrong result for properties that are Perl extensions.

  This module now works on EBCDIC platforms.

- utf8 has been upgraded to version 1.17

  A mismatch between the documentation and the code in `utf8::downgrade()` was fixed in favor of the documentation. The optional second argument is now correctly treated as a perl boolean (true/false semantics) and not as an integer.

- version has been upgraded to version 0.9909.

  Numerous changes. See the *Changes* file in the CPAN distribution for details.

- Win32 has been upgraded to version 0.51.

  `GetOSName()` now supports Windows 8.1, and building in C++ mode now works.

- Win32API::File has been upgraded to version 0.1202

  Building in C++ mode now works.

- XSLoader has been upgraded to version 0.20.

  Allow XSLoader to load modules from a different namespace. [perl #122455]

### Removed Modules and Pragmata

The following modules (and associated modules) have been removed from the core perl distribution:

- CGI

- Module::Build

## Documentation

### New Documentation

*perlunicook*

This document, by Tom Christiansen, provides examples of handling Unicode in Perl.

### Changes to Existing Documentation

*perlaix*

- A note on long doubles has been added.

*perlapi*

- Note that `SvSetSV` doesn't do set magic.
- `sv_usepvn_flags` − fix documentation to mention the use of `Newx` instead of `malloc`.

  [GH #13835] <https://github.com/Perl/perl5/issues/13835>
- Clarify where `NUL` may be embedded or is required to terminate a string.
- Some documentation that was previously missing due to formatting errors is now included.
- Entries are now organized into groups rather than by the file where they are found.
- Alphabetical sorting of entries is now done consistently (automatically by the POD generator) to make entries easier to find when scanning.

*perldata*

- The syntax of single-character variable names has been brought up-to-date and more fully explained.
- Hexadecimal floating point numbers are described, as are infinity and NaN.

*perlebcdic*

- This document has been significantly updated in the light of recent improvements to EBCDIC support.

*perlfilter*

- Added a LIMITATIONS section.

*perlfunc*

- Mention that `study()` is currently a no-op.
- Calling `delete` or `exists` on array values is now described as ''strongly discouraged'' rather than ''deprecated''.
- Improve documentation of `our`.
- `-l` now notes that it will return false if symlinks aren't supported by the file system. [GH #13695] <https://github.com/Perl/perl5/issues/13695>
- Note that `exec LIST` and `system LIST` may fall back to the shell on Win32. Only the indirect-object syntax `exec PROGRAM LIST` and `system PROGRAM LIST` will reliably avoid using the shell.

  This has also been noted in perlport.

  [GH #13907] <https://github.com/Perl/perl5/issues/13907>

*perlguts*

- The OOK example has been updated to account for COW changes and a change in the storage of the offset.
- Details on C level symbols and libperl.t added.
- Information on Unicode handling has been added
- Information on EBCDIC handling has been added

*perlhack*

- A note has been added about running on platforms with non-ASCII character sets
- A note has been added about performance testing

*perlhacktips*

- Documentation has been added illustrating the perils of assuming that there is no change to the contents of static memory pointed to by the return values of Perl's wrappers for C library functions.
- Replacements for `tmpfile`, `atoi`, `strtol`, and `strtoul` are now recommended.
- Updated documentation for the `test.valgrind make` target. [GH #13658] <https://github.com/Perl/perl5/issues/13658>
- Information is given about writing test files portably to non-ASCII platforms.
- A note has been added about how to get a C language stack backtrace.

*perlhpux*

- Note that the message ''Redeclaration of ''sendpath'' with a different storage class specifier'' is harmless.

*perllocale*

- Updated for the enhancements in v5.22, along with some clarifications.

*perlmodstyle*

- Instead of pointing to the module list, we are now pointing to PrePAN <http://prepan.org/>.

*perlop*

- Updated for the enhancements in v5.22, along with some clarifications.

*perlpodspec*

- The specification of the pod language is changing so that the default encoding of pods that aren't in UTF-8 (unless otherwise indicated) is CP1252 instead of ISO 8859-1 (Latin1).

*perlpolicy*

- We now have a code of conduct for the *p5p* mailing list, as documented in ''STANDARDS OF CONDUCT'' in perlpolicy.
- The conditions for marking an experimental feature as non-experimental are now set out.
- Clarification has been made as to what sorts of changes are permissible in maintenance releases.

*perlport*

- Out-of-date VMS-specific information has been fixed and/or simplified.
- Notes about EBCDIC have been added.

*perlre*

- The description of the `/x` modifier has been clarified to note that comments cannot be continued onto the next line by escaping them; and there is now a list of all the characters that are considered whitespace by this modifier.
- The new `/n` modifier is described.
- A note has been added on how to make bracketed character class ranges portable to non-ASCII machines.

*perlrebackslash*

- Added documentation of `\b{sb}`, `\b{wb}`, `\b{gcb}`, and `\b{g}`.

*perlrecharclass*

- Clarifications have been added to ''Character Ranges'' in perlrecharclass to the effect `[A-Z]`, `[a-z]`, `[0-9]` and any subranges thereof in regular expression bracketed character classes are guaranteed to match exactly what a naive English speaker would expect them to match, even on platforms (such as EBCDIC) where perl has to do extra work to accomplish this.

- The documentation of Bracketed Character Classes has been expanded to cover the improvements in `qr/[\N{named sequence}]/` (see under ‘‘Selected Bug Fixes’’).

*perlref*

- A new section has been added Assigning to References

*perlsec*

- Comments added on algorithmic complexity and tied hashes.

*perlsyn*

- An ambiguity in the documentation of the `...` statement has been corrected. [GH #14054] <https://github.com/Perl/perl5/issues/14054>
- The empty conditional in `for` and `while` is now documented in perlsyn.

*perlunicode*

- This has had extensive revisions to bring it up-to-date with current Unicode support and to make it more readable. Notable is that Unicode 7.0 changed what it should do with non-characters. Perl retains the old way of handling for reasons of backward compatibility. See ‘‘Noncharacter code points’’ in perlunicode.

*perluniintro*

- Advice for how to make sure your strings and regular expression patterns are interpreted as Unicode has been updated.

*perlvar*

- `$]` is no longer listed as being deprecated. Instead, discussion has been added on the advantages and disadvantages of using it versus `$^V`. `$OLD_PERL_VERSION` was re-added to the documentation as the long form of `$]`.
- `${^ENCODING}` is now marked as deprecated.
- The entry for `%^H` has been clarified to indicate it can only handle simple values.

*perlvms*

- Out-of-date and/or incorrect material has been removed.
- Updated documentation on environment and shell interaction in VMS.

*perlxs*

- Added a discussion of locale issues in XS code.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Bad symbol for scalar

  (P) An internal request asked to add a scalar entry to something that wasn’t a symbol table entry.

- Can’t use a hash as a reference

  (F) You tried to use a hash as a reference, as in `%foo->{"bar"}` or `%$ref->{"hello"}`. Versions of perl <= 5.6.1 used to allow this syntax, but shouldn’t have.

- Can’t use an array as a reference

  (F) You tried to use an array as a reference, as in `@foo->[23]` or `@$ref->[99]`. Versions of perl <= 5.6.1 used to allow this syntax, but shouldn’t have.

- Can’t use ’defined(@array)’ (Maybe you should just omit the **defined()**?)

  (F) `defined()` is not useful on arrays because it checks for an undefined *scalar* value. If you want to see if the array is empty, just use `if (@array) { # not empty }` for example.

- Can't use 'defined(%hash)' (Maybe you should just omit the **defined()**?)

  (F) `defined()` is not usually right on hashes.

  Although `defined %hash` is false on a plain not-yet-used hash, it becomes true in several non-obvious circumstances, including iterators, weak references, stash names, even remaining true after `undef %hash`. These things make `defined %hash` fairly useless in practice, so it now generates a fatal error.

  If a check for non-empty is what you wanted then just put it in boolean context (see ''Scalar values'' in perldata):

  ```
  if (%hash) {
     # not empty
  }
  ```

  If you had `defined %Foo::Bar::QUUX` to check whether such a package variable exists then that's never really been reliable, and isn't a good way to enquire about the features of a package, or whether it's loaded, etc.

- Cannot chr `%f`

  (F) You passed an invalid number (like an infinity or not-a-number) to `chr`.

- Cannot compress `%f` in pack

  (F) You tried converting an infinity or not-a-number to an unsigned character, which makes no sense.

- Cannot pack `%f` with '%c'

  (F) You tried converting an infinity or not-a-number to a character, which makes no sense.

- Cannot print `%f` with '%c'

  (F) You tried printing an infinity or not-a-number as a character (`%c`), which makes no sense. Maybe you meant `'%s'`, or just stringifying it?

- charnames alias definitions may not contain a sequence of multiple spaces

  (F) You defined a character name which had multiple space characters in a row. Change them to single spaces. Usually these names are defined in the `:alias` import argument to `use charnames`, but they could be defined by a translator installed into `$^H{charnames}`. See ''CUSTOM ALIASES'' in charnames.

- charnames alias definitions may not contain trailing white-space

  (F) You defined a character name which ended in a space character. Remove the trailing space(s). Usually these names are defined in the `:alias` import argument to `use charnames`, but they could be defined by a translator installed into `$^H{charnames}`. See ''CUSTOM ALIASES'' in charnames.

- :const is not permitted on named subroutines

  (F) The `const` attribute causes an anonymous subroutine to be run and its value captured at the time that it is cloned. Named subroutines are not cloned like this, so the attribute does not make sense on them.

- Hexadecimal float: internal error

  (F) Something went horribly bad in hexadecimal float handling.

- Hexadecimal float: unsupported long double format

  (F) You have configured Perl to use long doubles but the internals of the long double format are unknown, therefore the hexadecimal float output is impossible.

- Illegal suidscript

  (F) The script run under suidperl was somehow illegal.

- In '(?...)', the '(' and '?' must be adjacent in regex; marked by <-- HERE in m/%s/

  (F) The two-character sequence `"(?"` in this context in a regular expression pattern should be an indivisible token, with nothing intervening between the `"("` and the `"?"`, but you separated them.

- In '(*VERB...)', the '(' and '*' must be adjacent in regex; marked by <-- HERE in m/%s/

  (F) The two-character sequence `"(*"` in this context in a regular expression pattern should be an indivisible token, with nothing intervening between the `"("` and the `"*"`, but you separated them.

- Invalid quantifier in {,} in regex; marked by <-- HERE in m/%s/

  (F) The pattern looks like a {min,max} quantifier, but the min or max could not be parsed as a valid number: either it has leading zeroes, or it represents too big a number to cope with. The <-- HERE shows where in the regular expression the problem was discovered. See perlre.

- '%s' is an unknown bound type in regex

  (F) You used `\b{...}` or `\B{...}` and the `...` is not known to Perl. The current valid ones are given in ''\b{}, \b, \B{}, \B'' in perlrebackslash.

- Missing or undefined argument to require

  (F) You tried to call `require` with no argument or with an undefined value as an argument. `require` expects either a package name or a file-specification as an argument. See ''require'' in perlfunc.

  Formerly, `require` with no argument or `undef` warned about a Null filename.

*New Warnings*

- \C is deprecated in regex

  (D deprecated) The `/\C/` character class was deprecated in v5.20, and now emits a warning. It is intended that it will become an error in v5.24. This character class matches a single byte even if it appears within a multi-byte character, breaks encapsulation, and can corrupt UTF-8 strings.

- ''%s'' is more clearly written simply as ''%s'' in regex; marked by <-- HERE in m/%s/

  (W regexp) (only under `use re 'strict'` or within `(?[...])`)

  You specified a character that has the given plainer way of writing it, and which is also portable to platforms running with different character sets.

- Argument ''%s'' treated as 0 in increment (++)

  (W numeric) The indicated string was fed as an argument to the `++` operator which expects either a number or a string matching `/^[a-zA-Z]*[0-9]*\z/`. See ''Auto-increment and Auto-decrement'' in perlop for details.

- Both or neither range ends should be Unicode in regex; marked by <-- HERE in m/%s/

  (W regexp) (only under `use re 'strict'` or within `(?[...])`)

  In a bracketed character class in a regular expression pattern, you had a range which has exactly one end of it specified using `\N{}`, and the other end is specified using a non-portable mechanism. Perl treats the range as a Unicode range, that is, all the characters in it are considered to be the Unicode characters, and which may be different code points on some platforms Perl runs on. For example, `[\N{U+06}-\x08]` is treated as if you had instead said `[\N{U+06}-\N{U+08}]`, that is it matches the characters whose code points in Unicode are 6, 7, and 8. But that `\x08` might indicate that you meant something different, so the warning gets raised.

- Can't do %s(''%s'') on non-UTF-8 locale; resolved to ''%s''.

  (W locale) You are 1) running under "`use locale`"; 2) the current locale is not a UTF-8 one; 3) you tried to do the designated case-change operation on the specified Unicode character; and 4) the result of this operation would mix Unicode and locale rules, which likely conflict.

  The warnings category `locale` is new.

- :const is experimental

  (S experimental::const_attr) The `const` attribute is experimental. If you want to use the feature, disable the warning with `no warnings 'experimental::const_attr'`, but know that in doing so you are taking the risk that your code may break in a future Perl version.

- gmtime(%f) failed

  (W overflow) You called `gmtime` with a number that it could not handle: too large, too small, or NaN. The returned value is `undef`.

- Hexadecimal float: exponent overflow

  (W overflow) The hexadecimal floating point has larger exponent than the floating point supports.

- Hexadecimal float: exponent underflow

  (W overflow) The hexadecimal floating point has smaller exponent than the floating point supports.

- Hexadecimal float: mantissa overflow

  (W overflow) The hexadecimal floating point literal had more bits in the mantissa (the part between the `0x` and the exponent, also known as the fraction or the significand) than the floating point supports.

- Hexadecimal float: precision loss

  (W overflow) The hexadecimal floating point had internally more digits than could be output. This can be caused by unsupported long double formats, or by 64−bit integers not being available (needed to retrieve the digits under some configurations).

- Locale '%s' may not work well.%s

  (W locale) You are using the named locale, which is a non−UTF−8 one, and which perl has determined is not fully compatible with what it can handle. The second `%s` gives a reason.

  The warnings category `locale` is new.

- localtime(%f) failed

  (W overflow) You called `localtime` with a number that it could not handle: too large, too small, or NaN. The returned value is `undef`.

- Negative repeat count does nothing

  (W numeric) You tried to execute the `x` repetition operator fewer than 0 times, which doesn't make sense.

- NO-BREAK SPACE in a charnames alias definition is deprecated

  (D deprecated) You defined a character name which contained a no-break space character. Change it to a regular space. Usually these names are defined in the `:alias` import argument to `use charnames`, but they could be defined by a translator installed into `$^H{charnames}`. See ''CUSTOM ALIASES'' in charnames.

- Non-finite repeat count does nothing

  (W numeric) You tried to execute the `x` repetition operator `Inf` (or `−Inf`) or NaN times, which doesn't make sense.

- PerlIO layer ':win32' is experimental

  (S experimental::win32_perlio) The `:win32` PerlIO layer is experimental. If you want to take the risk of using this layer, simply disable this warning:

  ```
  no warnings "experimental::win32_perlio";
  ```

- Ranges of ASCII printables should be some subset of ''0−9'', ''A−Z'', or ''a−z'' in regex; marked by <−− HERE in m/%s/

  (W regexp) (only under `use re 'strict'` or within `(?[...])`)

  Stricter rules help to find typos and other errors. Perhaps you didn't even intend a range here, if

the "-" was meant to be some other character, or should have been escaped (like "\-"). If you did intend a range, the one that was used is not portable between ASCII and EBCDIC platforms, and doesn't have an obvious meaning to a casual reader.

```
[3-7]    # OK; Obvious and portable
[d-g]    # OK; Obvious and portable
[A-Y]    # OK; Obvious and portable
[A-z]    # WRONG; Not portable; not clear what is meant
[a-Z]    # WRONG; Not portable; not clear what is meant
[%-.]    # WRONG; Not portable; not clear what is meant
[\x41-Z] # WRONG; Not portable; not obvious to non-geek
```

(You can force portability by specifying a Unicode range, which means that the endpoints are specified by \N{...}, but the meaning may still not be obvious.) The stricter rules require that ranges that start or stop with an ASCII character that is not a control have all their endpoints be a literal character, and not some escape sequence (like "\x41"), and the ranges must be all digits, or all uppercase letters, or all lowercase letters.

- Ranges of digits should be from the same group in regex; marked by <-- HERE in m/%s/

  (W regexp) (only under `use re 'strict'` or within `(?[...])`)

  Stricter rules help to find typos and other errors. You included a range, and at least one of the end points is a decimal digit. Under the stricter rules, when this happens, both end points should be digits in the same group of 10 consecutive digits.

- Redundant argument in %s

  (W redundant) You called a function with more arguments than were needed, as indicated by information within other arguments you supplied (*e.g.* a printf format). Currently only emitted when a printf-type format required fewer arguments than were supplied, but might be used in the future for *e.g.* ''pack'' in perlfunc.

  The warnings category `redundant` is new. See also [GH #13534] <https://github.com/Perl/perl5/issues/13534>.

- Replacement list is longer than search list

  This is not a new diagnostic, but in earlier releases was accidentally not displayed if the transliteration contained wide characters. This is now fixed, so that you may see this diagnostic in places where you previously didn't (but should have).

- Use of \b{} for non-UTF-8 locale is wrong. Assuming a UTF-8 locale

  (W locale) You are matching a regular expression using locale rules, and a Unicode boundary is being matched, but the locale is not a Unicode one. This doesn't make sense. Perl will continue, assuming a Unicode (UTF-8) locale, but the results could well be wrong except if the locale happens to be ISO-8859-1 (Latin1) where this message is spurious and can be ignored.

  The warnings category `locale` is new.

- Using /u for '%s' instead of /%s in regex; marked by <-- HERE in m/%s/

  (W regexp) You used a Unicode boundary (\b{...} or \B{...}) in a portion of a regular expression where the character set modifiers /a or /aa are in effect. These two modifiers indicate an ASCII interpretation, and this doesn't make sense for a Unicode definition. The generated regular expression will compile so that the boundary uses all of Unicode. No other portion of the regular expression is affected.

- The bitwise feature is experimental

  (S experimental::bitwise) This warning is emitted if you use bitwise operators (`& | ^ ~ &. |. ^. ~.`) with the ''bitwise'' feature enabled. Simply suppress the warning if you want to use the feature, but know that in doing so you are taking the risk of using an experimental feature which may change or be removed in a future Perl version:

```
no warnings "experimental::bitwise";
use feature "bitwise";
$x |.= $y;
```

- Unescaped left brace in regex is deprecated, passed through in regex; marked by <−− HERE in m/%s/

  (D deprecated, regexp) You used a literal `"{"` character in a regular expression pattern. You should change to use `"\{"` instead, because a future version of Perl (tentatively v5.26) will consider this to be a syntax error. If the pattern delimiters are also braces, any matching right brace (`"}"`) should also be escaped to avoid confusing the parser, for example,

  ```
  qr{abc\{def\}ghi}
  ```

- Use of literal non-graphic characters in variable names is deprecated

  (D deprecated) Using literal non-graphic (including control) characters in the source to refer to the *^FOO* variables, like `$^X` and `${^GLOBAL_PHASE}` is now deprecated.

- Useless use of attribute ‘‘const’’

  (W misc) The `const` attribute has no effect except on anonymous closure prototypes. You applied it to a subroutine via attributes.pm. This is only useful inside an attribute handler for an anonymous subroutine.

- Useless use of /d modifier in transliteration operator

  This is not a new diagnostic, but in earlier releases was accidentally not displayed if the transliteration contained wide characters. This is now fixed, so that you may see this diagnostic in places where you previously didn’t (but should have).

- ‘‘use re ’strict’’’ is experimental

  (S experimental::re_strict) The things that are different when a regular expression pattern is compiled under `'strict'` are subject to change in future Perl releases in incompatible ways; there are also proposals to change how to enable strict checking instead of using this subpragma. This means that a pattern that compiles today may not in a future Perl release. This warning is to alert you to that risk.

- Warning: unable to close filehandle properly: `%s`

  Warning: unable to close filehandle `%s` properly: `%s`

  (S io) Previously, perl silently ignored any errors when doing an implicit close of a filehandle, *i.e.* where the reference count of the filehandle reached zero and the user’s code hadn’t already called `close()`; *e.g.*

  ```
  {
      open my $fh, '>', $file  or die "open: '$file': $!\n";
      print $fh, $data  or die;
  } # implicit close here
  ```

  In a situation such as disk full, due to buffering, the error may only be detected during the final close, so not checking the result of the close is dangerous.

  So perl now warns in such situations.

- Wide character (U+%X) in `%s`

  (W locale) While in a single-byte locale (*i.e.*, a non−UTF−8 one), a multi-byte character was encountered. Perl considers this character to be the specified Unicode code point. Combining non−UTF−8 locales and Unicode is dangerous. Almost certainly some characters will have two different representations. For example, in the ISO 8859−7 (Greek) locale, the code point 0xC3 represents a Capital Gamma. But so also does 0x393. This will make string comparisons unreliable.

  You likely need to figure out how this multi-byte character got mixed up with your single-byte locale (or perhaps you thought you had a UTF−8 locale, but Perl disagrees).

  The warnings category `locale` is new.

### Changes to Existing Diagnostics

- <> should be quotes

  This warning has been changed to <> at require-statement should be quotes to make the issue more identifiable.

- Argument ‘‘%s’’ isn’t numeric%s

  The perldiag entry for this warning has added this clarifying note:

  ```
  Note that for the Inf and NaN (infinity and not-a-number) the
  definition of "numeric" is somewhat unusual: the strings themselves
  (like "Inf") are considered numeric, and anything following them is
  considered non-numeric.
  ```

- Global symbol ‘‘%s’’ requires explicit package name

  This message has had ’(did you forget to declare ‘‘my `%s`’’?)’ appended to it, to make it more helpful to new Perl programmers. [GH #13732] <https://github.com/Perl/perl5/issues/13732>

- ’‘‘my’’ variable &foo::bar can’t be in a package’ has been reworded to say ’subroutine’ instead of ’variable’.

- \N{} in character class restricted to one character in regex; marked by <-- HERE in m/%s/

  This message has had *character class* changed to *inverted character class or as a range end-point is* to reflect improvements in `qr/[\N{named sequence}]/` (see under ‘‘Selected Bug Fixes’’).

- panic: frexp

  This message has had ’: `%f`’ appended to it, to show what the offending floating point number is.

- *Possible precedence problem on bitwise %c operator* reworded as Possible precedence problem on bitwise `%s` operator.

- Unsuccessful `%s` on filename containing newline

  This warning is now only produced when the newline is at the end of the filename.

- "Variable `%s` will not stay shared‘‘ has been changed to say ’’Subroutine" when it is actually a lexical sub that will not stay shared.

- Variable length lookbehind not implemented in regex m/%s/

  The perldiag entry for this warning has had information about Unicode behavior added.

### Diagnostic Removals

- ‘‘Ambiguous use of −foo resolved as −&**foo()**’’

  There is actually no ambiguity here, and this impedes the use of negated constants; *e.g.*, `-Inf`.

- ‘‘Constant is not a FOO reference’’

  Compile-time checking of constant dereferencing (*e.g.*, `my_constant->()`) has been removed, since it was not taking overloading into account. [GH #9891] <https://github.com/Perl/perl5/issues/9891> [GH #14044] <https://github.com/Perl/perl5/issues/14044>

## Utility Changes

### *find2perl*, *s2p* and *a2p* removal

- The *x2p/* directory has been removed from the Perl core.

  This removes find2perl, s2p and a2p. They have all been released to CPAN as separate distributions (`App::find2perl`, `App::s2p`, `App::a2p`).

### h2ph

- *h2ph* now handles hexadecimal constants in the compiler’s predefined macro definitions, as visible in `$Config{cppsymbols}`. [GH #14491] <https://github.com/Perl/perl5/issues/14491>.

### encguess

- No longer depends on non-core modules.

## Configuration and Compilation

- *Configure* now checks for `lrintl()`, `lroundl()`, `llrintl()`, and `llroundl()`.
- *Configure* with `-Dmksymlinks` should now be faster. [GH #13890] <https://github.com/Perl/perl5/issues/13890>.
- The `pthreads` and `cl` libraries will be linked by default if present. This allows XS modules that require threading to work on non-threaded perls. Note that you must still pass `-Dusethreads` if you want a threaded perl.
- To get more precision and range for floating point numbers one can now use the GCC quadmath library which implements the quadruple precision floating point numbers on x86 and IA-64 platforms. See *INSTALL* for details.
- MurmurHash64A and MurmurHash64B can now be configured as the internal hash function.
- `make test.valgrind` now supports parallel testing.

  For example:

  ```
  TEST_JOBS=9 make test.valgrind
  ```

  See ''valgrind'' in perlhacktips for more information.

  [GH #13658] <https://github.com/Perl/perl5/issues/13658>
- The MAD (Misc Attribute Decoration) build option has been removed

  This was an unmaintained attempt at preserving the Perl parse tree more faithfully so that automatic conversion of Perl 5 to Perl 6 would have been easier.

  This build-time configuration option had been unmaintained for years, and had probably seriously diverged on both Perl 5 and Perl 6 sides.
- A new compilation flag, `-DPERL_OP_PARENT` is available. For details, see the discussion below at ''Internal Changes''.
- Pathtools no longer tries to load XS on miniperl. This speeds up building perl slightly.

## Testing

- *t/porting/re_context.t* has been added to test that utf8 and its dependencies only use the subset of the `$1..$n` capture vars that `Perl_save_re_context()` is hard-coded to localize, because that function has no efficient way of determining at runtime what vars to localize.
- Tests for performance issues have been added in the file *t/perf/taint.t*.
- Some regular expression tests are written in such a way that they will run very slowly if certain optimizations break. These tests have been moved into new files, *t/re/speed.t* and *t/re/speed_thr.t*, and are run with a `watchdog()`.
- `test.pl` now allows `plan skip_all => $reason`, to make it more compatible with `Test::More`.
- A new test script, *op/infnan.t*, has been added to test if infinity and NaN are working correctly. See ''Infinity and NaN (not-a-number) handling improved''.

## Platform Support

### Regained Platforms

IRIX and Tru64 platforms are working again.

Some `make test` failures remain: [GH #14557] <https://github.com/Perl/perl5/issues/14557> and [GH #14727] <https://github.com/Perl/perl5/issues/14727> for IRIX; [GH #14629] <https://github.com/Perl/perl5/issues/14629>, [cpan #99605] <https://rt.cpan.org/Public/Bug/Display.html?id=99605>, and [cpan #104836] <https://rt.cpan.org/Ticket/Display.html?id=104836> for Tru64.

z/OS running EBCDIC Code Page 1047

Core perl now works on this EBCDIC platform. Earlier perls also worked, but, even though support wasn't officially withdrawn, recent perls would not compile and run well. Perl 5.20 would work, but had many bugs which have now been fixed. Many CPAN modules that ship with Perl

still fail tests, including `Pod::Simple`. However the version of `Pod::Simple` currently on CPAN should work; it was fixed too late to include in Perl 5.22. Work is under way to fix many of the still-broken CPAN modules, which likely will be installed on CPAN when completed, so that you may not have to wait until Perl 5.24 to get a working version.

## Discontinued Platforms

NeXTSTEP/OPENSTEP

NeXTSTEP was a proprietary operating system bundled with NeXT's workstations in the early to mid 90s; OPENSTEP was an API specification that provided a NeXTSTEP-like environment on a non-NeXTSTEP system. Both are now long dead, so support for building Perl on them has been removed.

## Platform-Specific Notes

EBCDIC

Special handling is required of the perl interpreter on EBCDIC platforms to get `qr/[i-j]/` to match only `"i"` and `"j"`, since there are 7 characters between the code points for `"i"` and `"j"`. This special handling had only been invoked when both ends of the range are literals. Now it is also invoked if any of the `\N{...}` forms for specifying a character by name or Unicode code point is used instead of a literal. See ''Character Ranges'' in perlrecharclass.

HP-UX

The archname now distinguishes use64bitint from use64bitall.

Android

Build support has been improved for cross-compiling in general and for Android in particular.

VMS

- When spawning a subprocess without waiting, the return value is now the correct PID.
- Fix a prototype so linking doesn't fail under the VMS C++ compiler.
- `finite`, `finitel`, and `isfinite` detection has been added to `configure.com`, environment handling has had some minor changes, and a fix for legacy feature checking status.

Win32

- *miniperl.exe* is now built with `-fno-strict-aliasing`, allowing 64-bit builds to complete on GCC 4.8. [GH #14556] <https://github.com/Perl/perl5/issues/14556>
- `nmake minitest` now works on Win32. Due to dependency issues you need to build `nmake test-prep` first, and a small number of the tests fail. [GH #14318] <https://github.com/Perl/perl5/issues/14318>
- Perl can now be built in C++ mode on Windows by setting the makefile macro `USE_CPLUSPLUS` to the value ''define''.
- The list form of piped open has been implemented for Win32. Note: unlike `system LIST` this does not fall back to the shell. [GH #13574] <https://github.com/Perl/perl5/issues/13574>
- New `DebugSymbols` and `DebugFull` configuration options added to Windows makefiles.
- Previously, compiling XS modules (including CPAN ones) using Visual C++ for Win64 resulted in around a dozen warnings per file from *hv_func.h*. These warnings have been silenced.
- Support for building without PerlIO has been removed from the Windows makefiles. Non-PerlIO builds were all but deprecated in Perl 5.18.0 and are already not supported by *Configure* on POSIX systems.
- Between 2 and 6 milliseconds and seven I/O calls have been saved per attempt to open a perl module for each path in `@INC`.
- Intel C builds are now always built with C99 mode on.
- `%I64d` is now being used instead of `%lld` for MinGW.

- In the experimental `:win32` layer, a crash in `open` was fixed. Also opening */dev/null* (which works under Win32 Perl's default `:unix` layer) was implemented for `:win32`. [GH #13968] <https://github.com/Perl/perl5/issues/13968>
- A new makefile option, `USE_LONG_DOUBLE`, has been added to the Windows dmake makefile for gcc builds only. Set this to ''define'' if you want perl to use long doubles to give more accuracy and range for floating point numbers.

OpenBSD

On OpenBSD, Perl will now default to using the system `malloc` due to the security features it provides. Perl's own malloc wrapper has been in use since v5.14 due to performance reasons, but the OpenBSD project believes the tradeoff is worth it and would prefer that users who need the speed specifically ask for it.

[GH #13888] <https://github.com/Perl/perl5/issues/13888>.

Solaris

- We now look for the Sun Studio compiler in both */opt/solstudio** and */opt/solarisstudio**.
- Builds on Solaris 10 with `-Dusedtrace` would fail early since make didn't follow implied dependencies to build `perldtrace.h`. Added an explicit dependency to `depend`. [GH #13334] <https://github.com/Perl/perl5/issues/13334>
- C99 options have been cleaned up; hints look for `solstudio` as well as `SUNWspro`; and support for native `setenv` has been added.

## Internal Changes

- Experimental support has been added to allow ops in the optree to locate their parent, if any. This is enabled by the non-default build option `-DPERL_OP_PARENT`. It is envisaged that this will eventually become enabled by default, so XS code which directly accesses the `op_sibling` field of ops should be updated to be future-proofed.

  On `PERL_OP_PARENT` builds, the `op_sibling` field has been renamed `op_sibparent` and a new flag, `op_moresib`, added. On the last op in a sibling chain, `op_moresib` is false and `op_sibparent` points to the parent (if any) rather than being `NULL`.

  To make existing code work transparently whether using `PERL_OP_PARENT` or not, a number of new macros and functions have been added that should be used, rather than directly manipulating `op_sibling`.

  For the case of just reading `op_sibling` to determine the next sibling, two new macros have been added. A simple scan through a sibling chain like this:

  ```
  for (; kid->op_sibling; kid = kid->op_sibling) { ... }
  ```

  should now be written as:

  ```
  for (; OpHAS_SIBLING(kid); kid = OpSIBLING(kid)) { ... }
  ```

  For altering optrees, a general-purpose function `op_sibling_splice()` has been added, which allows for manipulation of a chain of sibling ops. By analogy with the Perl function `splice()`, it allows you to cut out zero or more ops from a sibling chain and replace them with zero or more new ops. It transparently handles all the updating of sibling, parent, op_last pointers etc.

  If you need to manipulate ops at a lower level, then three new macros, `OpMORESIB_set`, `OpLASTSIB_set` and `OpMAYBESIB_set` are intended to be a low-level portable way to set `op_sibling` / `op_sibparent` while also updating `op_moresib`. The first sets the sibling pointer to a new sibling, the second makes the op the last sibling, and the third conditionally does the first or second action. Note that unlike `op_sibling_splice()` these macros won't maintain consistency in the parent at the same time (*e.g.* by updating `op_first` and `op_last` where appropriate).

  A C-level `Perl_op_parent()` function and a Perl-level `B::OP::parent()` method have been added. The C function only exists under `PERL_OP_PARENT` builds (using it is build-time error on vanilla perls). `B::OP::parent()` exists always, but on a vanilla build it always returns `NULL`. Under `PERL_OP_PARENT`, they return the parent of the current op, if any. The

variable `$B::OP::does_parent` allows you to determine whether `B` supports retrieving an op's parent.

`PERL_OP_PARENT` was introduced in 5.21.2, but the interface was changed considerably in 5.21.11. If you updated your code before the 5.21.11 changes, it may require further revision. The main changes after 5.21.2 were:

- The `OP_SIBLING` and `OP_HAS_SIBLING` macros have been renamed `OpSIBLING` and `OpHAS_SIBLING` for consistency with other op-manipulating macros.
- The `op_lastsib` field has been renamed `op_moresib`, and its meaning inverted.
- The macro `OpSIBLING_set` has been removed, and has been superseded by `OpMORESIB_set` *et al*.
- The `op_sibling_splice()` function now accepts a null `parent` argument where the splicing doesn't affect the first or last ops in the sibling chain

- Macros have been created to allow XS code to better manipulate the POSIX locale category `LC_NUMERIC`. See ''Locale-related functions and macros'' in perlapi.
- The previous `atoi` *et al* replacement function, `grok_atou`, has now been superseded by `grok_atoUV`. See perlclib for details.
- A new function, `Perl_sv_get_backrefs()`, has been added which allows you retrieve the weak references, if any, which point at an SV.
- The `screaminstr()` function has been removed. Although marked as public API, it was undocumented and had no usage in CPAN modules. Calling it has been fatal since 5.17.0.
- The `newDEFSVOP()`, `block_start()`, `block_end()` and `intro_my()` functions have been added to the API.
- The internal `convert` function in *op.c* has been renamed `op_convert_list` and added to the API.
- The `sv_magic()` function no longer forbids ''ext'' magic on read-only values. After all, perl can't know whether the custom magic will modify the SV or not. [GH #14202] <https://github.com/Perl/perl5/issues/14202>.
- Accessing ''CvPADLIST'' in perlapi on an XSUB is now forbidden.

  The `CvPADLIST` field has been reused for a different internal purpose for XSUBs. So in particular, you can no longer rely on it being NULL as a test of whether a CV is an XSUB. Use `CvISXSUB()` instead.
- SVs of type `SVt_NV` are now sometimes bodiless when the build configuration and platform allow it: specifically, when `sizeof(NV) <= sizeof(IV)`. ''Bodiless'' means that the NV value is stored directly in the head of an SV, without requiring a separate body to be allocated. This trick has already been used for IVs since 5.9.2 (though in the case of IVs, it is always used, regardless of platform and build configuration).
- The `$DB::single`, `$DB::signal` and `$DB::trace` variables now have set− and get-magic that stores their values as IVs, and those IVs are used when testing their values in `pp_dbstate()`. This prevents perl from recursing infinitely if an overloaded object is assigned to any of those variables. [GH #14013] <https://github.com/Perl/perl5/issues/14013>.
- `Perl_tmps_grow()`, which is marked as public API but is undocumented, has been removed from the public API. This change does not affect XS code that uses the `EXTEND_MORTAL` macro to pre-extend the mortal stack.
- Perl's internals no longer sets or uses the `SVs_PADMY` flag. `SvPADMY()` now returns a true value for anything not marked `PADTMP` and `SVs_PADMY` is now defined as 0.
- The macros `SETsv` and `SETsvUN` have been removed. They were no longer used in the core since commit 6f1401dc2a five years ago, and have not been found present on CPAN.
- The `SvFAKE` bit (unused on HVs) got informally reserved by David Mitchell for future work on vtables.

- The `sv_catpvn_flags()` function accepts `SV_CATBYTES` and `SV_CATUTF8` flags, which specify whether the appended string is bytes or UTF-8, respectively. (These flags have in fact been present since 5.16.0, but were formerly not regarded as part of the API.)
- A new opcode class, `METHOP`, has been introduced. It holds information used at runtime to improve the performance of class/object method calls.

  `OP_METHOD` and `OP_METHOD_NAMED` have changed from being `UNOP/SVOP` to being `METHOP`.
- `cv_name()` is a new API function that can be passed a CV or GV. It returns an SV containing the name of the subroutine, for use in diagnostics.

  [GH #12767] <https://github.com/Perl/perl5/issues/12767> [GH #13392] <https://github.com/Perl/perl5/issues/13392>
- `cv_set_call_checker_flags()` is a new API function that works like `cv_set_call_checker()`, except that it allows the caller to specify whether the call checker requires a full GV for reporting the subroutine's name, or whether it could be passed a CV instead. Whatever value is passed will be acceptable to `cv_name()`. `cv_set_call_checker()` guarantees there will be a GV, but it may have to create one on the fly, which is inefficient. [GH #12767] <https://github.com/Perl/perl5/issues/12767>
- `CvGV` (which is not part of the API) is now a more complex macro, which may call a function and reify a GV. For those cases where it has been used as a boolean, `CvHASGV` has been added, which will return true for CVs that notionally have GVs, but without reifying the GV. `CvGV` also returns a GV now for lexical subs. [GH #13392] <https://github.com/Perl/perl5/issues/13392>
- The ''sync_locale'' in perlapi function has been added to the public API. Changing the program's locale should be avoided by XS code. Nevertheless, certain non-Perl libraries called from XS need to do so, such as `Gtk`. When this happens, Perl needs to be told that the locale has changed. Use this function to do so, before returning to Perl.
- The defines and labels for the flags in the `op_private` field of OPs are now auto-generated from data in *regen/op_private*. The noticeable effect of this is that some of the flag output of `Concise` might differ slightly, and the flag output of `perl -Dx` may differ considerably (they both use the same set of labels now). Also, debugging builds now have a new assertion in `op_free()` to ensure that the op doesn't have any unrecognized flags set in `op_private`.
- The deprecated variable `PL_sv_objcount` has been removed.
- Perl now tries to keep the locale category `LC_NUMERIC` set to ''C'' except around operations that need it to be set to the program's underlying locale. This protects the many XS modules that cannot cope with the decimal radix character not being a dot. Prior to this release, Perl initialized this category to ''C'', but a call to `POSIX::setlocale()` would change it. Now such a call will change the underlying locale of the `LC_NUMERIC` category for the program, but the locale exposed to XS code will remain ''C''. There are new macros to manipulate the LC_NUMERIC locale, including `STORE_LC_NUMERIC_SET_TO_NEEDED` and `STORE_LC_NUMERIC_FORCE_TO_UNDERLYING`. See ''Locale-related functions and macros'' in perlapi.
- A new macro `isUTF8_CHAR` has been written which efficiently determines if the string given by its parameters begins with a well-formed UTF-8 encoded character.
- The following private API functions had their context parameter removed: `Perl_cast_ulong`, `Perl_cast_i32`, `Perl_cast_iv`, `Perl_cast_uv`, `Perl_cv_const_sv`, `Perl_mg_find`, `Perl_mg_findext`, `Perl_mg_magical`, `Perl_mini_mktime`, `Perl_my_dirfd`, `Perl_sv_backoff`, `Perl_utf8_hop`.

  Note that the prefix-less versions of those functions that are part of the public API, such as `cast_i32()`, remain unaffected.
- The `PADNAME` and `PADNAMELIST` types are now separate types, and no longer simply aliases for SV and AV. [GH #14250] <https://github.com/Perl/perl5/issues/14250>.

- Pad names are now always UTF-8. The `PadnameUTF8` macro always returns true. Previously, this was effectively the case already, but any support for two different internal representations of pad names has now been removed.
- A new op class, `UNOP_AUX`, has been added. This is a subclass of `UNOP` with an `op_aux` field added, which points to an array of unions of UV, SV* etc. It is intended for where an op needs to store more data than a simple `op_sv` or whatever. Currently the only op of this type is `OP_MULTIDEREF` (see next item).
- A new op has been added, `OP_MULTIDEREF`, which performs one or more nested array and hash lookups where the key is a constant or simple variable. For example the expression `$a[0]{$k}[$i]`, which previously involved ten `rv2Xv`, `Xelem`, `gvsv` and `const` ops is now performed by a single `multideref` op. It can also handle `local`, `exists` and `delete`. A non-simple index expression, such as `[$i+1]` is still done using `aelem`/`helem`, and single-level array lookup with a small constant index is still done using `aelemfast`.

## Selected Bug Fixes

- `close` now sets `$!`

  When an I/O error occurs, the fact that there has been an error is recorded in the handle. `close` returns false for such a handle. Previously, the value of `$!` would be untouched by `close`, so the common convention of writing `close $fh or die $!` did not work reliably. Now the handle records the value of `$!`, too, and `close` restores it.

- `no re` now can turn off everything that `use re` enables

  Previously, running `no re` would turn off only a few things. Now it can turn off all the enabled things. For example, the only way to stop debugging, once enabled, was to exit the enclosing block; that is now fixed.

- `pack("D", $x)` and `pack("F", $x)` now zero the padding on x86 long double builds. Under some build options on GCC 4.8 and later, they used to either overwrite the zero-initialized padding, or bypass the initialized buffer entirely. This caused *op/pack.t* to fail. [GH #14554] <https://github.com/Perl/perl5/issues/14554>
- Extending an array cloned from a parent thread could result in ''Modification of a read-only value attempted'' errors when attempting to modify the new elements. [GH #14605] <https://github.com/Perl/perl5/issues/14605>
- An assertion failure and subsequent crash with `*x=<y>` has been fixed. [GH #14493] <https://github.com/Perl/perl5/issues/14493>
- A possible crashing/looping bug related to compiling lexical subs has been fixed. [GH #14596] <https://github.com/Perl/perl5/issues/14596>
- UTF-8 now works correctly in function names, in unquoted HERE-document terminators, and in variable names used as array indexes. [GH #14601] <https://github.com/Perl/perl5/issues/14601>
- Repeated global pattern matches in scalar context on large tainted strings were exponentially slow depending on the current match position in the string. [GH #14238] <https://github.com/Perl/perl5/issues/14238>
- Various crashes due to the parser getting confused by syntax errors have been fixed. [GH #14496] <https://github.com/Perl/perl5/issues/14496> [GH #14497] <https://github.com/Perl/perl5/issues/14497> [GH #14548] <https://github.com/Perl/perl5/issues/14548> [GH #14564] <https://github.com/Perl/perl5/issues/14564>
- `split` in the scope of lexical `$_` has been fixed not to fail assertions. [GH #14483] <https://github.com/Perl/perl5/issues/14483>
- `my $x : attr` syntax inside various list operators no longer fails assertions. [GH #14500] <https://github.com/Perl/perl5/issues/14500>
- An `@` sign in quotes followed by a non-ASCII digit (which is not a valid identifier) would cause the parser to crash, instead of simply trying the `@` as literal. This has been fixed. [GH #14553] <https://github.com/Perl/perl5/issues/14553>

- `*bar::=*foo::=*glob_with_hash` has been crashing since Perl 5.14, but no longer does. [GH #14512] <https://github.com/Perl/perl5/issues/14512>
- `foreach` in scalar context was not pushing an item on to the stack, resulting in bugs. (`print 4, scalar do { foreach(@x){} } + 1` would print 5.) It has been fixed to return `undef`. [GH #14569] <https://github.com/Perl/perl5/issues/14569>
- Several cases of data used to store environment variable contents in core C code being potentially overwritten before being used have been fixed. [GH #14476] <https://github.com/Perl/perl5/issues/14476>
- Some patterns starting with `/.*..../` matched against long strings have been slow since v5.8, and some of the form `/.*..../i` have been slow since v5.18. They are now all fast again. [GH #14475] <https://github.com/Perl/perl5/issues/14475>.
- The original visible value of `$/` is now preserved when it is set to an invalid value. Previously if you set `$/` to a reference to an array, for example, perl would produce a runtime error and not set `PL_rs`, but Perl code that checked `$/` would see the array reference. [GH #14245] <https://github.com/Perl/perl5/issues/14245>.
- In a regular expression pattern, a POSIX class, like `[:ascii:]`, must be inside a bracketed character class, like `qr/[[:ascii:]]/`. A warning is issued when something looking like a POSIX class is not inside a bracketed class. That warning wasn't getting generated when the POSIX class was negated: `[:^ascii:]`. This is now fixed.
- Perl 5.14.0 introduced a bug whereby `eval { LABEL: }` would crash. This has been fixed. [GH #14438] <https://github.com/Perl/perl5/issues/14438>.
- Various crashes due to the parser getting confused by syntax errors have been fixed. [GH #14421] <https://github.com/Perl/perl5/issues/14421>. [GH #14472] <https://github.com/Perl/perl5/issues/14472>. [GH #14480] <https://github.com/Perl/perl5/issues/14480>. [GH #14447] <https://github.com/Perl/perl5/issues/14447>.
- Code like `/$a[/` used to read the next line of input and treat it as though it came immediately after the opening bracket. Some invalid code consequently would parse and run, but some code caused crashes, so this is now disallowed. [GH #14462] <https://github.com/Perl/perl5/issues/14462>.
- Fix argument underflow for `pack`. [GH #14525] <https://github.com/Perl/perl5/issues/14525>.
- Fix handling of non-strict `\x{}`. Now `\x{}` is equivalent to `\x{0}` instead of faulting.
- `stat -t` is now no longer treated as stackable, just like `-t stat`. [GH #14499] <https://github.com/Perl/perl5/issues/14499>.
- The following no longer causes a SEGV: `qr{x+(y(?0))*}`.
- Fixed infinite loop in parsing backrefs in regexp patterns.
- Several minor bug fixes in behavior of Infinity and NaN, including warnings when stringifying Infinity-like or NaN-like strings. For example, ''NaNcy'' doesn't numify to NaN anymore.
- A bug in regular expression patterns that could lead to segfaults and other crashes has been fixed. This occurred only in patterns compiled with `/i` while taking into account the current POSIX locale (which usually means they have to be compiled within the scope of `use locale`), and there must be a string of at least 128 consecutive bytes to match. [GH #14389] <https://github.com/Perl/perl5/issues/14389>.
- `s///g` now works on very long strings (where there are more than 2 billion iterations) instead of dying with 'Substitution loop'. [GH #11742] <https://github.com/Perl/perl5/issues/11742>. [GH #14190] <https://github.com/Perl/perl5/issues/14190>.
- `gmtime` no longer crashes with not-a-number values. [GH #14365] <https://github.com/Perl/perl5/issues/14365>.
- `\()` (a reference to an empty list), and `y///` with lexical `$_` in scope, could both do a bad write past the end of the stack. They have both been fixed to extend the stack first.

- `prototype()` with no arguments used to read the previous item on the stack, so `print "foo", prototype()` would print foo's prototype. It has been fixed to infer `$_` instead. [GH #14376] <https://github.com/Perl/perl5/issues/14376>.
- Some cases of lexical state subs declared inside predeclared subs could crash, for example when evalling a string including the name of an outer variable, but no longer do.
- Some cases of nested lexical state subs inside anonymous subs could cause 'Bizarre copy' errors or possibly even crashes.
- When trying to emit warnings, perl's default debugger (*perl5db.pl*) was sometimes giving 'Undefined subroutine &DB::db_warn called' instead. This bug, which started to occur in Perl 5.18, has been fixed. [GH #14400] <https://github.com/Perl/perl5/issues/14400>.
- Certain syntax errors in substitutions, such as `s/${<>{})//`, would crash, and had done so since Perl 5.10. (In some cases the crash did not start happening till 5.16.) The crash has, of course, been fixed. [GH #14391] <https://github.com/Perl/perl5/issues/14391>.
- Fix a couple of string grow size calculation overflows; in particular, a repeat expression like `33 x ~3` could cause a large buffer overflow since the new output buffer size was not correctly handled by `SvGROW()`. An expression like this now properly produces a memory wrap panic. [GH #14401] <https://github.com/Perl/perl5/issues/14401>.
- `formline("@...", "a");` would crash. The `FF_CHECKNL` case in `pp_formline()` didn't set the pointer used to mark the chop position, which led to the `FF_MORE` case crashing with a segmentation fault. This has been fixed. [GH #14388] <https://github.com/Perl/perl5/issues/14388>.
- A possible buffer overrun and crash when parsing a literal pattern during regular expression compilation has been fixed. [GH #14416] <https://github.com/Perl/perl5/issues/14416>.
- `fchmod()` and `futimes()` now set `$!` when they fail due to being passed a closed file handle. [GH #14073] <https://github.com/Perl/perl5/issues/14073>.
- `op_free()` and `scalarvoid()` no longer crash due to a stack overflow when freeing a deeply recursive op tree. [GH #11866] <https://github.com/Perl/perl5/issues/11866>.
- In Perl 5.20.0, `$^N` accidentally had the internal UTF-8 flag turned off if accessed from a code block within a regular expression, effectively UTF-8-encoding the value. This has been fixed. [GH #14211] <https://github.com/Perl/perl5/issues/14211>.
- A failed `semctl` call no longer overwrites existing items on the stack, which means that `(semctl(-1,0,0,0))[0]` no longer gives an ''uninitialized'' warning.
- `else{foo()}` with no space before `foo` is now better at assigning the right line number to that statement. [GH #14070] <https://github.com/Perl/perl5/issues/14070>.
- Sometimes the assignment in `@array = split` gets optimised so that `split` itself writes directly to the array. This caused a bug, preventing this assignment from being used in lvalue context. So `(@a=split//,"foo")=bar()` was an error. (This bug probably goes back to Perl 3, when the optimisation was added.) It has now been fixed. [GH #14183] <https://github.com/Perl/perl5/issues/14183>.
- When an argument list fails the checks specified by a subroutine signature (which is still an experimental feature), the resulting error messages now give the file and line number of the caller, not of the called subroutine. [GH #13643] <https://github.com/Perl/perl5/issues/13643>.
- The flip-flop operators (`..` and `...` in scalar context) used to maintain a separate state for each recursion level (the number of times the enclosing sub was called recursively), contrary to the documentation. Now each closure has one internal state for each flip-flop. [GH #14110] <https://github.com/Perl/perl5/issues/14110>.
- The flip-flop operator (`..` in scalar context) would return the same scalar each time, unless the containing subroutine was called recursively. Now it always returns a new scalar. [GH #14110] <https://github.com/Perl/perl5/issues/14110>.
- `use`, `no`, statement labels, special blocks (`BEGIN`) and pod are now permitted as the first thing in a `map` or `grep` block, the block after `print` or `say` (or other functions) returning a handle, and within `${...}`, `@{...}`, etc. [GH #14088] <https://github.com/Perl/perl5/issues/14088>.

- The repetition operator `x` now propagates lvalue context to its left-hand argument when used in contexts like `foreach`. That allows `for(($#that_array)x2) { ... }` to work as expected if the loop modifies `$_`.
- `(...) x ...` in scalar context used to corrupt the stack if one operand was an object with "x" overloading, causing erratic behavior. [GH #13811] <https://github.com/Perl/perl5/issues/13811>.
- Assignment to a lexical scalar is often optimised away; for example in `my $x; $x = $y + $z`, the assign operator is optimised away and the add operator writes its result directly to `$x`. Various bugs related to this optimisation have been fixed. Certain operators on the right-hand side would sometimes fail to assign the value at all or assign the wrong value, or would call STORE twice or not at all on tied variables. The operators affected were `$foo++`, `$foo--`, and `-$foo` under `use integer`, `chomp`, `chr` and `setpgrp`.
- List assignments were sometimes buggy if the same scalar ended up on both sides of the assignment due to use of `tied`, `values` or `each`. The result would be the wrong value getting assigned.
- `setpgrp($nonzero)` (with one argument) was accidentally changed in 5.16 to mean `setpgrp(0)`. This has been fixed.
- `__SUB__` could return the wrong value or even corrupt memory under the debugger (the `-d` switch) and in subs containing `eval $string`.
- When `sub () { $var }` becomes inlinable, it now returns a different scalar each time, just as a non-inlinable sub would, though Perl still optimises the copy away in cases where it would make no observable difference.
- `my sub f () { $var }` and `sub () : attr { $var }` are no longer eligible for inlining. The former would crash; the latter would just throw the attributes away. An exception is made for the little-known `:method` attribute, which does nothing much.
- Inlining of subs with an empty prototype is now more consistent than before. Previously, a sub with multiple statements, of which all but the last were optimised away, would be inlinable only if it were an anonymous sub containing a string `eval` or `state` declaration or closing over an outer lexical variable (or any anonymous sub under the debugger). Now any sub that gets folded to a single constant after statements have been optimised away is eligible for inlining. This applies to things like `sub () { jabber() if DEBUG; 42 }`.

  Some subroutines with an explicit `return` were being made inlinable, contrary to the documentation, Now `return` always prevents inlining.
- On some systems, such as VMS, `crypt` can return a non-ASCII string. If a scalar assigned to had contained a UTF-8 string previously, then `crypt` would not turn off the UTF-8 flag, thus corrupting the return value. This would happen with `$lexical = crypt ...`.
- `crypt` no longer calls `FETCH` twice on a tied first argument.
- An unterminated here-doc on the last line of a quote-like operator (`qq[${ <<END }]`, `/(?{ <<END })/`) no longer causes a double free. It started doing so in 5.18.
- `index()` and `rindex()` no longer crash when used on strings over 2GB in size. [GH #13700] <https://github.com/Perl/perl5/issues/13700>.
- A small, previously intentional, memory leak in `PERL_SYS_INIT`/`PERL_SYS_INIT3` on Win32 builds was fixed. This might affect embedders who repeatedly create and destroy perl engines within the same process.
- `POSIX::localeconv()` now returns the data for the program's underlying locale even when called from outside the scope of `use locale`.
- `POSIX::localeconv()` now works properly on platforms which don't have `LC_NUMERIC` and/or `LC_MONETARY`, or for which Perl has been compiled to disregard either or both of these locale categories. In such circumstances, there are now no entries for the corresponding values in the hash returned by `localeconv()`.
- `POSIX::localeconv()` now marks appropriately the values it returns as UTF-8 or not. Previously they were always returned as bytes, even if they were supposed to be encoded as UTF-8.

- On Microsoft Windows, within the scope of `use locale`, the following POSIX character classes gave results for many locales that did not conform to the POSIX standard: `[[:alnum:]]`, `[[:alpha:]]`, `[[:blank:]]`, `[[:digit:]]`, `[[:graph:]]`, `[[:lower:]]`, `[[:print:]]`, `[[:punct:]]`, `[[:upper:]]`, `[[:word:]]`, and `[[:xdigit:]]`. This was because the underlying Microsoft implementation does not follow the standard. Perl now takes special precautions to correct for this.
- Many issues have been detected by Coverity <http://www.coverity.com/> and fixed.
- `system()` and friends should now work properly on more Android builds.

  Due to an oversight, the value specified through `-Dtargetsh` to *Configure* would end up being ignored by some of the build process. This caused perls cross-compiled for Android to end up with defective versions of `system()`, `exec()` and backticks: the commands would end up looking for `/bin/sh` instead of `/system/bin/sh`, and so would fail for the vast majority of devices, leaving `$!` as `ENOENT`.
- `qr(...\(...\)...)`, `qr[...\[...\]...]`, and `qr{...\{...\}...}` now work. Previously it was impossible to escape these three left-characters with a backslash within a regular expression pattern where otherwise they would be considered metacharacters, and the pattern opening delimiter was the character, and the closing delimiter was its mirror character.
- `s///e` on tainted UTF-8 strings corrupted `pos()`. This bug, introduced in 5.20, is now fixed. [GH #13948] <https://github.com/Perl/perl5/issues/13948>.
- A non-word boundary in a regular expression (`\B`) did not always match the end of the string; in particular `q{} =~ /\B/` did not match. This bug, introduced in perl 5.14, is now fixed. [GH #13917] <https://github.com/Perl/perl5/issues/13917>.
- `" P" =~ /(?=.*P)P/` should match, but did not. This is now fixed. [GH #13954] <https://github.com/Perl/perl5/issues/13954>.
- Failing to compile `use Foo` in an `eval` could leave a spurious `BEGIN` subroutine definition, which would produce a "Subroutine BEGIN redefined" warning on the next use of `use`, or other `BEGIN` block. [GH #13926] <https://github.com/Perl/perl5/issues/13926>.
- `method { BLOCK } ARGS` syntax now correctly parses the arguments if they begin with an opening brace. [GH #9085] <https://github.com/Perl/perl5/issues/9085>.
- External libraries and Perl may have different ideas of what the locale is. This is problematic when parsing version strings if the locale's numeric separator has been changed. Version parsing has been patched to ensure it handles the locales correctly. [GH #13863] <https://github.com/Perl/perl5/issues/13863>.
- A bug has been fixed where zero-length assertions and code blocks inside of a regex could cause `pos` to see an incorrect value. [GH #14016] <https://github.com/Perl/perl5/issues/14016>.
- Dereferencing of constants now works correctly for typeglob constants. Previously the glob was stringified and its name looked up. Now the glob itself is used. [GH #9891] <https://github.com/Perl/perl5/issues/9891>
- When parsing a sigil (`$ @ % &`) followed by braces, the parser no longer tries to guess whether it is a block or a hash constructor (causing a syntax error when it guesses the latter), since it can only be a block.
- `undef $reference` now frees the referent immediately, instead of hanging on to it until the next statement. [GH #14032] <https://github.com/Perl/perl5/issues/14032>
- Various cases where the name of a sub is used (autoload, overloading, error messages) used to crash for lexical subs, but have been fixed.
- Bareword lookup now tries to avoid vivifying packages if it turns out the bareword is not going to be a subroutine name.
- Compilation of anonymous constants (*e.g.*, `sub () { 3 }`) no longer deletes any subroutine named `__ANON__` in the current package. Not only was `*__ANON__{CODE}` cleared, but there was a memory leak, too. This bug goes back to Perl 5.8.0.

- Stub declarations like `sub f;` and `sub f ();` no longer wipe out constants of the same name declared by `use constant`. This bug was introduced in Perl 5.10.0.
- `qr/[\N{named sequence}]/` now works properly in many instances.

  Some names known to `\N{...}` refer to a sequence of multiple characters, instead of the usual single character. Bracketed character classes generally only match single characters, but now special handling has been added so that they can match named sequences, but not if the class is inverted or the sequence is specified as the beginning or end of a range. In these cases, the only behavior change from before is a slight rewording of the fatal error message given when this class is part of a `?[...])` construct. When the `[...]` stands alone, the same non-fatal warning as before is raised, and only the first character in the sequence is used, again just as before.
- Tainted constants evaluated at compile time no longer cause unrelated statements to become tainted. [GH #14059] <https://github.com/Perl/perl5/issues/14059>
- `open $$fh, ...`, which vivifies a handle with a name like `"main::_GEN_0"`, was not giving the handle the right reference count, so a double free could happen.
- When deciding that a bareword was a method name, the parser would get confused if an `our` sub with the same name existed, and look up the method in the package of the `our` sub, instead of the package of the invocant.
- The parser no longer gets confused by `\U=` within a double-quoted string. It used to produce a syntax error, but now compiles it correctly. [GH #10882] <https://github.com/Perl/perl5/issues/10882>
- It has always been the intention for the `-B` and `-T` file test operators to treat UTF-8 encoded files as text. (perlfunc has been updated to say this.) Previously, it was possible for some files to be considered UTF-8 that actually weren't valid UTF-8. This is now fixed. The operators now work on EBCDIC platforms as well.
- Under some conditions warning messages raised during regular expression pattern compilation were being output more than once. This has now been fixed.
- Perl 5.20.0 introduced a regression in which a UTF-8 encoded regular expression pattern that contains a single ASCII lowercase letter did not match its uppercase counterpart. That has been fixed in both 5.20.1 and 5.22.0. [GH #14051] <https://github.com/Perl/perl5/issues/14051>
- Constant folding could incorrectly suppress warnings if lexical warnings (`use warnings` or `no warnings`) were not in effect and `$^W` were false at compile time and true at run time.
- Loading Unicode tables during a regular expression match could cause assertion failures under debugging builds if the previous match used the very same regular expression. [GH #14081] <https://github.com/Perl/perl5/issues/14081>
- Thread cloning used to work incorrectly for lexical subs, possibly causing crashes or double frees on exit.
- Since Perl 5.14.0, deleting `$SomePackage::{__ANON__}` and then undefining an anonymous subroutine could corrupt things internally, resulting in Devel::Peek crashing or B.pm giving nonsensical data. This has been fixed.
- `(caller $n)[3]` now reports names of lexical subs, instead of treating them as `"(unknown)"`.
- `sort subname LIST` now supports using a lexical sub as the comparison routine.
- Aliasing (*e.g.*, via `*x = *y`) could confuse list assignments that mention the two names for the same variable on either side, causing wrong values to be assigned. [GH #5788] <https://github.com/Perl/perl5/issues/5788>
- Long here-doc terminators could cause a bad read on short lines of input. This has been fixed. It is doubtful that any crash could have occurred. This bug goes back to when here-docs were introduced in Perl 3.000 twenty-five years ago.
- An optimization in `split` to treat `split /^/` like `split /^/m` had the unfortunate side-effect of also treating `split /\A/` like `split /^/m`, which it should not. This has been fixed. (Note, however, that `split /^x/` does not behave like `split /^x/m`, which is also

considered to be a bug and will be fixed in a future version.) [GH #14086] <https://github.com/Perl/perl5/issues/14086>

- The little-known `my Class $var` syntax (see fields and attributes) could get confused in the scope of `use utf8` if `Class` were a constant whose value contained Latin−1 characters.
- Locking and unlocking values via Hash::Util or `Internals::SvREADONLY` no longer has any effect on values that were read-only to begin with. Previously, unlocking such values could result in crashes, hangs or other erratic behavior.
- Some unterminated `(?(...)...)` constructs in regular expressions would either crash or give erroneous error messages. `/(?(1)/` is one such example.
- `pack "w", $tied` no longer calls FETCH twice.
- List assignments like `($x, $z) = (1, $y)` now work correctly if `$x` and `$y` have been aliased by `foreach`.
- Some patterns including code blocks with syntax errors, such as `/ (?{(^{})/`, would hang or fail assertions on debugging builds. Now they produce errors.
- An assertion failure when parsing `sort` with debugging enabled has been fixed. [GH #14087] <https://github.com/Perl/perl5/issues/14087>.
- `*a = *b; @a = split //, $b[1]` could do a bad read and produce junk results.
- In `() = @array = split`, the `() =` at the beginning no longer confuses the optimizer into assuming a limit of 1.
- Fatal warnings no longer prevent the output of syntax errors. [GH #14155] <https://github.com/Perl/perl5/issues/14155>.
- Fixed a NaN double-to-long-double conversion error on VMS. For quiet NaNs (and only on Itanium, not Alpha) negative infinity instead of NaN was produced.
- Fixed the issue that caused `make distclean` to incorrectly leave some files behind. [GH #14108] <https://github.com/Perl/perl5/issues/14108>.
- AIX now sets the length in `getsockopt` correctly. [GH #13484] <https://github.com/Perl/perl5/issues/13484>. [cpan #91183] <https://rt.cpan.org/Ticket/Display.html?id=91183>. [cpan #85570] <https://rt.cpan.org/Ticket/Display.html?id=85570>.
- The optimization phase of a regexp compilation could run ‘‘forever’’ and exhaust all memory under certain circumstances; now fixed. [GH #13984] <https://github.com/Perl/perl5/issues/13984>.
- The test script *t/op/crypt.t* now uses the SHA−256 algorithm if the default one is disabled, rather than giving failures. [GH #13715] <https://github.com/Perl/perl5/issues/13715>.
- Fixed an off-by-one error when setting the size of a shared array. [GH #14151] <https://github.com/Perl/perl5/issues/14151>.
- Fixed a bug that could cause perl to enter an infinite loop during compilation. In particular, a `while(1)` within a sublist, *e.g.*

  ```
  sub foo { () = ($a, my $b, ($c, do { while(1) {} })) }
  ```

  The bug was introduced in 5.20.0 [GH #14165] <https://github.com/Perl/perl5/issues/14165>.
- On Win32, if a variable was `local`−ized in a pseudo-process that later forked, restoring the original value in the child pseudo-process caused memory corruption and a crash in the child pseudo-process (and therefore the OS process). [GH #8641] <https://github.com/Perl/perl5/issues/8641>.
- Calling `write` on a format with a `^**` field could produce a panic in `sv_chop()` if there were insufficient arguments or if the variable used to fill the field was empty. [GH #14255] <https://github.com/Perl/perl5/issues/14255>.
- Non-ASCII lexical sub names now appear without trailing junk when they appear in error messages.

- The `\@` subroutine prototype no longer flattens parenthesized arrays (taking a reference to each element), but takes a reference to the array itself. [GH #9111] <https://github.com/Perl/perl5/issues/9111>.
- A block containing nothing except a C−style `for` loop could corrupt the stack, causing lists outside the block to lose elements or have elements overwritten. This could happen with `map { for(...){...} } ...` and with lists containing `do { for(...){...} }`. [GH #14269] <https://github.com/Perl/perl5/issues/14269>.
- `scalar()` now propagates lvalue context, so that `for(scalar($#foo)) { ... }` can modify `$#foo` through `$_`.
- `qr/@array(?{block})/` no longer dies with "Bizarre copy of ARRAY". [GH #14292] <https://github.com/Perl/perl5/issues/14292>.
- `eval '$variable'` in nested named subroutines would sometimes look up a global variable even with a lexical variable in scope.
- In perl 5.20.0, `sort CORE::fake` where 'fake' is anything other than a keyword, started chopping off the last 6 characters and treating the result as a sort sub name. The previous behavior of treating `CORE::fake` as a sort sub name has been restored. [GH #14323] <https://github.com/Perl/perl5/issues/14323>.
- Outside of `use utf8`, a single-character Latin−1 lexical variable is disallowed. The error message for it, "Can't use global `$foo`...", was giving garbage instead of the variable name.
- `readline` on a nonexistent handle was causing `${^LAST_FH}` to produce a reference to an undefined scalar (or fail an assertion). Now `${^LAST_FH}` ends up undefined.
- `(...) x ...` in void context now applies scalar context to the left-hand argument, instead of the context the current sub was called in. [GH #14174] <https://github.com/Perl/perl5/issues/14174>.

## Known Problems

- `pack`−ing a NaN on a perl compiled with Visual C 6 does not behave properly, leading to a test failure in *t/op/infnan.t*. [GH #14705] <https://github.com/Perl/perl5/issues/14705>
- A goal is for Perl to be able to be recompiled to work reasonably well on any Unicode version. In Perl 5.22, though, the earliest such version is Unicode 5.1 (current is 7.0).
- EBCDIC platforms
  - The `cmp` (and hence `sort`) operators do not necessarily give the correct results when both operands are UTF-EBCDIC encoded strings and there is a mixture of ASCII and/or control characters, along with other characters.
  - Ranges containing `\N{...}` in the `tr///` (and `y///`) transliteration operators are treated differently than the equivalent ranges in regular expression patterns. They should, but don't, cause the values in the ranges to all be treated as Unicode code points, and not native ones. ("Version 8 Regular Expressions" in perlre gives details as to how it should work.)
  - Encode and encoding are mostly broken.
  - Many CPAN modules that are shipped with core show failing tests.
  - `pack`/`unpack` with `"U0"` format may not work properly.
- The following modules are known to have test failures with this version of Perl. In many cases, patches have been submitted, so there will hopefully be new releases soon:
  - B::Generate version 1.50
  - B::Utils version 0.25
  - Coro version 6.42
  - Dancer version 1.3130
  - Data::Alias version 1.18

- Data::Dump::Streamer version 2.38
- Data::Util version 0.63
- Devel::Spy version 0.07
- invoker version 0.34
- Lexical::Var version 0.009
- LWP::ConsoleLogger version 0.000018
- Mason version 2.22
- NgxQueue version 0.02
- Padre version 1.00
- Parse::Keyword 0.08

## Obituary

Brian McCauley died on May 8, 2015. He was a frequent poster to Usenet, Perl Monks, and other Perl forums, and made several CPAN contributions under the nick NOBULL, including to the Perl FAQ. He attended almost every YAPC::Europe, and indeed, helped organise YAPC::Europe 2006 and the QA Hackathon 2009. His wit and his delight in intricate systems were particularly apparent in his love of board games; many Perl mongers will have fond memories of playing Fluxx and other games with Brian. He will be missed.

## Acknowledgements

Perl 5.22.0 represents approximately 12 months of development since Perl 5.20.0 and contains approximately 590,000 lines of changes across 2,400 files from 94 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 370,000 lines of changes to 1,500 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.22.0:

Aaron Crane, Abhijit Menon-Sen, Abigail, Alberto Simões, Alex Solovey, Alex Vandiver, Alexandr Ciornii, Alexandre (Midnite) Jousset, Andreas König, Andreas Voegele, Andrew Fresh, Andy Dougherty, Anthony Heading, Aristotle Pagaltzis, brian d foy, Brian Fraser, Chad Granum, Chris 'BinGOs' Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, Daniel Dragan, Darin McBride, Dave Rolsky, David Golden, David Mitchell, David Wheeler, Dmitri Tikhonov, Doug Bell, E. Choroba, Ed J, Eric Herman, Father Chrysostomos, George Greer, Glenn D. Golden, Graham Knop, H.Merijn Brand, Herbert Breunung, Hugo van der Sanden, James E Keenan, James McCoy, James Raspass, Jan Dubois, Jarkko Hietaniemi, Jasmine Ngan, Jerry D. Hedden, Jim Cromie, John Goodyear, kafka, Karen Etheridge, Karl Williamson, Kent Fredric, kmx, Lajos Veres, Leon Timmermans, Lukas Mai, Mathieu Arnold, Matthew Horsfall, Max Maischein, Michael Bunk, Nicholas Clark, Niels Thykier, Niko Tyni, Norman Koch, Olivier Mengué, Peter John Acklam, Peter Martini, Petr PísaX, Philippe Bruhat (BooK), Pierre Bogossian, Rafael Garcia-Suarez, Randy Stauner, Reini Urban, Ricardo Signes, Rob Hoelz, Rostislav Skudnov, Sawyer X, Shirakata Kentaro, Shlomi Fish, Sisyphus, Slaven Rezic, Smylers, Steffen Müller, Steve Hay, Sullivan Beck, syber, Tadeusz SoXnierz, Thomas Sibley, Todd Rinaldo, Tony Cook, Vincent Pit, Vladimir Marek, Yaroslav Kuzmin, Yves Orton, ÆEvar Arnfjörð Bjarmason.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at <https://rt.perl.org/>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release.

Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5221delta − what is new for perl v5.22.1

## DESCRIPTION

This document describes differences between the 5.22.0 release and the 5.22.1 release.

If you are upgrading from an earlier release such as 5.20.0, first read perl5220delta, which describes differences between 5.20.0 and 5.22.0.

## Incompatible Changes

There are no changes intentionally incompatible with 5.20.0 other than the following single exception, which we deemed to be a sensible change to make in order to get the new `\b{wb}` and (in particular) `\b{sb}` features sane before people decided they're worthless because of bugs in their Perl 5.22.0 implementation and avoided them in the future. If any others exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

### Bounds Checking Constructs

Several bugs, including a segmentation fault, have been fixed with the bounds checking constructs (introduced in Perl 5.22) `\b{gcb}`, `\b{sb}`, `\b{wb}`, `\B{gcb}`, `\B{sb}`, and `\B{wb}`. All the `\B{}` ones now match an empty string; none of the `\b{}` ones do. [GH #14976] <https://github.com/Perl/perl5/issues/14976>

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20150520 to 5.20151213.
- PerlIO::scalar has been upgraded from version 0.22 to 0.23.
- POSIX has been upgraded from version 1.53 to 1.53_01.

  If `POSIX::strerror` was passed `$!` as its argument then it accidentally cleared `$!`. This has been fixed. [GH #14951] <https://github.com/Perl/perl5/issues/14951>
- Storable has been upgraded from version 2.53 to 2.53_01.
- warnings has been upgraded from version 1.32 to 1.34.

  The `warnings::enabled` example now actually uses `warnings::enabled`. [GH #14905] <https://github.com/Perl/perl5/issues/14905>
- Win32 has been upgraded from version 0.51 to 0.52.

  This has been updated for Windows 8.1, 10 and 2012 R2 Server.

## Documentation

### Changes to Existing Documentation

*perltie*

- The usage of `FIRSTKEY` and `NEXTKEY` has been clarified.

*perlvar*

- The specific true value of `$!{E...}` is now documented, noting that it is subject to change and not guaranteed.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### Changes to Existing Diagnostics

- The `printf` and `sprintf` builtins are now more careful about the warnings they emit: argument reordering now disables the ''redundant argument'' warning in all cases. [GH #14772] <https://github.com/Perl/perl5/issues/14772>

## Configuration and Compilation

- Using the `NO_HASH_SEED` define in combination with the default hash algorithm `PERL_HASH_FUNC_ONE_AT_A_TIME_HARD` resulted in a fatal error while compiling the interpreter, since Perl 5.17.10. This has been fixed.

- Configuring with ccflags containing quotes (e.g. `-Accflags='-DAPPLLIB_EXP=\"/usr/libperl\"'`) was broken in Perl 5.22.0 but has now been fixed again. [GH #14732] <https://github.com/Perl/perl5/issues/14732>

## Platform Support

### Platform-Specific Notes

IRIX

- Under some circumstances IRIX stdio **fgetc()** and **fread()** set the errno to `ENOENT`, which made no sense according to either IRIX or POSIX docs. Errno is now cleared in such cases. [GH #14557] <https://github.com/Perl/perl5/issues/14557>
- Problems when multiplying long doubles by infinity have been fixed. [GH #14993] <https://github.com/Perl/perl5/issues/14993>
- All tests pass now on IRIX with the default build configuration.

## Selected Bug Fixes

- `qr/(?[ () ])/` no longer segfaults, giving a syntax error message instead. [GH #14851] <https://github.com/Perl/perl5/issues/14851>
- Regular expression possessive quantifier Perl 5.20 regression now fixed. `qr/`*PAT*`{`*min*`,`*max*`}+/` is supposed to behave identically to `qr/(?>`*PAT*`{`*min*`,`*max*`})/`. Since Perl 5.20, this didn't work if *min* and *max* were equal. [GH #14857] <https://github.com/Perl/perl5/issues/14857>
- Certain syntax errors in ''Extended Bracketed Character Classes'' in perlrecharclass caused panics instead of the proper error message. This has now been fixed. [GH #15016] <https://github.com/Perl/perl5/issues/15016>
- `BEGIN <>` no longer segfaults and properly produces an error message. [GH #13546] <https://github.com/Perl/perl5/issues/13546>
- A regression from Perl 5.20 has been fixed, in which some syntax errors in `(?[...])` constructs within regular expression patterns could cause a segfault instead of a proper error message. [GH #14933] <https://github.com/Perl/perl5/issues/14933>
- Another problem with `(?[...])` constructs has been fixed wherein things like `\c]` could cause panics. [GH #14934] <https://github.com/Perl/perl5/issues/14934>
- In Perl 5.22.0, the logic changed when parsing a numeric parameter to the -C option, such that the successfully parsed number was not saved as the option value if it parsed to the end of the argument. [GH #14748] <https://github.com/Perl/perl5/issues/14748>
- Warning fatality is now ignored when rewinding the stack. This prevents infinite recursion when the now fatal error also causes rewinding of the stack. [GH #14319] <https://github.com/Perl/perl5/issues/14319>
- A crash with `%::=(); J->${\"::"}` has been fixed. [GH #14790] <https://github.com/Perl/perl5/issues/14790>
- Nested quantifiers such as `/.{1}??/` should cause perl to throw a fatal error, but were being silently accepted since Perl 5.20.0. This has been fixed. [GH #14960] <https://github.com/Perl/perl5/issues/14960>
- Regular expression sequences such as `/(?i/` (and similarly with other recognized flags or combination of flags) should cause perl to throw a fatal error, but were being silently accepted since Perl 5.18.0. This has been fixed. [GH #14931] <https://github.com/Perl/perl5/issues/14931>
- A bug in hexadecimal floating point literal support meant that high-order bits could be lost in cases where mantissa overflow was caused by too many trailing zeros in the fractional part. This has been fixed. [GH #15032] <https://github.com/Perl/perl5/issues/15032>
- Another hexadecimal floating point bug, causing low-order bits to be lost in cases where the last hexadecimal digit of the mantissa has bits straddling the limit of the number of bits allowed for the mantissa, has also been fixed. [GH #15033] <https://github.com/Perl/perl5/issues/15033>
- Further hexadecimal floating point bugs have been fixed: In some circumstances, the `%a` format specifier could variously lose the sign of the negative zero, fail to display zeros after the radix point with the requested precision, or even lose the radix point after the leftmost hexadecimal digit completely.

- A crash caused by incomplete expressions within `/(?[ ])/` (e.g. `/(?[[0]+()+])/`) has been fixed. [GH #15045] <https://github.com/Perl/perl5/issues/15045>

## Acknowledgements

Perl 5.22.1 represents approximately 6 months of development since Perl 5.22.0 and contains approximately 19,000 lines of changes across 130 files from 27 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,700 lines of changes to 44 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.22.1:

Aaron Crane, Abigail, Andy Broad, Aristotle Pagaltzis, Chase Whitener, Chris ’BinGOs’ Williams, Craig A. Berry, Daniel Dragan, David Mitchell, Father Chrysostomos, Herbert Breunung, Hugo van der Sanden, James E Keenan, Jan Dubois, Jarkko Hietaniemi, Karen Etheridge, Karl Williamson, Lukas Mai, Matthew Horsfall, Peter Martini, Rafael Garcia-Suarez, Ricardo Signes, Shlomi Fish, Sisyphus, Steve Hay, Tony Cook, Victor Adam.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5222delta − what is new for perl v5.22.2

## DESCRIPTION

This document describes differences between the 5.22.1 release and the 5.22.2 release.

If you are upgrading from an earlier release such as 5.22.0, first read perl5221delta, which describes differences between 5.22.0 and 5.22.1.

## Security

### Fix out of boundary access in Win32 path handling

This is CVE−2015−8608. For more information see [GH #15067] <https://github.com/Perl/perl5/issues/15067>.

### Fix loss of taint in `canonpath()`

This is CVE−2015−8607. For more information see [GH #15084] <https://github.com/Perl/perl5/issues/15084>.

### Set proper umask before calling `mkstemp(3)`

In 5.22.0 perl started setting umask to `0600` before calling `mkstemp(3)` and restoring it afterwards. This wrongfully tells `open(2)` to strip the owner read and write bits from the given mode before applying it, rather than the intended negation of leaving only those bits in place.

Systems that use mode `0666` in `mkstemp(3)` (like old versions of glibc) create a file with permissions `0066`, leaving world read and write permissions regardless of current umask.

This has been fixed by using umask `0177` instead.

[GH #15135] <https://github.com/Perl/perl5/issues/15135>

### Avoid accessing uninitialized memory in Win32 `crypt()`

Validation that will detect both a short salt and invalid characters in the salt has been added.

[GH #15091] <https://github.com/Perl/perl5/issues/15091>

### Remove duplicate environment variables from `environ`

Previously, if an environment variable appeared more than once in `environ[]`, `%ENV` would contain the last entry for that name, while a typical `getenv()` would return the first entry. We now make sure `%ENV` contains the same as what `getenv()` returns.

Secondly, we now remove duplicates from `environ[]`, so if a setting with that name is set in `%ENV` we won't pass an unsafe value to a child process.

This is CVE−2016−2381.

## Incompatible Changes

There are no changes intentionally incompatible with Perl 5.22.1. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- File::Spec has been upgraded from version 3.56 to 3.56_01.

  `canonpath()` now preserves taint. See "Fix loss of taint in `canonpath()`".

- Module::CoreList has been upgraded from version 5.20151213 to 5.20160429.

  The version number of Digest::SHA listed for Perl 5.18.4 was wrong and has been corrected. Likewise for the version number of Config in 5.18.3 and 5.18.4. [GH #15202] <https://github.com/Perl/perl5/issues/15202>

## Documentation

### Changes to Existing Documentation

*perldiag*

- The explanation of the warning ‘‘unable to close filehandle `%s` properly: `%s`’’ which can occur when doing an implicit close of a filehandle has been expanded and improved.

*perlfunc*

- The documentation of `hex()` has been revised to clarify valid inputs.

## Configuration and Compilation

- Dtrace builds now build successfully on systems with a newer dtrace that require an input object file that uses the probes in the *.d* file.

  Previously the probe would fail and cause a build failure.

  [GH #13985] <https://github.com/Perl/perl5/issues/13985>

- *Configure* no longer probes for *libnm* by default. Originally this was the ''New Math'' library, but the name has been re-used by the GNOME NetworkManager.

  [GH #15115] <https://github.com/Perl/perl5/issues/15115>

- *Configure* now knows about gcc 5.

- Compiling perl with **-DPERL_MEM_LOG** now works again.

## Platform Support

### Platform-Specific Notes

Darwin

Compiling perl with **-Dusecbacktrace** on Darwin now works again.

[GH #15245] <https://github.com/Perl/perl5/issues/15245>

OS X/Darwin

Builds with both **-DDEBUGGING** and threading enabled would fail with a ''panic: free from wrong pool'' error when built or tested from Terminal on OS X. This was caused by perl's internal management of the environment conflicting with an atfork handler using the libc `setenv()` function to update the environment.

Perl now uses `setenv()`/`unsetenv()` to update the environment on OS X.

[GH #14955] <https://github.com/Perl/perl5/issues/14955>

ppc64el

The floating point format of ppc64el (Debian naming for little-endian PowerPC) is now detected correctly.

Tru64

A test failure in *t/porting/extrefs.t* has been fixed.

## Internal Changes

- An unwarranted assertion in `Perl_newATTRSUB_x()` has been removed. If a stub subroutine definition with a prototype has been seen, then any subsequent stub (or definition) of the same subroutine with an attribute was causing an assertion failure because of a null pointer.

  [GH #15081] <https://github.com/Perl/perl5/issues/15081>

## Selected Bug Fixes

- Calls to the placeholder `&PL_sv_yes` used internally when an `import()` or `unimport()` method isn't found now correctly handle scalar context. [GH #14902] <https://github.com/Perl/perl5/issues/14902>

- The `pipe()` operator would assert for `DEBUGGING` builds instead of producing the correct error message. The condition asserted on is detected and reported on correctly without the assertions, so the assertions were removed. [GH #15015] <https://github.com/Perl/perl5/issues/15015>

- In some cases, failing to parse a here-doc would attempt to use freed memory. This was caused by a pointer not being restored correctly. [GH #15009] <https://github.com/Perl/perl5/issues/15009>

- Perl now reports more context when it sees an array where it expects to see an operator, and avoids an assertion failure. [GH #14472] <https://github.com/Perl/perl5/issues/14472>

- If a here-doc was found while parsing another operator, the parser had already read end of file, and the here-doc was not terminated, perl could produce an assertion or a segmentation fault. This now reliably complains about the unterminated here-doc. [GH #14789] <https://github.com/Perl/perl5/issues/14789>

- Parsing beyond the end of the buffer when processing a `#line` directive with no filename is now avoided. [GH #15139] <https://github.com/Perl/perl5/issues/15139>
- Perl 5.22.0 added support for the C99 hexadecimal floating point notation, but sometimes misparsed hex floats. This has been fixed. [GH #15120] <https://github.com/Perl/perl5/issues/15120>
- Certain regex patterns involving a complemented posix class in an inverted bracketed character class, and matching something else optionally would improperly fail to match. An example of one that could fail is `qr/_?[^\Wbar]\x{100}/`. This has been fixed. [GH #15181] <https://github.com/Perl/perl5/issues/15181>
- Fixed an issue with `pack()` where `pack "H"` (and `pack "h"`) could read past the source when given a non-utf8 source and a utf8 target. [GH #14977] <https://github.com/Perl/perl5/issues/14977>
- Fixed some cases where perl would abort due to a segmentation fault, or a C-level assert. [GH #14941] <https://github.com/Perl/perl5/issues/14941> [GH #14962] <https://github.com/Perl/perl5/issues/14962> [GH #14963] <https://github.com/Perl/perl5/issues/14963> [GH #14997] <https://github.com/Perl/perl5/issues/14997> [GH #15039] <https://github.com/Perl/perl5/issues/15039> [GH #15247] <https://github.com/Perl/perl5/issues/15247> [GH #15251] <https://github.com/Perl/perl5/issues/15251>
- A memory leak when setting `$ENV{foo}` on Darwin has been fixed. [GH #14955] <https://github.com/Perl/perl5/issues/14955>
- Perl now correctly raises an error when trying to compile patterns with unterminated character classes while there are trailing backslashes. [GH #14919] <https://github.com/Perl/perl5/issues/14919>
- `NOTHING` regops and `EXACTFU_SS` regops in `make_trie()` are now handled properly. [GH #14945] <https://github.com/Perl/perl5/issues/14945>
- Perl now only tests `semctl()` if we have everything needed to use it. In FreeBSD the `semctl()` entry point may exist, but it can be disabled by policy. [GH #15180] <https://github.com/Perl/perl5/issues/15180>
- A regression that allowed undeclared barewords as hash keys to work despite strictures has been fixed. [GH #15099] <https://github.com/Perl/perl5/issues/15099>
- As an optimization (introduced in Perl 5.20.0), `uc()`, `lc()`, `ucfirst()` and `lcfirst()` sometimes modify their argument in-place rather than returning a modified copy. The criteria for this optimization has been made stricter to avoid these functions accidentally modifying in-place when they should not, which has been happening in some cases, e.g. in List::Util.
- Excessive memory usage in the compilation of some regular expressions involving non-ASCII characters has been reduced. A more complete fix is forthcoming in Perl 5.24.0.

## Acknowledgements

Perl 5.22.2 represents approximately 5 months of development since Perl 5.22.1 and contains approximately 3,000 lines of changes across 110 files from 24 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,500 lines of changes to 52 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.22.2:

Aaron Crane, Abigail, Andreas König, Aristotle Pagaltzis, Chris 'BinGOs' Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, David Golden, David Mitchell, H.Merijn Brand, James E Keenan, Jarkko Hietaniemi, Karen Etheridge, Karl Williamson, Matthew Horsfall, Niko Tyni, Ricardo Signes, Sawyer X, Stevan Little, Steve Hay, Todd Rinaldo, Tony Cook, Vladimir Timofeev, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5223delta − what is new for perl v5.22.3

## DESCRIPTION

This document describes differences between the 5.22.2 release and the 5.22.3 release.

If you are upgrading from an earlier release such as 5.22.1, first read perl5222delta, which describes differences between 5.22.1 and 5.22.2.

## Security

### −Di switch is now required for PerlIO debugging output

Previously PerlIO debugging output would be sent to the file specified by the `PERLIO_DEBUG` environment variable if perl wasn't running setuid and the **−T** or **−t** switches hadn't been parsed yet.

If perl performed output at a point where it hadn't yet parsed its switches this could result in perl creating or overwriting the file named by `PERLIO_DEBUG` even when the **−T** switch had been supplied.

Perl now requires the **−Di** switch to produce PerlIO debugging output. By default this is written to `stderr`, but can optionally be redirected to a file by setting the `PERLIO_DEBUG` environment variable.

If perl is running setuid or the **−T** switch was supplied `PERLIO_DEBUG` is ignored and the debugging output is sent to `stderr` as for any other **−D** switch.

### Core modules and tools no longer search "." for optional modules

The tools and many modules supplied in core no longer search the default current directory entry in `@INC` for optional modules. For example, Storable will remove the final "." from `@INC` before trying to load Log::Agent.

This prevents an attacker injecting an optional module into a process run by another user where the current directory is writable by the attacker, e.g. the */tmp* directory.

In most cases this removal should not cause problems, but difficulties were encountered with base, which treats every module name supplied as optional. These difficulties have not yet been resolved, so for this release there are no changes to base. We hope to have a fix for base in Perl 5.22.4.

To protect your own code from this attack, either remove the default "." entry from `@INC` at the start of your script, so:

```
#!/usr/bin/perl
use strict;
...
```

becomes:

```
#!/usr/bin/perl
BEGIN { pop @INC if $INC[-1] eq '.' }
use strict;
...
```

or for modules, remove "." from a localized `@INC`, so:

```
my $can_foo = eval { require Foo; }
```

becomes:

```
my $can_foo = eval {
    local @INC = @INC;
    pop @INC if $INC[-1] eq '.';
    require Foo;
};
```

## Incompatible Changes

Other than the security changes above there are no changes intentionally incompatible with Perl 5.22.2. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.04 to 2.04_01.
- bignum has been upgraded from version 0.39 to 0.39_01.
- CPAN has been upgraded from version 2.11 to 2.11_01.
- Digest has been upgraded from version 1.17 to 1.17_01.
- Digest::SHA has been upgraded from version 5.95 to 5.95_01.
- Encode has been upgraded from version 2.72 to 2.72_01.
- ExtUtils::Command has been upgraded from version 1.20 to 1.20_01.
- ExtUtils::MakeMaker has been upgraded from version 7.04_01 to 7.04_02.
- File::Fetch has been upgraded from version 0.48 to 0.48_01.
- File::Spec has been upgraded from version 3.56_01 to 3.56_02.
- HTTP::Tiny has been upgraded from version 0.054 to 0.054_01.
- IO has been upgraded from version 1.35 to 1.35_01.
- The IO-Compress modules have been upgraded from version 2.068 to 2.068_001.
- IPC::Cmd has been upgraded from version 0.92 to 0.92_01.
- JSON::PP has been upgraded from version 2.27300 to 2.27300_01.
- Locale::Maketext has been upgraded from version 1.26 to 1.26_01.
- Locale::Maketext::Simple has been upgraded from version 0.21 to 0.21_01.
- Memoize has been upgraded from version 1.03 to 1.03_01.
- Module::CoreList has been upgraded from version 5.20160429 to 5.20170114_22.
- Net::Ping has been upgraded from version 2.43 to 2.43_01.
- Parse::CPAN::Meta has been upgraded from version 1.4414 to 1.4414_001.
- Pod::Html has been upgraded from version 1.22 to 1.2201.
- Pod::Perldoc has been upgraded from version 3.25 to 3.25_01.
- Storable has been upgraded from version 2.53_01 to 2.53_02.
- Sys::Syslog has been upgraded from version 0.33 to 0.33_01.
- Test has been upgraded from version 1.26 to 1.26_01.
- Test::Harness has been upgraded from version 3.35 to 3.35_01.
- XSLoader has been upgraded from version 0.20 to 0.20_01, fixing a security hole in which binary files could be loaded from a path outside of `@INC`. [GH #15418] <https://github.com/Perl/perl5/issues/15418>

## Documentation

### Changes to Existing Documentation

*perlapio*

- The documentation of `PERLIO_DEBUG` has been updated.

*perlrun*

- The new **-Di** switch has been documented, and the documentation of `PERLIO_DEBUG` has been updated.

## Testing

- A new test script, *t/run/switchDx.t*, has been added to test that the new **-Di** switch is working correctly.

## Selected Bug Fixes

- The `PadlistNAMES` macro is an lvalue again.

## Acknowledgements

Perl 5.22.3 represents approximately 9 months of development since Perl 5.22.2 and contains approximately 4,400 lines of changes across 240 files from 20 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 2,200 lines of changes to 170 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.22.3:

Aaron Crane, Abigail, Alex Vandiver, Aristotle Pagaltzis, Chad Granum, Chris 'BinGOs' Williams, Craig A. Berry, David Mitchell, Father Chrysostomos, James E Keenan, Jarkko Hietaniemi, Karen Etheridge, Karl Williamson, Matthew Horsfall, Niko Tyni, Ricardo Signes, Sawyer X, Stevan Little, Steve Hay, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the Perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5224delta − what is new for perl v5.22.4

## DESCRIPTION

This document describes differences between the 5.22.3 release and the 5.22.4 release.

If you are upgrading from an earlier release such as 5.22.2, first read perl5223delta, which describes differences between 5.22.2 and 5.22.3.

## Security

### Improved handling of '.' in `@INC` in base.pm

The handling of (the removal of) `'.'` in `@INC` in base has been improved. This resolves some problematic behaviour in the approach taken in Perl 5.22.3, which is probably best described in the following two threads on the Perl 5 Porters mailing list: <http://www.nntp.perl.org/group/perl.perl5.porters/2016/08/msg238991.html>, <http://www.nntp.perl.org/group/perl.perl5.porters/2016/10/msg240297.html>.

### "Escaped" colons and relative paths in PATH

On Unix systems, Perl treats any relative paths in the PATH environment variable as tainted when starting a new process. Previously, it was allowing a backslash to escape a colon (unlike the OS), consequently allowing relative paths to be considered safe if the PATH was set to something like `/\:.`. The check has been fixed to treat `.` as tainted in that example.

## Modules and Pragmata

### Updated Modules and Pragmata

- base has been upgraded from version 2.22 to 2.22_01.
- Module::CoreList has been upgraded from version 5.20170114_22 to 5.20170715_22.

## Selected Bug Fixes

- Fixed a crash with `s///l` where it thought it was dealing with UTF−8 when it wasn't. [GH #15543] <https://github.com/Perl/perl5/issues/15543>

## Acknowledgements

Perl 5.22.4 represents approximately 6 months of development since Perl 5.22.3 and contains approximately 2,200 lines of changes across 52 files from 16 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 970 lines of changes to 18 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.22.4:

Aaron Crane, Abigail, Aristotle Pagaltzis, Chris 'BinGOs' Williams, David Mitchell, Eric Herman, Father Chrysostomos, James E Keenan, Karl Williamson, Lukas Mai, Renee Baecker, Ricardo Signes, Sawyer X, Stevan Little, Steve Hay, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help

assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5240delta − what is new for perl v5.24.0

## DESCRIPTION

This document describes the differences between the 5.22.0 release and the 5.24.0 release.

## Core Enhancements

### Postfix dereferencing is no longer experimental

Using the `postderef` and `postderef_qq` features no longer emits a warning. Existing code that disables the `experimental::postderef` warning category that they previously used will continue to work. The `postderef` feature has no effect; all Perl code can use postfix dereferencing, regardless of what feature declarations are in scope. The `5.24` feature bundle now includes the `postderef_qq` feature.

### Unicode 8.0 is now supported

For details on what is in this release, see <http://www.unicode.org/versions/Unicode8.0.0/>.

### perl will now croak when closing an in-place output file fails

Until now, failure to close the output file for an in-place edit was not detected, meaning that the input file could be clobbered without the edit being successfully completed. Now, when the output file cannot be closed successfully, an exception is raised.

### New `\b{lb}` boundary in regular expressions

`lb` stands for Line Break. It is a Unicode property that determines where a line of text is suitable to break (typically so that it can be output without overflowing the available horizontal space). This capability has long been furnished by the Unicode::LineBreak module, but now a light-weight, non-customizable version that is suitable for many purposes is in core Perl.

### `qr/(?[ ])/` now works in UTF−8 locales

Extended Bracketed Character Classes now will successfully compile when `use locale` is in effect. The compiled pattern will use standard Unicode rules. If the runtime locale is not a UTF−8 one, a warning is raised and standard Unicode rules are used anyway. No tainting is done since the outcome does not actually depend on the locale.

### Integer shift (<< and >>) now more explicitly defined

Negative shifts are reverse shifts: left shift becomes right shift, and right shift becomes left shift.

Shifting by the number of bits in a native integer (or more) is zero, except when the ‘‘overshift’’ is right shifting a negative value under `use integer`, in which case the result is −1 (arithmetic shift).

Until now negative shifting and overshifting have been undefined because they have relied on whatever the C implementation happens to do. For example, for the overshift a common C behavior is ‘‘modulo shift’’:

```
1 >> 64 == 1 >> (64 % 64) == 1 >> 0 == 1  # Common C behavior.

# And the same for <<, while Perl now produces 0 for both.
```

Now these behaviors are well-defined under Perl, regardless of what the underlying C implementation does. Note, however, that you are still constrained by the native integer width: you need to know how far left you can go. You can use for example:

```
use Config;
my $wordbits = $Config{uvsize} * 8;  # Or $Config{uvsize} << 3.
```

If you need a more bits on the left shift, you can use for example the `bigint` pragma, or the `Bit::Vector` module from CPAN.

### printf and sprintf now allow reordered precision arguments

That is, `sprintf '|%.*2$d|', 2, 3` now returns `|002|`. This extends the existing reordering mechanism (which allows reordering for arguments that are used as format fields, widths, and vector separators).

### More fields provided to `sigaction` callback with `SA_SIGINFO`

When passing the `SA_SIGINFO` flag to sigaction, the `errno`, `status`, `uid`, `pid`, `addr` and `band` fields are now included in the hash passed to the handler, if supported by the platform.

### Hashbang redirection to Perl 6

Previously perl would redirect to another interpreter if it found a hashbang path unless the path contains ‘‘perl’’ (see perlrun). To improve compatibility with Perl 6 this behavior has been extended to also redirect if ‘‘perl’’ is followed by ‘‘6’’.

## Security

### Set proper umask before calling `mkstemp(3)`

In 5.22 perl started setting umask to 0600 before calling `mkstemp(3)` and restoring it afterwards. This wrongfully tells `open(2)` to strip the owner read and write bits from the given mode before applying it, rather than the intended negation of leaving only those bits in place.

Systems that use mode 0666 in `mkstemp(3)` (like old versions of glibc) create a file with permissions 0066, leaving world read and write permissions regardless of current umask.

This has been fixed by using umask 0177 instead. [perl #127322]

### Fix out of boundary access in Win32 path handling

This is CVE-2015-8608. For more information see [GH #15067] <https://github.com/Perl/perl5/issues/15067>

### Fix loss of taint in canonpath

This is CVE-2015-8607. For more information see [GH #15084] <https://github.com/Perl/perl5/issues/15084>

### Avoid accessing uninitialized memory in win32 `crypt()`

Added validation that will detect both a short salt and invalid characters in the salt. [GH #15091] <https://github.com/Perl/perl5/issues/15091>

### Remove duplicate environment variables from `environ`

Previously, if an environment variable appeared more than once in `environ[]`, `%ENV` would contain the last entry for that name, while a typical `getenv()` would return the first entry. We now make sure `%ENV` contains the same as what `getenv` returns.

Second, we remove duplicates from `environ[]`, so if a setting with that name is set in `%ENV`, we won’t pass an unsafe value to a child process.

[CVE-2016-2381]

## Incompatible Changes

### The `autoderef` feature has been removed

The experimental `autoderef` feature (which allowed calling `push`, `pop`, `shift`, `unshift`, `splice`, `keys`, `values`, and `each` on a scalar argument) has been deemed unsuccessful. It has now been removed; trying to use the feature (or to disable the `experimental::autoderef` warning it previously triggered) now yields an exception.

### Lexical `$_` has been removed

`my $_` was introduced in Perl 5.10, and subsequently caused much confusion with no obvious solution. In Perl 5.18.0, it was made experimental on the theory that it would either be removed or redesigned in a less confusing (but backward-incompatible) way. Over the following years, no alternatives were proposed. The feature has now been removed and will fail to compile.

### `qr/\b{wb}/` is now tailored to Perl expectations

This is now more suited to be a drop-in replacement for plain `\b`, but giving better results for parsing natural language. Previously it strictly followed the current Unicode rules which calls for it to match between each white space character. Now it doesn’t generally match within spans of white space, behaving like `\b` does. See ‘‘\b{wb}’’ in perlrebackslash

### Regular expression compilation errors

Some regular expression patterns that had runtime errors now don’t compile at all.

Almost all Unicode properties using the `\p{}` and `\P{}` regular expression pattern constructs are now checked for validity at pattern compilation time, and invalid ones will cause the program to not compile. In earlier releases, this check was often deferred until run time. Whenever an error check is moved from run- to compile time, erroneous code is caught 100% of the time, whereas before it would only get caught if and when the offending portion actually gets executed, which for unreachable code might be never.

#### `qr/\N{}/` now disallowed under `use re ''strict''`

An empty `\N{}` makes no sense, but for backwards compatibility is accepted as doing nothing, though a deprecation warning is raised by default. But now this is a fatal error under the experimental feature "'strict' mode" in re.

#### Nested declarations are now disallowed

A `my`, `our`, or `state` declaration is no longer allowed inside of another `my`, `our`, or `state` declaration.

For example, these are now fatal:

```
my ($x, my($y));
our (my $x);
```

[GH #14799] <https://github.com/Perl/perl5/issues/14799>

[GH #13548] <https://github.com/Perl/perl5/issues/13548>

#### The `/\C/` character class has been removed.

This regular expression character class was deprecated in v5.20.0 and has produced a deprecation warning since v5.22.0. It is now a compile-time error. If you need to examine the individual bytes that make up a UTF8-encoded character, then use `utf8::encode()` on the string (or a copy) first.

#### `chdir('')` no longer chdirs home

Using `chdir('')` or `chdir(undef)` to chdir home has been deprecated since perl v5.8, and will now fail. Use `chdir()` instead.

#### ASCII characters in variable names must now be all visible

It was legal until now on ASCII platforms for variable names to contain non-graphical ASCII control characters (ordinals 0 through 31, and 127, which are the C0 controls and `DELETE`). This usage has been deprecated since v5.20, and as of now causes a syntax error. The variables these names referred to are special, reserved by Perl for whatever use it may choose, now, or in the future. Each such variable has an alternative way of spelling it. Instead of the single non-graphic control character, a two character sequence beginning with a caret is used, like `$^]` and `${^GLOBAL_PHASE}`. Details are at perlvar. It remains legal, though unwise and deprecated (raising a deprecation warning), to use certain non-graphic non-ASCII characters in variables names when not under `use utf8`. No code should do this, as all such variables are reserved by Perl, and Perl doesn't currently define any of them (but could at any time, without notice).

#### An off by one issue in `$Carp::MaxArgNums` has been fixed

`$Carp::MaxArgNums` is supposed to be the number of arguments to display. Prior to this version, it was instead showing `$Carp::MaxArgNums` + 1 arguments, contrary to the documentation.

#### Only blanks and tabs are now allowed within `[...]` within `(?[...])`.

The experimental Extended Bracketed Character Classes can contain regular bracketed character classes within them. These differ from regular ones in that white space is generally ignored, unless escaped by preceding it with a backslash. The white space that is ignored is now limited to just tab `\t` and SPACE characters. Previously, it was any white space. See "Extended Bracketed Character Classes" in perlrecharclass.

### Deprecations

#### Using code points above the platform's `IV_MAX` is now deprecated

Unicode defines code points in the range `0..0x10FFFF`. Some standards at one time defined them up to 2**31 - 1, but Perl has allowed them to be as high as anything that will fit in a word on the platform being used. However, use of those above the platform's `IV_MAX` is broken in some constructs, notably `tr///`, regular expression patterns involving quantifiers, and in some arithmetic and comparison operations, such as being the upper limit of a loop. Now the use of such code points raises a deprecation warning, unless that warning category is turned off. `IV_MAX` is typically 2**31 -1 on 32-bit platforms, and 2**63-1 on 64-bit ones.

#### Doing bitwise operations on strings containing code points above 0xFF is deprecated

The string bitwise operators treat their operands as strings of bytes, and values beyond 0xFF are nonsensical in this context. To operate on encoded bytes, first encode the strings. To operate on code points' numeric values, use `split` and `map ord`. In the future, this warning will be replaced by an exception.

### `sysread()`, `syswrite()`, `recv()` and `send()` are deprecated on :utf8 handles

The `sysread()`, `recv()`, `syswrite()` and `send()` operators are deprecated on handles that have the `:utf8` layer, either explicitly, or implicitly, eg., with the `:encoding(UTF-16LE)` layer.

Both `sysread()` and `recv()` currently use only the `:utf8` flag for the stream, ignoring the actual layers. Since `sysread()` and `recv()` do no UTF-8 validation they can end up creating invalidly encoded scalars.

Similarly, `syswrite()` and `send()` use only the `:utf8` flag, otherwise ignoring any layers. If the flag is set, both write the value UTF-8 encoded, even if the layer is some different encoding, such as the example above.

Ideally, all of these operators would completely ignore the `:utf8` state, working only with bytes, but this would result in silently breaking existing code. To avoid this a future version of perl will throw an exception when any of `sysread()`, `recv()`, `syswrite()` or `send()` are called on handle with the `:utf8` layer.

## Performance Enhancements

- The overhead of scope entry and exit has been considerably reduced, so for example subroutine calls, loops and basic blocks are all faster now. This empty function call now takes about a third less time to execute:

  ```
  sub f{} f();
  ```

- Many languages, such as Chinese, are caseless. Perl now knows about most common ones, and skips much of the work when a program tries to change case in them (like `ucfirst()`) or match caselessly (`qr//i`). This will speed up a program, such as a web server, that can operate on multiple languages, while it is operating on a caseless one.

- `/fixed-substr/` has been made much faster.

  On platforms with a libc `memchr()` implementation which makes good use of underlying hardware support, patterns which include fixed substrings will now often be much faster; for example with glibc on a recent x86_64 CPU, this:

  ```
  $s = "a" x 1000 . "wxyz";
  $s =~ /wxyz/ for 1..30000
  ```

  is now about 7 times faster. On systems with slow `memchr()`, e.g. 32-bit ARM Raspberry Pi, there will be a small or little speedup. Conversely, some pathological cases, such as `"ab" x 1000 =~ /aa/` will be slower now; up to 3 times slower on the rPi, 1.5x slower on x86_64.

- Faster addition, subtraction and multiplication.

  Since 5.8.0, arithmetic became slower due to the need to support 64-bit integers. To deal with 64-bit integers, a lot more corner cases need to be checked, which adds time. We now detect common cases where there is no need to check for those corner cases, and special-case them.

- Preincrement, predecrement, postincrement, and postdecrement have been made faster by internally splitting the functions which handled multiple cases into different functions.

- Creating Perl debugger data structures (see ‘‘Debugger Internals’’ in perldebguts) for XSUBs and const subs has been removed. This removed one glob/scalar combo for each unique `.c` file that XSUBs and const subs came from. On startup (`perl -e"0"`) about half a dozen glob/scalar debugger combos were created. Loading XS modules created more glob/scalar combos. These things were being created regardless of whether the perl debugger was being used, and despite the fact that it can’t debug C code anyway

- On Win32, `stat`ing or `-X`ing a path, if the file or directory does not exist, is now 3.5x faster than before.

- Single arguments in list assign are now slightly faster:

  ```
  ($x) = (...);
  (...) = ($x);
  ```

- Less peak memory is now used when compiling regular expression patterns.

## Modules and Pragmata

### Updated Modules and Pragmata

- arybase has been upgraded from version 0.10 to 0.11.
- Attribute::Handlers has been upgraded from version 0.97 to 0.99.
- autodie has been upgraded from version 2.26 to 2.29.
- autouse has been upgraded from version 1.08 to 1.11.
- B has been upgraded from version 1.58 to 1.62.
- B::Deparse has been upgraded from version 1.35 to 1.37.
- base has been upgraded from version 2.22 to 2.23.
- Benchmark has been upgraded from version 1.2 to 1.22.
- bignum has been upgraded from version 0.39 to 0.42.
- bytes has been upgraded from version 1.04 to 1.05.
- Carp has been upgraded from version 1.36 to 1.40.
- Compress::Raw::Bzip2 has been upgraded from version 2.068 to 2.069.
- Compress::Raw::Zlib has been upgraded from version 2.068 to 2.069.
- Config::Perl::V has been upgraded from version 0.24 to 0.25.
- CPAN::Meta has been upgraded from version 2.150001 to 2.150005.
- CPAN::Meta::Requirements has been upgraded from version 2.132 to 2.140.
- CPAN::Meta::YAML has been upgraded from version 0.012 to 0.018.
- Data::Dumper has been upgraded from version 2.158 to 2.160.
- Devel::Peek has been upgraded from version 1.22 to 1.23.
- Devel::PPPort has been upgraded from version 3.31 to 3.32.
- Dumpvalue has been upgraded from version 1.17 to 1.18.
- DynaLoader has been upgraded from version 1.32 to 1.38.
- Encode has been upgraded from version 2.72 to 2.80.
- encoding has been upgraded from version 2.14 to 2.17.
- encoding::warnings has been upgraded from version 0.11 to 0.12.
- English has been upgraded from version 1.09 to 1.10.
- Errno has been upgraded from version 1.23 to 1.25.
- experimental has been upgraded from version 0.013 to 0.016.
- ExtUtils::CBuilder has been upgraded from version 0.280221 to 0.280225.
- ExtUtils::Embed has been upgraded from version 1.32 to 1.33.
- ExtUtils::MakeMaker has been upgraded from version 7.04_01 to 7.10_01.
- ExtUtils::ParseXS has been upgraded from version 3.28 to 3.31.
- ExtUtils::Typemaps has been upgraded from version 3.28 to 3.31.
- feature has been upgraded from version 1.40 to 1.42.
- fields has been upgraded from version 2.17 to 2.23.
- File::Find has been upgraded from version 1.29 to 1.34.
- File::Glob has been upgraded from version 1.24 to 1.26.
- File::Path has been upgraded from version 2.09 to 2.12_01.
- File::Spec has been upgraded from version 3.56 to 3.63.

- Filter::Util::Call has been upgraded from version 1.54 to 1.55.
- Getopt::Long has been upgraded from version 2.45 to 2.48.
- Hash::Util has been upgraded from version 0.18 to 0.19.
- Hash::Util::FieldHash has been upgraded from version 1.15 to 1.19.
- HTTP::Tiny has been upgraded from version 0.054 to 0.056.
- I18N::Langinfo has been upgraded from version 0.12 to 0.13.
- if has been upgraded from version 0.0604 to 0.0606.
- IO has been upgraded from version 1.35 to 1.36.
- IO-Compress has been upgraded from version 2.068 to 2.069.
- IPC::Open3 has been upgraded from version 1.18 to 1.20.
- IPC::SysV has been upgraded from version 2.04 to 2.06_01.
- List::Util has been upgraded from version 1.41 to 1.42_02.
- locale has been upgraded from version 1.06 to 1.08.
- Locale::Codes has been upgraded from version 3.34 to 3.37.
- Math::BigInt has been upgraded from version 1.9997 to 1.999715.
- Math::BigInt::FastCalc has been upgraded from version 0.31 to 0.40.
- Math::BigRat has been upgraded from version 0.2608 to 0.260802.
- Module::CoreList has been upgraded from version 5.20150520 to 5.20160320.
- Module::Metadata has been upgraded from version 1.000026 to 1.000031.
- mro has been upgraded from version 1.17 to 1.18.
- ODBM_File has been upgraded from version 1.12 to 1.14.
- Opcode has been upgraded from version 1.32 to 1.34.
- parent has been upgraded from version 0.232 to 0.234.
- Parse::CPAN::Meta has been upgraded from version 1.4414 to 1.4417.
- Perl::OSType has been upgraded from version 1.008 to 1.009.
- perlfaq has been upgraded from version 5.021009 to 5.021010.
- PerlIO::encoding has been upgraded from version 0.21 to 0.24.
- PerlIO::mmap has been upgraded from version 0.014 to 0.016.
- PerlIO::scalar has been upgraded from version 0.22 to 0.24.
- PerlIO::via has been upgraded from version 0.15 to 0.16.
- Pod::Functions has been upgraded from version 1.09 to 1.10.
- Pod::Perldoc has been upgraded from version 3.25 to 3.25_02.
- Pod::Simple has been upgraded from version 3.29 to 3.32.
- Pod::Usage has been upgraded from version 1.64 to 1.68.
- POSIX has been upgraded from version 1.53 to 1.65.
- Scalar::Util has been upgraded from version 1.41 to 1.42_02.
- SDBM_File has been upgraded from version 1.13 to 1.14.
- SelfLoader has been upgraded from version 1.22 to 1.23.
- Socket has been upgraded from version 2.018 to 2.020_03.
- Storable has been upgraded from version 2.53 to 2.56.
- strict has been upgraded from version 1.09 to 1.11.

- Term::ANSIColor has been upgraded from version 4.03 to 4.04.
- Term::Cap has been upgraded from version 1.15 to 1.17.
- Test has been upgraded from version 1.26 to 1.28.
- Test::Harness has been upgraded from version 3.35 to 3.36.
- Thread::Queue has been upgraded from version 3.05 to 3.08.
- threads has been upgraded from version 2.01 to 2.06.
- threads::shared has been upgraded from version 1.48 to 1.50.
- Tie::File has been upgraded from version 1.01 to 1.02.
- Tie::Scalar has been upgraded from version 1.03 to 1.04.
- Time::HiRes has been upgraded from version 1.9726 to 1.9732.
- Time::Piece has been upgraded from version 1.29 to 1.31.
- Unicode::Collate has been upgraded from version 1.12 to 1.14.
- Unicode::Normalize has been upgraded from version 1.18 to 1.25.
- Unicode::UCD has been upgraded from version 0.61 to 0.64.
- UNIVERSAL has been upgraded from version 1.12 to 1.13.
- utf8 has been upgraded from version 1.17 to 1.19.
- version has been upgraded from version 0.9909 to 0.9916.
- warnings has been upgraded from version 1.32 to 1.36.
- Win32 has been upgraded from version 0.51 to 0.52.
- Win32API::File has been upgraded from version 0.1202 to 0.1203.
- XS::Typemap has been upgraded from version 0.13 to 0.14.
- XSLoader has been upgraded from version 0.20 to 0.21.

## Documentation

### Changes to Existing Documentation

*perlapi*

- The process of using undocumented globals has been documented, namely, that one should send email to perl5−porters@perl.org <mailto:perl5-porters@perl.org> first to get the go-ahead for documenting and using an undocumented function or global variable.

*perlcall*

- A number of cleanups have been made to perlcall, including:
  - use `EXTEND(SP, n)` and `PUSHs()` instead of `XPUSHs()` where applicable and update prose to match
  - add POPu, POPul and POPpbytex to the ‘‘complete list of POP macros’’ and clarify the documentation for some of the existing entries, and a note about side-effects
  - add API documentation for POPu and POPul
  - use ERRSV more efficiently
  - approaches to thread-safety storage of SVs.

*perlfunc*

- The documentation of `hex` has been revised to clarify valid inputs.
- Better explain meaning of negative PIDs in `waitpid`. [GH #15108] <https://github.com/Perl/perl5/issues/15108>
- General cleanup: there’s more consistency now (in POD usage, grammar, code examples), better practices in code examples (use of `my`, removal of bareword filehandles, dropped usage of `&` when calling subroutines, ...), etc.

*perlguts*

- A new section has been added, ‘‘Dynamic Scope and the Context Stack’’ in perlguts, which explains how the perl context stack works.

*perllocale*

- A stronger caution about using locales in threaded applications is given. Locales are not thread-safe, and you can get wrong results or even segfaults if you use them there.

*perlmodlib*

- We now recommend contacting the module-authors list or PAUSE in seeking guidance on the naming of modules.

*perlop*

- The documentation of `qx//` now describes how `$?` is affected.

*perlpolicy*

- This note has been added to perlpolicy:

```
While civility is required, kindness is encouraged; if you have any
doubt about whether you are being civil, simply ask yourself, "Am I
being kind?" and aspire to that.
```

*perlreftut*

- Fix some examples to be strict clean.

*perlrebackslash*

- Clarify that in languages like Japanese and Thai, dictionary lookup is required to determine word boundaries.

*perlsub*

- Updated to note that anonymous subroutines can have signatures.

*perlsyn*

- Fixed a broken example where `=` was used instead of `==` in conditional in do/while example.

*perltie*

- The usage of `FIRSTKEY` and `NEXTKEY` has been clarified.

*perlunicode*

- Discourage use of ’In’ as a prefix signifying the Unicode Block property.

*perlvar*

- The documentation of `$@` was reworded to clarify that it is not just for syntax errors in `eval`. [GH #14572] <https://github.com/Perl/perl5/issues/14572>
- The specific true value of `$!{E...}` is now documented, noting that it is subject to change and not guaranteed.
- Use of `$OLD_PERL_VERSION` is now discouraged.

*perlxs*

- The documentation of `PROTOTYPES` has been corrected; they are *disabled* by default, not *enabled*.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- `%s` must not be a named sequence in transliteration operator
- Can’t find Unicode property definition ‘‘%s’’ in regex;

- Can't redeclare ''%s'' in ''%s''
- Character following \p must be '{' or a single-character Unicode property name in regex;
- Empty \%c in regex; marked by <-- HERE in m/%s/
- Illegal user-defined property name
- Invalid number '%s' for -C option.
- Sequence (?... not terminated in regex; marked by <-- HERE in m/%s/
- Sequence (?P<... not terminated in regex; marked by <-- HERE in m/%s/
- Sequence (?P>... not terminated in regex; marked by <-- HERE in m/%s/

*New Warnings*

- Assuming NOT a POSIX class since `%s` in regex; marked by <-- HERE in m/%s/
- %s() is deprecated on :utf8 handles

### Changes to Existing Diagnostics

- Accessing the `IO` part of a glob as `FILEHANDLE` instead of `IO` is no longer deprecated. It is discouraged to encourage uniformity (so that, for example, one can grep more easily) but it will not be removed. [GH #15105] <https://github.com/Perl/perl5/issues/15105>
- The diagnostic `Hexadecimal float: internal error` has been changed to `Hexadecimal float: internal error (%s)` to include more information.
- Can't modify non-lvalue subroutine call of &%s

  This error now reports the name of the non-lvalue subroutine you attempted to use as an lvalue.
- When running out of memory during an attempt the increase the stack size, previously, perl would die using the cryptic message `panic: av_extend_guts() negative count (-9223372036854775681)`. This has been fixed to show the prettier message: Out of memory during stack extend

## Configuration and Compilation

- `Configure` now acts as if the `-O` option is always passed, allowing command line options to override saved configuration. This should eliminate confusion when command line options are ignored for no obvious reason. `-O` is now permitted, but ignored.
- Bison 3.0 is now supported.
- *Configure* no longer probes for *libnm* by default. Originally this was the ''New Math'' library, but the name has been re-used by the GNOME NetworkManager. [GH #15115] <https://github.com/Perl/perl5/issues/15115>
- Added *Configure* probes for `newlocale`, `freelocale`, and `uselocale`.
- `PPPort.so/PPPort.dll` no longer get installed, as they are not used by `PPPort.pm`, only by its test files.
- It is now possible to specify which compilation date to show on `perl -V` output, by setting the macro `PERL_BUILD_DATE`.
- Using the `NO_HASH_SEED` define in combination with the default hash algorithm `PERL_HASH_FUNC_ONE_AT_A_TIME_HARD` resulted in a fatal error while compiling the interpreter, since Perl 5.17.10. This has been fixed.
- *Configure* should handle spaces in paths a little better.
- No longer generate EBCDIC POSIX-BC tables. We don't believe anyone is using Perl and POSIX-BC at this time, and by not generating these tables it saves time during development, and makes the resulting tar ball smaller.
- The GNU Make makefile for Win32 now supports parallel builds. [perl #126632]
- You can now build perl with MSVC++ on Win32 using GNU Make. [perl #126632]
- The Win32 miniperl now has a real `getcwd` which increases build performance resulting in `getcwd()` being 605x faster in Win32 miniperl.

- Configure now takes `-Dusequadmath` into account when calculating the `alignbytes` configuration variable. Previously the mis-calculated `alignbytes` could cause alignment errors on debugging builds. [perl #127894]

## Testing

- A new test (*t/op/aassign.t*) has been added to test the list assignment operator `OP_AASSIGN`.
- Parallel building has been added to the dmake `makefile.mk` makefile. All Win32 compilers are supported.

## Platform Support

### Platform-Specific Notes

AmigaOS

- The AmigaOS port has been reintegrated into the main tree, based off of Perl 5.22.1.

Cygwin

- Tests are more robust against unusual cygdrive prefixes. [GH #15076] <https://github.com/Perl/perl5/issues/15076>

EBCDIC

UTF-EBCDIC extended

UTF-EBCDIC is like UTF-8, but for EBCDIC platforms. It now has been extended so that it can represent code points up to 2 ** 64 − 1 on platforms with 64-bit words. This brings it into parity with UTF-8. This enhancement requires an incompatible change to the representation of code points in the range 2 ** 30 to 2 ** 31 −1 (the latter was the previous maximum representable code point). This means that a file that contains one of these code points, written out with previous versions of perl cannot be read in, without conversion, by a perl containing this change. We do not believe any such files are in existence, but if you do have one, submit a ticket at perlbug@perl.org <mailto:perlbug@perl.org>, and we will write a conversion script for you.

EBCDIC `cmp()` and `sort()` fixed for UTF-EBCDIC strings

Comparing two strings that were both encoded in UTF-8 (or more precisely, UTF-EBCDIC) did not work properly until now. Since `sort()` uses `cmp()`, this fixes that as well.

EBCDIC `tr///` and `y///` fixed for `\N{}`, and `use utf8` ranges

Perl v5.22 introduced the concept of portable ranges to regular expression patterns. A portable range matches the same set of characters no matter what platform is being run on. This concept is now extended to `tr///`. See `tr///`.

There were also some problems with these operations under `use utf8`, which are now fixed

FreeBSD

- Use the `fdclose()` function from FreeBSD if it is available. [GH #15082] <https://github.com/Perl/perl5/issues/15082>

IRIX

- Under some circumstances IRIX stdio `fgetc()` and `fread()` set the errno to `ENOENT`, which made no sense according to either IRIX or POSIX docs. Errno is now cleared in such cases. [GH #14557] <https://github.com/Perl/perl5/issues/14557>
- Problems when multiplying long doubles by infinity have been fixed. [GH #14993] <https://github.com/Perl/perl5/issues/14993>

MacOS X

- Until now OS X builds of perl have specified a link target of 10.3 (Panther, 2003) but have not specified a compiler target. From now on, builds of perl on OS X 10.6 or later (Snow Leopard, 2008) by default capture the current OS X version and specify that as the explicit build target in both compiler and linker flags, thus preserving binary compatibility for extensions built later regardless of changes in OS X, SDK, or compiler and linker versions. To override the default value used in the build and preserved in the flags, specify `export MACOSX_DEPLOYMENT_TARGET=10.N` before configuring and building perl, where 10.N is the version of OS X you wish to target. In OS X 10.5 or earlier there is no change to the behavior present when those systems were current; the link target is still OS X 10.3 and there is no explicit compiler target.

- Builds with both -DDEBUGGING and threading enabled would fail with a ''panic: free from wrong pool'' error when built or tested from Terminal on OS X. This was caused by perl's internal management of the environment conflicting with an atfork handler using the libc `setenv()` function to update the environment.

  Perl now uses `setenv()`/`unsetenv()` to update the environment on OS X. [GH #14955] <https://github.com/Perl/perl5/issues/14955>

Solaris

- All Solaris variants now build a shared libperl

  Solaris and variants like OpenIndiana now always build with the shared Perl library (Configure -Duseshrplib). This was required for the OpenIndiana builds, but this has also been the setting for Oracle/Sun Perl builds for several years.

Tru64

- Workaround where Tru64 balks when prototypes are listed as `PERL_STATIC_INLINE`, but where the test is build with `-DPERL_NO_INLINE_FUNCTIONS`.

VMS

- On VMS, the math function prototypes in `math.h` are now visible under C++. Now building the POSIX extension with C++ will no longer crash.
- VMS has had `setenv`/`unsetenv` since v7.0 (released in 1996), `Perl_vmssetenv` now always uses `setenv`/`unsetenv`.
- Perl now implements its own `killpg` by scanning for processes in the specified process group, which may not mean exactly the same thing as a Unix process group, but allows us to send a signal to a parent (or master) process and all of its sub-processes. At the perl level, this means we can now send a negative pid like so:

  ```
  kill SIGKILL, -$pid;
  ```

  to signal all processes in the same group as `$pid`.
- For those `%ENV` elements based on the CRTL environ array, we've always preserved case when setting them but did look-ups only after upcasing the key first, which made lower- or mixed-case entries go missing. This problem has been corrected by making `%ENV` elements derived from the environ array case-sensitive on look-up as well as case-preserving on store.
- Environment look-ups for `PERL5LIB` and `PERLLIB` previously only considered logical names, but now consider all sources of `%ENV` as determined by `PERL_ENV_TABLES` and as documented in ''%ENV'' in perlvms.
- The minimum supported version of VMS is now v7.3-2, released in 2003. As a side effect of this change, VAX is no longer supported as the terminal release of OpenVMS VAX was v7.3 in 2001.

Win32

- A new build option `USE_NO_REGISTRY` has been added to the makefiles. This option is off by default, meaning the default is to do Windows registry lookups. This option stops Perl from looking inside the registry for anything. For what values are looked up in the registry see perlwin32. Internally, in C, the name of this option is `WIN32_NO_REGISTRY`.
- The behavior of Perl using `HKEY_CURRENT_USER\Software\Perl` and `HKEY_LOCAL_MACHINE\Software\Perl` to lookup certain values, including `%ENV` vars starting with `PERL` has changed. Previously, the 2 keys were checked for entries at all times through the perl process's life time even if they did not exist. For performance reasons, now, if the root key (i.e. `HKEY_CURRENT_USER\Software\Perl` or `HKEY_LOCAL_MACHINE\Software\Perl`) does not exist at process start time, it will not be checked again for `%ENV` override entries for the remainder of the perl process's life. This more closely matches Unix behavior in that the environment is copied or inherited on startup and changing the variable in the parent process or another process or editing *.bashrc* will not change the environmental variable in other existing, running, processes.

- One glob fetch was removed for each `-X` or `stat` call whether done from Perl code or internally from Perl's C code. The glob being looked up was `${^WIN32_SLOPPY_STAT}` which is a special variable. This makes `-X` and `stat` slightly faster.
- During miniperl's process startup, during the build process, 4 to 8 IO calls related to the process starting *.pl* and the *buildcustomize.pl* file were removed from the code opening and executing the first 1 or 2 *.pl* files.
- Builds using Microsoft Visual C++ 2003 and earlier no longer produce an ''INTERNAL COMPILER ERROR'' message. [perl #126045]
- Visual C++ 2013 builds will now execute on XP and higher. Previously they would only execute on Vista and higher.
- You can now build perl with GNU Make and GCC. [perl #123440]
- `truncate($filename, $size)` now works for files over 4GB in size. [perl #125347]
- Parallel building has been added to the dmake `makefile.mk` makefile. All Win32 compilers are supported.
- Building a 64-bit perl with a 64-bit GCC but a 32-bit gmake would result in an invalid `$Config{archname}` for the resulting perl. [perl #127584]
- Errors set by Winsock functions are now put directly into `$^E`, and the relevant `WSAE*` error codes are now exported from the Errno and POSIX modules for testing this against.

  The previous behavior of putting the errors (converted to POSIX-style `E*` error codes since Perl 5.20.0) into `$!` was buggy due to the non-equivalence of like-named Winsock and POSIX error constants, a relationship between which has unfortunately been established in one way or another since Perl 5.8.0.

  The new behavior provides a much more robust solution for checking Winsock errors in portable software without accidentally matching POSIX tests that were intended for other OSes and may have different meanings for Winsock.

  The old behavior is currently retained, warts and all, for backwards compatibility, but users are encouraged to change any code that tests `$!` against `E*` constants for Winsock errors to instead test `$^E` against `WSAE*` constants. After a suitable deprecation period, the old behavior may be removed, leaving `$!` unchanged after Winsock function calls, to avoid any possible confusion over which error variable to check.

ppc64el
: floating point
  : The floating point format of ppc64el (Debian naming for little-endian PowerPC) is now detected correctly.

## Internal Changes

- The implementation of perl's context stack system, and its internal API, have been heavily reworked. Note that no significant changes have been made to any external APIs, but XS code which relies on such internal details may need to be fixed. The main changes are:
  - The `PUSHBLOCK()`, `POPSUB()` etc. macros have been replaced with static inline functions such as `cx_pushblock()`, `cx_popsub()` etc. These use function args rather than implicitly relying on local vars such as `gimme` and `newsp` being available. Also their functionality has changed: in particular, `cx_popblock()` no longer decrements `cxstack_ix`. The ordering of the steps in the `pp_leave*` functions involving `cx_popblock()`, `cx_popsub()` etc. has changed. See the new documentation, ''Dynamic Scope and the Context Stack'' in perlguts, for details on how to use them.
  - Various macros, which now consistently have a CX_ prefix, have been added:

    ```
    CX_CUR(), CX_LEAVE_SCOPE(), CX_POP()
    ```

    or renamed:

    ```
    CX_POP_SAVEARRAY(), CX_DEBUG(), CX_PUSHSUBST(), CX_POPSUBST()
    ```

- `cx_pushblock()` now saves `PL_savestack_ix` and `PL_tmps_floor`, so `pp_enter*` and `pp_leave*` no longer do

  ```
  ENTER; SAVETMPS; ....; LEAVE
  ```

- `cx_popblock()` now also restores `PL_curpm`.
- In `dounwind()` for every context type, the current savestack frame is now processed before each context is popped; formerly this was only done for sub-like context frames. This action has been removed from `cx_popsub()` and placed into its own macro, `CX_LEAVE_SCOPE(cx)`, which must be called before `cx_popsub()` etc.

  `dounwind()` now also does a `cx_popblock()` on the last popped frame (formerly it only did the `cx_popsub()` etc. actions on each frame).
- The temps stack is now freed on scope exit; previously, temps created during the last statement of a block wouldn't be freed until the next `nextstate` following the block (apart from an existing hack that did this for recursive subs in scalar context); and in something like `f(g())`, the temps created by the last statement in `g()` would formerly not be freed until the statement following the return from `f()`.
- Most values that were saved on the savestack on scope entry are now saved in suitable new fields in the context struct, and saved and restored directly by `cx_pushfoo()` and `cx_popfoo()`, which is much faster.
- Various context struct fields have been added, removed or modified.
- The handling of `@_` in `cx_pushsub()` and `cx_popsub()` has been considerably tidied up, including removing the `argarray` field from the context struct, and extracting out some common (but rarely used) code into a separate function, `clear_defarray()`. Also, useful subsets of `cx_popsub()` which had been unrolled in places like `pp_goto` have been gathered into the new functions `cx_popsub_args()` and `cx_popsub_common()`.
- `pp_leavesub` and `pp_leavesublv` now use the same function as the rest of the `pp_leave*`'s to process return args.
- `CXp_FOR_PAD` and `CXp_FOR_GV` flags have been added, and `CXt_LOOP_FOR` has been split into `CXt_LOOP_LIST`, `CXt_LOOP_ARY`.
- Some variables formerly declared by `dMULTICALL` (but not documented) have been removed.

- The obscure `PL_timesbuf` variable, effectively a vestige of Perl 1, has been removed. It was documented as deprecated in Perl 5.20, with a statement that it would be removed early in the 5.21.x series; that has now finally happened. [GH #13632] <https://github.com/Perl/perl5/issues/13632>
- An unwarranted assertion in `Perl_newATTRSUB_x()` has been removed. If a stub subroutine definition with a prototype has been seen, then any subsequent stub (or definition) of the same subroutine with an attribute was causing an assertion failure because of a null pointer. [GH #15081] <https://github.com/Perl/perl5/issues/15081>
- `::` has been replaced by `__` in `ExtUtils::ParseXS`, like it's done for parameters/return values. This is more consistent, and simplifies writing XS code wrapping C++ classes into a nested Perl namespace (it requires only a typedef for `Foo__Bar` rather than two, one for `Foo_Bar` and the other for `Foo::Bar`).
- The `to_utf8_case()` function is now deprecated. Instead use `toUPPER_utf8`, `toTITLE_utf8`, `toLOWER_utf8`, and `toFOLD_utf8`. (See <http://nntp.perl.org/group/perl.perl5.porters/233287>.)
- Perl core code and the threads extension have been annotated so that, if Perl is configured to use threads, then during compile-time clang (3.6 or later) will warn about suspicious uses of mutexes. See <http://clang.llvm.org/docs/ThreadSafetyAnalysis.html> for more information.
- The `signbit()` emulation has been enhanced. This will help older and/or more exotic platforms or configurations.

- Most EBCDIC-specific code in the core has been unified with non-EBCDIC code, to avoid repetition and make maintenance easier.
- MSWin32 code for `$^X` has been moved out of the *win32* directory to *caretx.c*, where other operating systems set that variable.
- `sv_ref()` is now part of the API.
- ''sv_backoff'' in perlapi had its return type changed from `int` to `void`. It previously has always returned `0` since Perl 5.000 stable but that was undocumented. Although `sv_backoff` is marked as public API, XS code is not expected to be impacted since the proper API call would be through public API `sv_setsv(sv, &PL_sv_undef)`, or quasi-public `SvOOK_off`, or non-public `SvOK_off` calls, and the return value of `sv_backoff` was previously a meaningless constant that can be rewritten as `(sv_backoff(sv),0)`.
- The `EXTEND` and `MEXTEND` macros have been improved to avoid various issues with integer truncation and wrapping. In particular, some casts formerly used within the macros have been removed. This means for example that passing an unsigned `nitems` argument is likely to raise a compiler warning now (it's always been documented to require a signed value; formerly int, lately SSize_t).
- `PL_sawalias` and `GPf_ALIASED_SV` have been removed.
- `GvASSIGN_GENERATION` and `GvASSIGN_GENERATION_set` have been removed.

## Selected Bug Fixes

- It now works properly to specify a user-defined property, such as

  ```
  qr/\p{mypkg1::IsMyProperty}/i
  ```

  with `/i` caseless matching, an explicit package name, and *IsMyProperty* not defined at the time of the pattern compilation.
- Perl's `memcpy()`, `memmove()`, `memset()` and `memcmp()` fallbacks are now more compatible with the originals. [perl #127619]
- Fixed the issue where a `s///r`) with **−DPERL_NO_COW** attempts to modify the source SV, resulting in the program dying. [perl #127635]
- Fixed an EBCDIC-platform-only case where a pattern could fail to match. This occurred when matching characters from the set of C1 controls when the target matched string was in UTF−8.
- Narrow the filename check in *strict.pm* and *warnings.pm*. Previously, it assumed that if the filename (without the *.pmc?* extension) differed from the package name, if was a misspelled use statement (i.e. `use Strict` instead of `use strict`). We now check whether there's really a miscapitalization happening, and not some other issue.
- Turn an assertion into a more user friendly failure when parsing regexes. [perl #127599]
- Correctly raise an error when trying to compile patterns with unterminated character classes while there are trailing backslashes. [perl #126141].
- Line numbers larger than 2**31−1 but less than 2**32 are no longer returned by `caller()` as negative numbers. [perl #126991]
- `unless ( assignment )` now properly warns when syntax warnings are enabled. [perl #127122]
- Setting an `ISA` glob to an array reference now properly adds `isaelem` magic to any existing elements. Previously modifying such an element would not update the ISA cache, so method calls would call the wrong function. Perl would also crash if the `ISA` glob was destroyed, since new code added in 5.23.7 would try to release the `isaelem` magic from the elements. [perl #127351]
- If a here-doc was found while parsing another operator, the parser had already read end of file, and the here-doc was not terminated, perl could produce an assertion or a segmentation fault. This now reliably complains about the unterminated here-doc. [perl #125540]
- `untie()` would sometimes return the last value returned by the `UNTIE()` handler as well as its normal value, messing up the stack. [perl #126621]

- Fixed an operator precedence problem when `castflags & 2` is true. [perl #127474]
- Caching of DESTROY methods could result in a non-pointer or a non-STASH stored in the `SvSTASH()` slot of a stash, breaking the B `STASH()` method. The DESTROY method is now cached in the MRO metadata for the stash. [perl #126410]
- The AUTOLOAD method is now called when searching for a DESTROY method, and correctly sets `$AUTOLOAD` too. [perl #124387] [perl #127494]
- Avoid parsing beyond the end of the buffer when processing a `#line` directive with no filename. [perl #127334]
- Perl now raises a warning when a regular expression pattern looks like it was supposed to contain a POSIX class, like `qr/[[:alpha:]]/`, but there was some slight defect in its specification which causes it to instead be treated as a regular bracketed character class. An example would be missing the second colon in the above like this: `qr/[[:alpha]]/`. This compiles to match a sequence of two characters. The second is `"]"`, and the first is any of: `"["`, `":"`, `"a"`, `"h"`, `"l"`, or `"p"`. This is unlikely to be the intended meaning, and now a warning is raised. No warning is raised unless the specification is very close to one of the 14 legal POSIX classes. (See ''POSIX Character Classes'' in perlrecharclass.) [perl #8904]
- Certain regex patterns involving a complemented POSIX class in an inverted bracketed character class, and matching something else optionally would improperly fail to match. An example of one that could fail is `qr/_?[^\Wbar]\x{100}/`. This has been fixed. [perl #127537]
- Perl 5.22 added support to the C99 hexadecimal floating point notation, but sometimes misparses hex floats. This has been fixed. [perl #127183]
- A regression that allowed undeclared barewords in hash keys to work despite strictures has been fixed. [GH #15099] <https://github.com/Perl/perl5/issues/15099>
- Calls to the placeholder `&PL_sv_yes` used internally when an `import()` or `unimport()` method isn't found now correctly handle scalar context. [GH #14902] <https://github.com/Perl/perl5/issues/14902>
- Report more context when we see an array where we expect to see an operator and avoid an assertion failure. [GH #14472] <https://github.com/Perl/perl5/issues/14472>
- Modifying an array that was previously a package `@ISA` no longer causes assertion failures or crashes. [GH #14492] <https://github.com/Perl/perl5/issues/14492>
- Retain binary compatibility across plain and DEBUGGING perl builds. [GH #15122] <https://github.com/Perl/perl5/issues/15122>
- Avoid leaking memory when setting `$ENV{foo}` on darwin. [GH #14955] <https://github.com/Perl/perl5/issues/14955>
- `/...\G/` no longer crashes on utf8 strings. When `\G` is a fixed number of characters from the start of the regex, perl needs to count back that many characters from the current `pos()` position and start matching from there. However, it was counting back bytes rather than characters, which could lead to panics on utf8 strings.
- In some cases operators that return integers would return negative integers as large positive integers. [GH #15049] <https://github.com/Perl/perl5/issues/15049>
- The `pipe()` operator would assert for DEBUGGING builds instead of producing the correct error message. The condition asserted on is detected and reported on correctly without the assertions, so the assertions were removed. [GH #15015] <https://github.com/Perl/perl5/issues/15015>
- In some cases, failing to parse a here-doc would attempt to use freed memory. This was caused by a pointer not being restored correctly. [GH #15009] <https://github.com/Perl/perl5/issues/15009>
- `@x = sort { *a = 0; $a <=> $b } 0 .. 1` no longer frees the GP for *a before restoring its SV slot. [GH #14595] <https://github.com/Perl/perl5/issues/14595>
- Multiple problems with the new hexadecimal floating point printf format `%a` were fixed: [GH #15032] <https://github.com/Perl/perl5/issues/15032>, [GH #15033] <https://github.com/Perl/perl5/issues/15033>, [GH #15074] <https://github.com/Perl/perl5/issues/15074>

- Calling `mg_set()` in `leave_scope()` no longer leaks.
- A regression from Perl v5.20 was fixed in which debugging output of regular expression compilation was wrong. (The pattern was correctly compiled, but what got displayed for it was wrong.)
- `\b{sb}` works much better. In Perl v5.22.0, this new construct didn't seem to give the expected results, yet passed all the tests in the extensive suite furnished by Unicode. It turns out that it was because these were short input strings, and the failures had to do with longer inputs.
- Certain syntax errors in ''Extended Bracketed Character Classes'' in perlrecharclass caused panics instead of the proper error message. This has now been fixed. [perl #126481]
- Perl 5.20 added a message when a quantifier in a regular expression was useless, but then caused the parser to skip it; this caused the surplus quantifier to be silently ignored, instead of throwing an error. This is now fixed. [perl #126253]
- The switch to building non-XS modules last in win32/makefile.mk (introduced by design as part of the changes to enable parallel building) caused the build of POSIX to break due to problems with the version module. This is now fixed.
- Improved parsing of hex float constants.
- Fixed an issue with `pack` where `pack "H"` (and `pack "h"`) could read past the source when given a non−utf8 source, and a utf8 target. [perl #126325]
- Fixed several cases where perl would abort due to a segmentation fault, or a C−level assert. [perl #126615], [perl #126602], [perl #126193].
- There were places in regular expression patterns where comments (`(?#...)`) weren't allowed, but should have been. This is now fixed. [GH #12755] <https://github.com/Perl/perl5/issues/12755>
- Some regressions from Perl 5.20 have been fixed, in which some syntax errors in `(?[...])` constructs within regular expression patterns could cause a segfault instead of a proper error message. [GH #14933] <https://github.com/Perl/perl5/issues/14933> [GH #14996] <https://github.com/Perl/perl5/issues/14996>
- Another problem with `(?[...])` constructs has been fixed wherein things like `\c]` could cause panics. [GH #14934] <https://github.com/Perl/perl5/issues/14934>
- Some problems with attempting to extend the perl stack to around 2G or 4G entries have been fixed. This was particularly an issue on 32−bit perls built to use 64−bit integers, and was easily noticeable with the list repetition operator, e.g.

  ```
  @a = (1) x $big_number
  ```

  Formerly perl may have crashed, depending on the exact value of `$big_number`; now it will typically raise an exception. [GH #14880] <https://github.com/Perl/perl5/issues/14880>
- In a regex conditional expression `(?(condition)yes−pattern|no−pattern)`, if the condition is `(?!)` then perl failed the match outright instead of matching the no-pattern. This has been fixed. [GH #14947] <https://github.com/Perl/perl5/issues/14947>
- The special backtracking control verbs `(*VERB:ARG)` now all allow an optional argument and set `REGERROR`/`REGMARK` appropriately as well. [GH #14937] <https://github.com/Perl/perl5/issues/14937>
- Several bugs, including a segmentation fault, have been fixed with the boundary checking constructs (introduced in Perl 5.22) `\b{gcb}`, `\b{sb}`, `\b{wb}`, `\B{gcb}`, `\B{sb}`, and `\B{wb}`. All the `\B{}` ones now match an empty string; none of the `\b{}` ones do. [GH #14976] <https://github.com/Perl/perl5/issues/14976>
- Duplicating a closed file handle for write no longer creates a filename of the form *GLOB(0xXXXXXXXX)*. [perl #125115]
- Warning fatality is now ignored when rewinding the stack. This prevents infinite recursion when the now fatal error also causes rewinding of the stack. [perl #123398]

- In perl v5.22.0, the logic changed when parsing a numeric parameter to the −C option, such that the successfully parsed number was not saved as the option value if it parsed to the end of the argument. [perl #125381]
- The PadlistNAMES macro is an lvalue again.
- Zero −DPERL_TRACE_OPS memory for sub-threads.

  `perl_clone_using()` was missing Zero init of PL_op_exec_cnt[]. This caused sub-threads in threaded −DPERL_TRACE_OPS builds to spew exceedingly large op-counts at destruct. These counts would print `%x` as "ABABABAB", clearly a mem-poison value.
- A leak in the XS typemap caused one scalar to be leaked each time a `FILE *` or a `PerlIO *` was `OUTPUT:`ed or imported to Perl, since perl 5.000. These particular typemap entries are thought to be extremely rarely used by XS modules. [perl #124181]
- `alarm()` and `sleep()` will now warn if the argument is a negative number and return undef. Previously they would pass the negative value to the underlying C function which may have set up a timer with a surprising value.
- Perl can again be compiled with any Unicode version. This used to (mostly) work, but was lost in v5.18 through v5.20. The property `Name_Alias` did not exist prior to Unicode 5.0. Unicode::UCD incorrectly said it did. This has been fixed.
- Very large code-points (beyond Unicode) in regular expressions no longer cause a buffer overflow in some cases when converted to UTF−8. [GH #14858] <https://github.com/Perl/perl5/issues/14858>
- The integer overflow check for the range operator (...) in list context now correctly handles the case where the size of the range is larger than the address space. This could happen on 32−bits with −Duse64bitint. [GH #14843] <https://github.com/Perl/perl5/issues/14843>
- A crash with `%::=(); J->${\"::"}` has been fixed. [GH #14790] <https://github.com/Perl/perl5/issues/14790>
- `qr/(?[ () ])/` no longer segfaults, giving a syntax error message instead. [perl #125805]
- Regular expression possessive quantifier v5.20 regression now fixed. `qr/`*PAT*`{`*min*`,`*max*`}+/` is supposed to behave identically to `qr/(?>`*PAT*`{`*min*`,`*max*`})/`. Since v5.20, this didn't work if *min* and *max* were equal. [perl #125825]
- `BEGIN <>` no longer segfaults and properly produces an error message. [perl #125341]
- In `tr///` an illegal backwards range like `tr/\x{101}-\x{100}//` was not always detected, giving incorrect results. This is now fixed.

## Acknowledgements

Perl 5.24.0 represents approximately 11 months of development since Perl 5.24.0 and contains approximately 360,000 lines of changes across 1,800 files from 75 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 250,000 lines of changes to 1,200 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.24.0:

Aaron Crane, Aaron Priven, Abigail, Achim Gratz, Alexander D'Archangel, Alex Vandiver, Andreas König, Andy Broad, Andy Dougherty, Aristotle Pagaltzis, Chase Whitener, Chas. Owens, Chris 'BinGOs' Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, Dan Collins, Daniel Dragan, David Golden, David Mitchell, Doug Bell, Dr.Ruud, Ed Avis, Ed J, Father Chrysostomos, Herbert Breunung, H.Merijn Brand, Hugo van der Sanden, Ivan Pozdeev, James E Keenan, Jan Dubois, Jarkko Hietaniemi, Jerry D. Hedden, Jim Cromie, John Peacock, John SJ Anderson, Karen Etheridge, Karl Williamson, kmx, Leon Timmermans, Ludovic E. R. Tolhurst-Cleaver, Lukas Mai, Martijn Lievaart, Matthew Horsfall, Mattia Barbon, Max Maischein, Mohammed El-Afifi, Nicholas Clark, Nicolas R., Niko Tyni, Peter John Acklam, Peter Martini, Peter Rabbitson, Pip Cet, Rafael Garcia-Suarez, Reini Urban, Ricardo Signes, Sawyer X, Shlomi Fish, Sisyphus, Stanislaw Pusep, Steffen Müller, Stevan Little, Steve Hay, Sullivan Beck, Thomas Sibley, Todd Rinaldo, Tom Hukins, Tony Cook, Unicode Consortium, Victor Adam, Vincent Pit, Vladimir Timofeev, Yves Orton, Zachary Storer, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at https://rt.perl.org/ . There may also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5241delta − what is new for perl v5.24.1

## DESCRIPTION

This document describes differences between the 5.24.0 release and the 5.24.1 release.

If you are upgrading from an earlier release such as 5.22.0, first read perl5240delta, which describes differences between 5.22.0 and 5.24.0.

## Security

### −Di switch is now required for PerlIO debugging output

Previously PerlIO debugging output would be sent to the file specified by the `PERLIO_DEBUG` environment variable if perl wasn't running setuid and the **−T** or **−t** switches hadn't been parsed yet.

If perl performed output at a point where it hadn't yet parsed its switches this could result in perl creating or overwriting the file named by `PERLIO_DEBUG` even when the **−T** switch had been supplied.

Perl now requires the **−Di** switch to produce PerlIO debugging output. By default this is written to `stderr`, but can optionally be redirected to a file by setting the `PERLIO_DEBUG` environment variable.

If perl is running setuid or the **−T** switch was supplied `PERLIO_DEBUG` is ignored and the debugging output is sent to `stderr` as for any other **−D** switch.

### Core modules and tools no longer search "." for optional modules

The tools and many modules supplied in core no longer search the default current directory entry in `@INC` for optional modules. For example, Storable will remove the final "." from `@INC` before trying to load Log::Agent.

This prevents an attacker injecting an optional module into a process run by another user where the current directory is writable by the attacker, e.g. the */tmp* directory.

In most cases this removal should not cause problems, but difficulties were encountered with base, which treats every module name supplied as optional. These difficulties have not yet been resolved, so for this release there are no changes to base. We hope to have a fix for base in Perl 5.24.2.

To protect your own code from this attack, either remove the default "." entry from `@INC` at the start of your script, so:

```
#!/usr/bin/perl
use strict;
...
```

becomes:

```
#!/usr/bin/perl
BEGIN { pop @INC if $INC[-1] eq '.' }
use strict;
...
```

or for modules, remove "." from a localized `@INC`, so:

```
my $can_foo = eval { require Foo; }
```

becomes:

```
my $can_foo = eval {
    local @INC = @INC;
    pop @INC if $INC[-1] eq '.';
    require Foo;
};
```

## Incompatible Changes

Other than the security changes above there are no changes intentionally incompatible with Perl 5.24.0. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.04 to 2.04_01.
- bignum has been upgraded from version 0.42 to 0.42_01.
- CPAN has been upgraded from version 2.11 to 2.11_01.
- Digest has been upgraded from version 1.17 to 1.17_01.
- Digest::SHA has been upgraded from version 5.95 to 5.95_01.
- Encode has been upgraded from version 2.80 to 2.80_01.
- ExtUtils::MakeMaker has been upgraded from version 7.10_01 to 7.10_02.
- File::Fetch has been upgraded from version 0.48 to 0.48_01.
- File::Spec has been upgraded from version 3.63 to 3.63_01.
- HTTP::Tiny has been upgraded from version 0.056 to 0.056_001.
- IO has been upgraded from version 1.36 to 1.36_01.
- The IO-Compress modules have been upgraded from version 2.069 to 2.069_001.
- IPC::Cmd has been upgraded from version 0.92 to 0.92_01.
- JSON::PP has been upgraded from version 2.27300 to 2.27300_01.
- Locale::Maketext has been upgraded from version 1.26 to 1.26_01.
- Locale::Maketext::Simple has been upgraded from version 0.21 to 0.21_01.
- Memoize has been upgraded from version 1.03 to 1.03_01.
- Module::CoreList has been upgraded from version 5.20160506 to 5.20170114_24.
- Net::Ping has been upgraded from version 2.43 to 2.43_01.
- Parse::CPAN::Meta has been upgraded from version 1.4417 to 1.4417_001.
- Pod::Html has been upgraded from version 1.22 to 1.2201.
- Pod::Perldoc has been upgraded from version 3.25_02 to 3.25_03.
- Storable has been upgraded from version 2.56 to 2.56_01.
- Sys::Syslog has been upgraded from version 0.33 to 0.33_01.
- Test has been upgraded from version 1.28 to 1.28_01.
- Test::Harness has been upgraded from version 3.36 to 3.36_01.
- XSLoader has been upgraded from version 0.21 to 0.22, fixing a security hole in which binary files could be loaded from a path outside of `@INC`. [GH #15418] <https://github.com/Perl/perl5/issues/15418>

## Documentation

### Changes to Existing Documentation

*perlapio*

- The documentation of `PERLIO_DEBUG` has been updated.

*perlrun*

- The new **-Di** switch has been documented, and the documentation of `PERLIO_DEBUG` has been updated.

## Testing

- A new test script, *t/run/switchDx.t*, has been added to test that the new **-Di** switch is working correctly.

## Selected Bug Fixes

- The change to hashbang redirection introduced in Perl 5.24.0, whereby perl would redirect to another interpreter (Perl 6) if it found a hashbang path which contains ''perl'' followed by ''6'', has been reverted because it broke in cases such as `#!/opt/perl64/bin/perl`.

## Acknowledgements

Perl 5.24.1 represents approximately 8 months of development since Perl 5.24.0 and contains approximately 8,100 lines of changes across 240 files from 18 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 2,200 lines of changes to 170 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.24.1:

Aaron Crane, Alex Vandiver, Aristotle Pagaltzis, Chad Granum, Chris 'BinGOs' Williams, Craig A. Berry, Father Chrysostomos, James E Keenan, Jarkko Hietaniemi, Karen Etheridge, Leon Timmermans, Matthew Horsfall, Ricardo Signes, Sawyer X, Sébastien Aperghis-Tramoni, Stevan Little, Steve Hay, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the Perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5242delta − what is new for perl v5.24.2

## DESCRIPTION

This document describes differences between the 5.24.1 release and the 5.24.2 release.

If you are upgrading from an earlier release such as 5.24.0, first read perl5241delta, which describes differences between 5.24.0 and 5.24.1.

## Security

### Improved handling of '.' in `@INC` in base.pm

The handling of (the removal of) `'.'` in `@INC` in base has been improved. This resolves some problematic behaviour in the approach taken in Perl 5.24.1, which is probably best described in the following two threads on the Perl 5 Porters mailing list: <http://www.nntp.perl.org/group/perl.perl5.porters/2016/08/msg238991.html>, <http://www.nntp.perl.org/group/perl.perl5.porters/2016/10/msg240297.html>.

### "Escaped" colons and relative paths in PATH

On Unix systems, Perl treats any relative paths in the PATH environment variable as tainted when starting a new process. Previously, it was allowing a backslash to escape a colon (unlike the OS), consequently allowing relative paths to be considered safe if the PATH was set to something like `/\:.`. The check has been fixed to treat `.` as tainted in that example.

## Modules and Pragmata

### Updated Modules and Pragmata

- base has been upgraded from version 2.23 to 2.23_01.
- Module::CoreList has been upgraded from version 5.20170114_24 to 5.20170715_24.

## Selected Bug Fixes

- Fixed a crash with `s///l` where it thought it was dealing with UTF−8 when it wasn't. [GH #15543] <https://github.com/Perl/perl5/issues/15543>

## Acknowledgements

Perl 5.24.2 represents approximately 6 months of development since Perl 5.24.1 and contains approximately 2,500 lines of changes across 53 files from 18 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 960 lines of changes to 17 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.24.2:

Aaron Crane, Abigail, Aristotle Pagaltzis, Chris 'BinGOs' Williams, Dan Collins, David Mitchell, Eric Herman, Father Chrysostomos, James E Keenan, Karl Williamson, Lukas Mai, Renee Baecker, Ricardo Signes, Sawyer X, Stevan Little, Steve Hay, Tony Cook, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5243delta − what is new for perl v5.24.3

## DESCRIPTION

This document describes differences between the 5.24.2 release and the 5.24.3 release.

If you are upgrading from an earlier release such as 5.24.1, first read perl5242delta, which describes differences between 5.24.1 and 5.24.2.

## Security

### [CVE−2017−12837] Heap buffer overflow in regular expression compiler

Compiling certain regular expression patterns with the case-insensitive modifier could cause a heap buffer overflow and crash perl. This has now been fixed. [GH #16021] <https://github.com/Perl/perl5/issues/16021>

### [CVE−2017−12883] Buffer over-read in regular expression parser

For certain types of syntax error in a regular expression pattern, the error message could either contain the contents of a random, possibly large, chunk of memory, or could crash perl. This has now been fixed. [GH #16025] <https://github.com/Perl/perl5/issues/16025>

### [CVE−2017−12814] `$ENV{$key}` stack buffer overflow on Windows

A possible stack buffer overflow in the `%ENV` code on Windows has been fixed by removing the buffer completely since it was superfluous anyway. [GH #16051] <https://github.com/Perl/perl5/issues/16051>

## Incompatible Changes

There are no changes intentionally incompatible with 5.24.2. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20170715_24 to 5.20170922_24.
- POSIX has been upgraded from version 1.65 to 1.65_01.
- Time::HiRes has been upgraded from version 1.9733 to 1.9741.

  [GH #15396] <https://github.com/Perl/perl5/issues/15396> [GH #15401] <https://github.com/Perl/perl5/issues/15401> [GH #15524] <https://github.com/Perl/perl5/issues/15524> [cpan #120032] <https://rt.cpan.org/Public/Bug/Display.html?id=120032>

## Configuration and Compilation

- When building with GCC 6 and link-time optimization (the **−flto** option to **gcc**), *Configure* was treating all probed symbols as present on the system, regardless of whether they actually exist. This has been fixed. [GH #15322] <https://github.com/Perl/perl5/issues/15322>
- *Configure* now aborts if both `-Duselongdouble` and `-Dusequadmath` are requested. [GH #14944] <https://github.com/Perl/perl5/issues/14944>
- Fixed a bug in which *Configure* could append `-quadmath` to the archname even if it was already present. [GH #15423] <https://github.com/Perl/perl5/issues/15423>
- Clang builds with `-DPERL_GLOBAL_STRUCT` or `-DPERL_GLOBAL_STRUCT_PRIVATE` have been fixed (by disabling Thread Safety Analysis for these configurations).

## Platform Support

### Platform-Specific Notes

VMS

- `configure.com` now recognizes the VSI-branded C compiler.

Windows

- Building XS modules with GCC 6 in a 64−bit build of Perl failed due to incorrect mapping of `strtoll` and `strtoull`. This has now been fixed. [GH #16074] <https://github.com/Perl/perl5/issues/16074> [cpan #121683] <https://rt.cpan.org/Public/Bug/Display.html?id=121683> [cpan #122353] <https://rt.cpan.org/Public/Bug/Display.html?id=122353>

## Selected Bug Fixes

- `/@0{0*->@*/*0` and similar contortions used to crash, but no longer do, but merely produce a syntax error. [GH #15333] <https://github.com/Perl/perl5/issues/15333>
- `do` or `require` with an argument which is a reference or typeglob which, when stringified, contains a null character, started crashing in Perl 5.20, but has now been fixed. [GH #15337] <https://github.com/Perl/perl5/issues/15337>
- Expressions containing an `&&` or `||` operator (or their synonyms `and` and `or`) were being compiled incorrectly in some cases. If the left-hand side consisted of either a negated bareword constant or a negated `do {}` block containing a constant expression, and the right-hand side consisted of a negated non-foldable expression, one of the negations was effectively ignored. The same was true of `if` and `unless` statement modifiers, though with the left-hand and right-hand sides swapped. This long-standing bug has now been fixed. [GH #15285] <https://github.com/Perl/perl5/issues/15285>
- `reset` with an argument no longer crashes when encountering stash entries other than globs. [GH #15314] <https://github.com/Perl/perl5/issues/15314>
- Assignment of hashes to, and deletion of, typeglobs named `*::::::` no longer causes crashes. [GH #15307] <https://github.com/Perl/perl5/issues/15307>
- Assignment variants of any bitwise ops under the `bitwise` feature would crash if the left-hand side was an array or hash. [GH #15346] <https://github.com/Perl/perl5/issues/15346>
- `socket` now leaves the error code returned by the system in `$!` on failure. [GH #15383] <https://github.com/Perl/perl5/issues/15383>
- Parsing bad POSIX charclasses no longer leaks memory. [GH #15382] <https://github.com/Perl/perl5/issues/15382>
- Since Perl 5.20, line numbers have been off by one when perl is invoked with the **-x** switch. This has been fixed. [GH #15413] <https://github.com/Perl/perl5/issues/15413>
- Some obscure cases of subroutines and file handles being freed at the same time could result in crashes, but have been fixed. The crash was introduced in Perl 5.22. [GH #15435] <https://github.com/Perl/perl5/issues/15435>
- Some regular expression parsing glitches could lead to assertion failures with regular expressions such as `/(?<=/` and `/(?<!/`. This has now been fixed. [GH #15332] <https://github.com/Perl/perl5/issues/15332>
- `gethostent` and similar functions now perform a null check internally, to avoid crashing with the torsocks library. This was a regression from Perl 5.22. [GH #15478] <https://github.com/Perl/perl5/issues/15478>
- Mentioning the same constant twice in a row (which is a syntax error) no longer fails an assertion under debugging builds. This was a regression from Perl 5.20. [GH #15017] <https://github.com/Perl/perl5/issues/15017>
- In Perl 5.24 `fchown` was changed not to accept negative one as an argument because in some platforms that is an error. However, in some other platforms that is an acceptable argument. This change has been reverted. [GH #15523] <https://github.com/Perl/perl5/issues/15523>.
- `@{x` followed by a newline where `"x"` represents a control or non-ASCII character no longer produces a garbled syntax error message or a crash. [GH #15518] <https://github.com/Perl/perl5/issues/15518>
- A regression in Perl 5.24 with `tr/\N{U+...}/foo/` when the code point was between 128 and 255 has been fixed. [GH #15475] <https://github.com/Perl/perl5/issues/15475>.
- Many issues relating to `printf "%a"` of hexadecimal floating point were fixed. In addition, the ‘‘subnormals’’ (formerly known as ‘‘denormals’’) floating point numbers are now supported both with the plain IEEE 754 floating point numbers (64−bit or 128−bit) and the x86 80−bit ‘‘extended precision’’. Note that subnormal hexadecimal floating point literals will give a warning about ‘‘exponent underflow’’. [GH #15495] <https://github.com/Perl/perl5/issues/15495> [GH #15502] <https://github.com/Perl/perl5/issues/15502> [GH #15503] <https://github.com/Perl/perl5/issues/15503> [GH #15504]

<https://github.com/Perl/perl5/issues/15504> [GH #15505]
<https://github.com/Perl/perl5/issues/15505> [GH #15510]
<https://github.com/Perl/perl5/issues/15510> [GH #15512]
<https://github.com/Perl/perl5/issues/15512>

- The parser could sometimes crash if a bareword came after `evalbytes`. [GH #15586] <https://github.com/Perl/perl5/issues/15586>
- Fixed a place where the regex parser was not setting the syntax error correctly on a syntactically incorrect pattern. [GH #15565] <https://github.com/Perl/perl5/issues/15565>
- A vulnerability in Perl's `sprintf` implementation has been fixed by avoiding a possible memory wrap. [GH #15970] <https://github.com/Perl/perl5/issues/15970>

## Acknowledgements

Perl 5.24.3 represents approximately 2 months of development since Perl 5.24.2 and contains approximately 3,200 lines of changes across 120 files from 23 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,600 lines of changes to 56 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.24.3:

Aaron Crane, Craig A. Berry, Dagfinn Ilmari Mannsåker, Dan Collins, Daniel Dragan, Dave Cross, David Mitchell, Eric Herman, Father Chrysostomos, H.Merijn Brand, Hugo van der Sanden, James E Keenan, Jarkko Hietaniemi, John SJ Anderson, Karl Williamson, Ken Brown, Lukas Mai, Matthew Horsfall, Stevan Little, Steve Hay, Steven Humphrey, Tony Cook, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5244delta − what is new for perl v5.24.4

## DESCRIPTION

This document describes differences between the 5.24.3 release and the 5.24.4 release.

If you are upgrading from an earlier release such as 5.24.2, first read perl5243delta, which describes differences between 5.24.2 and 5.24.3.

## Security

### [CVE−2018−6797] heap-buffer-overflow (WRITE of size 1) in S_regatom (regcomp.c)

A crafted regular expression could cause a heap buffer write overflow, with control over the bytes written. [GH #16185] <https://github.com/Perl/perl5/issues/16185>

### [CVE−2018−6798] Heap-buffer-overflow in Perl_ _byte_dump_string (utf8.c)

Matching a crafted locale dependent regular expression could cause a heap buffer read overflow and potentially information disclosure. [GH #16143] <https://github.com/Perl/perl5/issues/16143>

### [CVE−2018−6913] heap-buffer-overflow in S_pack_rec

`pack()` could cause a heap buffer write overflow with a large item count. [GH #16098] <https://github.com/Perl/perl5/issues/16098>

### Assertion failure in Perl_ _core_swash_init (utf8.c)

Control characters in a supposed Unicode property name could cause perl to crash. This has been fixed. [perl #132055] <https://rt.perl.org/Public/Bug/Display.html?id=132055> [perl #132553] <https://rt.perl.org/Public/Bug/Display.html?id=132553> [perl #132658] <https://rt.perl.org/Public/Bug/Display.html?id=132658>

## Incompatible Changes

There are no changes intentionally incompatible with 5.24.3. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20170922_24 to 5.20180414_24.

## Selected Bug Fixes

- The `readpipe()` built-in function now checks at compile time that it has only one parameter expression, and puts it in scalar context, thus ensuring that it doesn’t corrupt the stack at runtime. [GH #2793] <https://github.com/Perl/perl5/issues/2793>

## Acknowledgements

Perl 5.24.4 represents approximately 7 months of development since Perl 5.24.3 and contains approximately 2,400 lines of changes across 49 files from 12 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,300 lines of changes to 12 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.24.4:

Abigail, Chris ’BinGOs’ Williams, John SJ Anderson, Karen Etheridge, Karl Williamson, Renee Baecker, Sawyer X, Steve Hay, Todd Rinaldo, Tony Cook, Yves Orton, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5260delta − what is new for perl v5.26.0

## DESCRIPTION

This document describes the differences between the 5.24.0 release and the 5.26.0 release.

## Notice

This release includes three updates with widespread effects:

- `"."` no longer in `@INC`

  For security reasons, the current directory (`"."`) is no longer included by default at the end of the module search path (`@INC`). This may have widespread implications for the building, testing and installing of modules, and for the execution of scripts. See the section "Removal of the current directory (`"."`) from `@INC`" for the full details.

- `do` may now warn

  `do` now gives a deprecation warning when it fails to load a file which it would have loaded had `"."` been in `@INC`.

- In regular expression patterns, a literal left brace `"{"` should be escaped

  See "Unescaped literal `"{"` characters in regular expression patterns are no longer permissible".

## Core Enhancements

### Lexical subroutines are no longer experimental

Using the `lexical_subs` feature introduced in v5.18 no longer emits a warning. Existing code that disables the `experimental::lexical_subs` warning category that the feature previously used will continue to work. The `lexical_subs` feature has no effect; all Perl code can use lexical subroutines, regardless of what feature declarations are in scope.

### Indented Here-documents

This adds a new modifier `"~"` to here-docs that tells the parser that it should look for `/^\s*$DELIM\n/` as the closing delimiter.

These syntaxes are all supported:

```
<<~EOF;
<<~\EOF;
<<~'EOF';
<<~"EOF";
<<~`EOF`;
<<~ 'EOF';
<<~ "EOF";
<<~ `EOF`;
```

The `"~"` modifier will strip, from each line in the here-doc, the same whitespace that appears before the delimiter.

Newlines will be copied as-is, and lines that don't include the proper beginning whitespace will cause perl to croak.

For example:

```
if (1) {
  print <<~EOF;
    Hello there
    EOF
}
```

prints ‘‘Hello there\n’’ with no leading whitespace.

### New regular expression modifier `/xx`

Specifying two `"x"` characters to modify a regular expression pattern does everything that a single one does, but additionally TAB and SPACE characters within a bracketed character class are generally ignored and can be added to improve readability, like `/[ ^ A-Z d-f p-x ]/xx`. Details are at ‘‘/x and /xx’’ in perlre.

#### `@{^CAPTURE}`, `%{^CAPTURE}`, and `%{^CAPTURE_ALL}`

`@{^CAPTURE}` exposes the capture buffers of the last match as an array. So `$1` is `${^CAPTURE}[0]`. This is a more efficient equivalent to code like `substr($matched_string,$-[0],$+[0]-$-[0])`, and you don't have to keep track of the `$matched_string` either. This variable has no single character equivalent. Note that, like the other regex magic variables, the contents of this variable is dynamic; if you wish to store it beyond the lifetime of the match you must copy it to another array.

`%{^CAPTURE}` is equivalent to `%+` (*i.e.*, named captures). Other than being more self-documenting there is no difference between the two forms.

`%{^CAPTURE_ALL}` is equivalent to `%-` (*i.e.*, all named captures). Other than being more self-documenting there is no difference between the two forms.

#### Declaring a reference to a variable

As an experimental feature, Perl now allows the referencing operator to come after `my()`, `state()`, `our()`, or `local()`. This syntax must be enabled with `use feature 'declared_refs'`. It is experimental, and will warn by default unless `no warnings 'experimental::refaliasing'` is in effect. It is intended mainly for use in assignments to references. For example:

```
use experimental 'refaliasing', 'declared_refs';
my \$a = \$b;
```

See ''Assigning to References'' in perlref for more details.

#### Unicode 9.0 is now supported

A list of changes is at <http://www.unicode.org/versions/Unicode9.0.0/>. Modules that are shipped with core Perl but not maintained by p5p do not necessarily support Unicode 9.0. Unicode::Normalize does work on 9.0.

#### Use of `\p{`*`script`*`}` uses the improved Script_Extensions property

Unicode 6.0 introduced an improved form of the Script (`sc`) property, and called it Script_Extensions (`scx`). Perl now uses this improved version when a property is specified as just `\p{`*`script`*`}`. This should make programs more accurate when determining if a character is used in a given script, but there is a slight chance of breakage for programs that very specifically needed the old behavior. The meaning of compound forms, like `\p{sc=`*`script`*`}` are unchanged. See ''Scripts'' in perlunicode.

#### Perl can now do default collation in UTF-8 locales on platforms that support it

Some platforms natively do a reasonable job of collating and sorting in UTF-8 locales. Perl now works with those. For portability and full control, Unicode::Collate is still recommended, but now you may not need to do anything special to get good-enough results, depending on your application. See "Category `LC_COLLATE`: Collation: Text Comparisons and Sorting" in perllocale.

#### Better locale collation of strings containing embedded `NUL` characters

In locales that have multi-level character weights, `NUL`s are now ignored at the higher priority ones. There are still some gotchas in some strings, though. See "Collation of strings containing embedded `NUL` characters" in perllocale.

#### `CORE` subroutines for hash and array functions callable via reference

The hash and array functions in the `CORE` namespace (`keys`, `each`, `values`, `push`, `pop`, `shift`, `unshift` and `splice`) can now be called with ampersand syntax (`&CORE::keys(\%hash)` and via reference (`my $k = \&CORE::keys; $k->(\%hash)`). Previously they could only be used when inlined.

#### New Hash Function For 64-bit Builds

We have switched to a hybrid hash function to better balance performance for short and long keys.

For short keys, 16 bytes and under, we use an optimised variant of One At A Time Hard, and for longer keys we use Siphash 1-3. For very long keys this is a big improvement in performance. For shorter keys there is a modest improvement.

## Security

#### Removal of the current directory (''.'') from `@INC`

The perl binary includes a default set of paths in `@INC`. Historically it has also included the current directory (`"."`) as the final entry, unless run with taint mode enabled (`perl -T`). While convenient,

this has security implications: for example, where a script attempts to load an optional module when its current directory is untrusted (such as */tmp*), it could load and execute code from under that directory.

Starting with v5.26, `"."` is always removed by default, not just under tainting. This has major implications for installing modules and executing scripts.

The following new features have been added to help ameliorate these issues.

- *Configure -Udefault_inc_excludes_dot*

  There is a new *Configure* option, `default_inc_excludes_dot` (enabled by default) which builds a perl executable without `"."`; unsetting this option using `-U` reverts perl to the old behaviour. This may fix your path issues but will reintroduce all the security concerns, so don't build a perl executable like this unless you're *really* confident that such issues are not a concern in your environment.

- `PERL_USE_UNSAFE_INC`

  There is a new environment variable recognised by the perl interpreter. If this variable has the value 1 when the perl interpreter starts up, then `"."` will be automatically appended to `@INC` (except under tainting).

  This allows you restore the old perl interpreter behaviour on a case-by-case basis. But note that this is intended to be a temporary crutch, and this feature will likely be removed in some future perl version. It is currently set by the `cpan` utility and `Test::Harness` to ease installation of CPAN modules which have not been updated to handle the lack of dot. Once again, don't use this unless you are sure that this will not reintroduce any security concerns.

- A new deprecation warning issued by `do`.

  While it is well-known that `use` and `require` use `@INC` to search for the file to load, many people don't realise that `do "file"` also searches `@INC` if the file is a relative path. With the removal of `"."`, a simple `do "file.pl"` will fail to read in and execute `file.pl` from the current directory. Since this is commonly expected behaviour, a new deprecation warning is now issued whenever `do` fails to load a file which it otherwise would have found if a dot had been in `@INC`.

Here are some things script and module authors may need to do to make their software work in the new regime.

- Script authors

  If the issue is within your own code (rather than within included modules), then you have two main options. Firstly, if you are confident that your script will only be run within a trusted directory (under which you expect to find trusted files and modules), then add `"."` back into the path; *e.g.*:

```
BEGIN {
    my $dir = "/some/trusted/directory";
    chdir $dir or die "Can't chdir to $dir: $!\n";
    # safe now
    push @INC, '.';
}

use "Foo::Bar"; # may load /some/trusted/directory/Foo/Bar.pm
do "config.pl"; # may load /some/trusted/directory/config.pl
```

  On the other hand, if your script is intended to be run from within untrusted directories (such as */tmp*), then your script suddenly failing to load files may be indicative of a security issue. You most likely want to replace any relative paths with full paths; for example,

```
do "foo_config.pl"
```

  might become

```
do "$ENV{HOME}/foo_config.pl"
```

  If you are absolutely certain that you want your script to load and execute a file from the current

directory, then use a `./` prefix; for example:

```
do "./foo_config.pl"
```

- Installing and using CPAN modules

  If you install a CPAN module using an automatic tool like `cpan`, then this tool will itself set the `PERL_USE_UNSAFE_INC` environment variable while building and testing the module, which may be sufficient to install a distribution which hasn't been updated to be dot-aware. If you want to install such a module manually, then you'll need to replace the traditional invocation:

  ```
  perl Makefile.PL && make && make test && make install
  ```

  with something like

  ```
  (export PERL_USE_UNSAFE_INC=1; \
   perl Makefile.PL && make && make test && make install)
  ```

  Note that this only helps build and install an unfixed module. It's possible for the tests to pass (since they were run under `PERL_USE_UNSAFE_INC=1`), but for the module itself to fail to perform correctly in production. In this case, you may have to temporarily modify your script until a fixed version of the module is released. For example:

  ```
  use Foo::Bar;
  {
      local @INC = (@INC, '.');
      # assuming read_config() needs '.' in @INC
      $config = Foo::Bar->read_config();
  }
  ```

  This is only rarely expected to be necessary. Again, if doing this, assess the resultant risks first.

- Module Authors

  If you maintain a CPAN distribution, it may need updating to run in a dotless environment. Although `cpan` and other such tools will currently set the `PERL_USE_UNSAFE_INC` during module build, this is a temporary workaround for the set of modules which rely on `"."` being in `@INC` for installation and testing, and this may mask deeper issues. It could result in a module which passes tests and installs, but which fails at run time.

  During build, test, and install, it will normally be the case that any perl processes will be executing directly within the root directory of the untarred distribution, or a known subdirectory of that, such as *t/*. It may well be that *Makefile.PL* or *t/foo.t* will attempt to include local modules and configuration files using their direct relative filenames, which will now fail.

  However, as described above, automatic tools like *cpan* will (for now) set the `PERL_USE_UNSAFE_INC` environment variable, which introduces dot during a build.

  This makes it likely that your existing build and test code will work, but this may mask issues with your code which only manifest when used after install. It is prudent to try and run your build process with that variable explicitly disabled:

  ```
  (export PERL_USE_UNSAFE_INC=0; \
   perl Makefile.PL && make && make test && make install)
  ```

  This is more likely to show up any potential problems with your module's build process, or even with the module itself. Fixing such issues will ensure both that your module can again be installed manually, and that it will still build once the `PERL_USE_UNSAFE_INC` crutch goes away.

  When fixing issues in tests due to the removal of dot from `@INC`, reinsertion of dot into `@INC` should be performed with caution, for this too may suppress real errors in your runtime code. You are encouraged wherever possible to apply the aforementioned approaches with explicit absolute/relative paths, or to relocate your needed files into a subdirectory and insert that subdirectory into `@INC` instead.

  If your runtime code has problems under the dotless `@INC`, then the comments above on how to fix for script authors will mostly apply here too. Bear in mind though that it is considered bad form for a module to globally add a dot to `@INC`, since it introduces both a security risk and hides

issues of accidentally requiring dot in `@INC`, as explained above.

### Escaped colons and relative paths in `PATH`

On Unix systems, Perl treats any relative paths in the `PATH` environment variable as tainted when starting a new process. Previously, it was allowing a backslash to escape a colon (unlike the OS), consequently allowing relative paths to be considered safe if the PATH was set to something like `/\:.`. The check has been fixed to treat `"."` as tainted in that example.

### New `-Di` switch is now required for PerlIO debugging output

This is used for debugging of code within PerlIO to avoid recursive calls. Previously this output would be sent to the file specified by the `PERLIO_DEBUG` environment variable if perl wasn't running setuid and the `-T` or `-t` switches hadn't been parsed yet.

If perl performed output at a point where it hadn't yet parsed its switches this could result in perl creating or overwriting the file named by `PERLIO_DEBUG` even when the `-T` switch had been supplied.

Perl now requires the `-Di` switch to be present before it will produce PerlIO debugging output. By default this is written to `stderr`, but can optionally be redirected to a file by setting the `PERLIO_DEBUG` environment variable.

If perl is running setuid or the `-T` switch was supplied, `PERLIO_DEBUG` is ignored and the debugging output is sent to `stderr` as for any other `-D` switch.

## Incompatible Changes

### Unescaped literal ''{'' characters in regular expression patterns are no longer permissible

You have to now say something like `"\{"` or `"[{]"` to specify to match a LEFT CURLY BRACKET; otherwise, it is a fatal pattern compilation error. This change will allow future extensions to the language.

These have been deprecated since v5.16, with a deprecation message raised for some uses starting in v5.22. Unfortunately, the code added to raise the message was buggy and failed to warn in some cases where it should have. Therefore, enforcement of this ban for these cases is deferred until Perl 5.30, but the code has been fixed to raise a default-on deprecation message for them in the meantime.

Some uses of literal `"{"` occur in contexts where we do not foresee the meaning ever being anything but the literal, such as the very first character in the pattern, or after a `"|"` meaning alternation. Thus

```
qr/{fee|{fie/
```

matches either of the strings `{fee` or `{fie`. To avoid forcing unnecessary code changes, these uses do not need to be escaped, and no warning is raised about them, and there are no current plans to change this.

But it is always correct to escape `"{"`, and the simple rule to remember is to always do so.

See Unescaped left brace in regex is illegal here.

### `scalar(%hash)` return signature changed

The value returned for `scalar(%hash)` will no longer show information about the buckets allocated in the hash. It will simply return the count of used keys. It is thus equivalent to `0+keys(%hash)`.

A form of backward compatibility is provided via `Hash::Util::bucket_ratio()` which provides the same behavior as `scalar(%hash)` provided in Perl 5.24 and earlier.

### `keys` returned from an lvalue subroutine

`keys` returned from an lvalue subroutine can no longer be assigned to in list context.

```
sub foo : lvalue { keys(%INC) }
(foo) = 3; # death
sub bar : lvalue { keys(@_) }
(bar) = 3; # also an error
```

This makes the lvalue sub case consistent with `(keys %hash) = ...` and `(keys @_) = ...`, which are also errors. [GH #15339] <https://github.com/Perl/perl5/issues/15339>

### The `${^ENCODING}` facility has been removed

The special behaviour associated with assigning a value to this variable has been removed. As a consequence, the encoding pragma's default mode is no longer supported. If you still need to write your source code in encodings other than UTF-8, use a source filter such as Filter::Encoding on CPAN or encoding's `Filter` option.

### `POSIX::tmpnam()` has been removed

The fundamentally unsafe `tmpnam()` interface was deprecated in Perl 5.22 and has now been removed. In its place, you can use, for example, the File::Temp interfaces.

### require ::Foo::Bar is now illegal.

Formerly, `require ::Foo::Bar` would try to read */Foo/Bar.pm*. Now any bareword require which starts with a double colon dies instead.

### Literal control character variable names are no longer permissible

A variable name may no longer contain a literal control character under any circumstances. These previously were allowed in single-character names on ASCII platforms, but have been deprecated there since Perl 5.20. This affects things like `$\cT`, where *\cT* is a literal control (such as a `NAK` or `NEGATIVE ACKNOWLEDGE` character) in the source code.

### `NBSP` is no longer permissible in `\N{...}`

The name of a character may no longer contain non-breaking spaces. It has been deprecated to do so since Perl 5.22.

## Deprecations

### String delimiters that aren't stand-alone graphemes are now deprecated

For Perl to eventually allow string delimiters to be Unicode grapheme clusters (which look like a single character, but may be a sequence of several ones), we have to stop allowing a single character delimiter that isn't a grapheme by itself. These are unlikely to exist in actual code, as they would typically display as attached to the character in front of them.

### `\c`*X* that maps to a printable is no longer deprecated

This means we have no plans to remove this feature. It still raises a warning, but only if syntax warnings are enabled. The feature was originally intended to be a way to express non-printable characters that don't have a mnemonic (`\t` and `\n` are mnemonics for two non-printable characters, but most non-printables don't have a mnemonic.) But the feature can be used to specify a few printable characters, though those are more clearly expressed as the printable itself. See <http://www.nntp.perl.org/group/perl.perl5.porters/2017/02/msg242944.html>.

## Performance Enhancements

- A hash in boolean context is now sometimes faster, *e.g.*

  ```
  if (!%h) { ... }
  ```

  This was already special-cased, but some cases were missed (such as `grep %$_, @AoH`), and even the ones which weren't have been improved.

- New Faster Hash Function on 64 bit builds

  We use a different hash function for short and long keys. This should improve performance and security, especially for long keys.

- readline is faster

  Reading from a file line-by-line with `readline()` or `<>` should now typically be faster due to a better implementation of the code that searches for the next newline character.

- Assigning one reference to another, *e.g.* `$ref1 = $ref2` has been optimized in some cases.

- Remove some exceptions to creating Copy-on-Write strings. The string buffer growth algorithm has been slightly altered so that you're less likely to encounter a string which can't be COWed.

- Better optimise array and hash assignment: where an array or hash appears in the LHS of a list assignment, such as `(..., @a) = (...);`, it's likely to be considerably faster, especially if it involves emptying the array/hash. For example, this code runs about a third faster compared to Perl 5.24.0:

```
my @a;
for my $i (1..10_000_000) {
    @a = (1,2,3);
    @a = ();
}
```

- Converting a single-digit string to a number is now substantially faster.
- The `split` builtin is now slightly faster in many cases: in particular for the two specially-handled forms

  ```
  my    @a = split ...;
  local @a = split ...;
  ```

- The rather slow implementation for the experimental subroutine signatures feature has been made much faster; it is now comparable in speed with the traditional `my ($a, $b, @c) = @_`.
- Bareword constant strings are now permitted to take part in constant folding. They were originally exempted from constant folding in August 1999, during the development of Perl 5.6, to ensure that `use strict "subs"` would still apply to bareword constants. That has now been accomplished a different way, so barewords, like other constants, now gain the performance benefits of constant folding.

  This also means that void-context warnings on constant expressions of barewords now report the folded constant operand, rather than the operation; this matches the behaviour for non-bareword constants.

## Modules and Pragmata

### Updated Modules and Pragmata

- IO::Compress has been upgraded from version 2.069 to 2.074.
- Archive::Tar has been upgraded from version 2.04 to 2.24.
- arybase has been upgraded from version 0.11 to 0.12.
- attributes has been upgraded from version 0.27 to 0.29.

  The deprecation message for the `:unique` and `:locked` attributes now mention that they will disappear in Perl 5.28.
- B has been upgraded from version 1.62 to 1.68.
- B::Concise has been upgraded from version 0.996 to 0.999.

  Its output is now more descriptive for `op_private` flags.
- B::Debug has been upgraded from version 1.23 to 1.24.
- B::Deparse has been upgraded from version 1.37 to 1.40.
- B::Xref has been upgraded from version 1.05 to 1.06.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- base has been upgraded from version 2.23 to 2.25.
- bignum has been upgraded from version 0.42 to 0.47.
- Carp has been upgraded from version 1.40 to 1.42.
- charnames has been upgraded from version 1.43 to 1.44.
- Compress::Raw::Bzip2 has been upgraded from version 2.069 to 2.074.
- Compress::Raw::Zlib has been upgraded from version 2.069 to 2.074.
- Config::Perl::V has been upgraded from version 0.25 to 0.28.
- CPAN has been upgraded from version 2.11 to 2.18.
- CPAN::Meta has been upgraded from version 2.150005 to 2.150010.

- Data::Dumper has been upgraded from version 2.160 to 2.167.

  The XS implementation now supports Deparse.

- DB_File has been upgraded from version 1.835 to 1.840.
- Devel::Peek has been upgraded from version 1.23 to 1.26.
- Devel::PPPort has been upgraded from version 3.32 to 3.35.
- Devel::SelfStubber has been upgraded from version 1.05 to 1.06.

  It now uses 3-arg `open()` instead of 2-arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>

- diagnostics has been upgraded from version 1.34 to 1.36.

  It now uses 3-arg `open()` instead of 2-arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>

- Digest has been upgraded from version 1.17 to 1.17_01.
- Digest::MD5 has been upgraded from version 2.54 to 2.55.
- Digest::SHA has been upgraded from version 5.95 to 5.96.
- DynaLoader has been upgraded from version 1.38 to 1.42.
- Encode has been upgraded from version 2.80 to 2.88.
- encoding has been upgraded from version 2.17 to 2.19.

  This module's default mode is no longer supported. It now dies when imported, unless the `Filter` option is being used.

- encoding::warnings has been upgraded from version 0.12 to 0.13.

  This module is no longer supported. It emits a warning to that effect and then does nothing.

- Errno has been upgraded from version 1.25 to 1.28.

  It now documents that using `%!` automatically loads Errno for you.

  It now uses 3-arg `open()` instead of 2-arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>

- ExtUtils::Embed has been upgraded from version 1.33 to 1.34.

  It now uses 3-arg `open()` instead of 2-arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>

- ExtUtils::MakeMaker has been upgraded from version 7.10_01 to 7.24.
- ExtUtils::Miniperl has been upgraded from version 1.05 to 1.06.
- ExtUtils::ParseXS has been upgraded from version 3.31 to 3.34.
- ExtUtils::Typemaps has been upgraded from version 3.31 to 3.34.
- feature has been upgraded from version 1.42 to 1.47.
- File::Copy has been upgraded from version 2.31 to 2.32.
- File::Fetch has been upgraded from version 0.48 to 0.52.
- File::Glob has been upgraded from version 1.26 to 1.28.

  It now Issues a deprecation message for `File::Glob::glob()`.

- File::Spec has been upgraded from version 3.63 to 3.67.
- FileHandle has been upgraded from version 2.02 to 2.03.
- Filter::Simple has been upgraded from version 0.92 to 0.93.

  It no longer treats `no MyFilter` immediately following `use MyFilter` as end-of-file. [GH #11853] <https://github.com/Perl/perl5/issues/11853>

- Getopt::Long has been upgraded from version 2.48 to 2.49.
- Getopt::Std has been upgraded from version 1.11 to 1.12.
- Hash::Util has been upgraded from version 0.19 to 0.22.
- HTTP::Tiny has been upgraded from version 0.056 to 0.070.

  Internal 599−series errors now include the redirect history.
- I18N::LangTags has been upgraded from version 0.40 to 0.42.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- IO has been upgraded from version 1.36 to 1.38.
- IO::Socket::IP has been upgraded from version 0.37 to 0.38.
- IPC::Cmd has been upgraded from version 0.92 to 0.96.
- IPC::SysV has been upgraded from version 2.06_01 to 2.07.
- JSON::PP has been upgraded from version 2.27300 to 2.27400_02.
- lib has been upgraded from version 0.63 to 0.64.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- List::Util has been upgraded from version 1.42_02 to 1.46_02.
- Locale::Codes has been upgraded from version 3.37 to 3.42.
- Locale::Maketext has been upgraded from version 1.26 to 1.28.
- Locale::Maketext::Simple has been upgraded from version 0.21 to 0.21_01.
- Math::BigInt has been upgraded from version 1.999715 to 1.999806.
- Math::BigInt::FastCalc has been upgraded from version 0.40 to 0.5005.
- Math::BigRat has been upgraded from version 0.260802 to 0.2611.
- Math::Complex has been upgraded from version 1.59 to 1.5901.
- Memoize has been upgraded from version 1.03 to 1.03_01.
- Module::CoreList has been upgraded from version 5.20170420 to 5.20170530.
- Module::Load::Conditional has been upgraded from version 0.64 to 0.68.
- Module::Metadata has been upgraded from version 1.000031 to 1.000033.
- mro has been upgraded from version 1.18 to 1.20.
- Net::Ping has been upgraded from version 2.43 to 2.55.

  IPv6 addresses and `AF_INET6` sockets are now supported, along with several other enhancements.
- NEXT has been upgraded from version 0.65 to 0.67.
- Opcode has been upgraded from version 1.34 to 1.39.
- open has been upgraded from version 1.10 to 1.11.
- OS2::Process has been upgraded from version 1.11 to 1.12.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- overload has been upgraded from version 1.26 to 1.28.

  Its compilation speed has been improved slightly.
- parent has been upgraded from version 0.234 to 0.236.

- perl5db.pl has been upgraded from version 1.50 to 1.51.

  It now ignores */dev/tty* on non-Unix systems. [GH #12244] <https://github.com/Perl/perl5/issues/12244>

- Perl::OSType has been upgraded from version 1.009 to 1.010.
- perlfaq has been upgraded from version 5.021010 to 5.021011.
- PerlIO has been upgraded from version 1.09 to 1.10.
- PerlIO::encoding has been upgraded from version 0.24 to 0.25.
- PerlIO::scalar has been upgraded from version 0.24 to 0.26.
- Pod::Checker has been upgraded from version 1.60 to 1.73.
- Pod::Functions has been upgraded from version 1.10 to 1.11.
- Pod::Html has been upgraded from version 1.22 to 1.2202.
- Pod::Perldoc has been upgraded from version 3.25_02 to 3.28.
- Pod::Simple has been upgraded from version 3.32 to 3.35.
- Pod::Usage has been upgraded from version 1.68 to 1.69.
- POSIX has been upgraded from version 1.65 to 1.76.

  This remedies several defects in making its symbols exportable. [GH #15260] <https://github.com/Perl/perl5/issues/15260>

  The `POSIX::tmpnam()` interface has been removed, see ‘‘**POSIX::tmpnam()** has been removed’’.

  The following deprecated functions have been removed:

  ```
  POSIX::isalnum
  POSIX::isalpha
  POSIX::iscntrl
  POSIX::isdigit
  POSIX::isgraph
  POSIX::islower
  POSIX::isprint
  POSIX::ispunct
  POSIX::isspace
  POSIX::isupper
  POSIX::isxdigit
  POSIX::tolower
  POSIX::toupper
  ```

  Trying to import POSIX subs that have no real implementations (like `POSIX::atend()`) now fails at import time, instead of waiting until runtime.

- re has been upgraded from version 0.32 to 0.34

  This adds support for the new `/xx` regular expression pattern modifier, and a change to the `use re 'strict'` experimental feature. When `re 'strict'` is enabled, a warning now will be generated for all unescaped uses of the two characters `"}"` and `"]"` in regular expression patterns (outside bracketed character classes) that are taken literally. This brings them more in line with the `")"` character which is always a metacharacter unless escaped. Being a metacharacter only sometimes, depending on an action at a distance, can lead to silently having the pattern mean something quite different than was intended, which the `re 'strict'` mode is intended to minimize.

- Safe has been upgraded from version 2.39 to 2.40.
- Scalar::Util has been upgraded from version 1.42_02 to 1.46_02.
- Storable has been upgraded from version 2.56 to 2.62.

  Fixes [GH #15714] <https://github.com/Perl/perl5/issues/15714>.

- Symbol has been upgraded from version 1.07 to 1.08.
- Sys::Syslog has been upgraded from version 0.33 to 0.35.
- Term::ANSIColor has been upgraded from version 4.04 to 4.06.
- Term::ReadLine has been upgraded from version 1.15 to 1.16.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- Test has been upgraded from version 1.28 to 1.30.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- Test::Harness has been upgraded from version 3.36 to 3.38.
- Test::Simple has been upgraded from version 1.001014 to 1.302073.
- Thread::Queue has been upgraded from version 3.09 to 3.12.
- Thread::Semaphore has been upgraded from 2.12 to 2.13.

  Added the `down_timed` method.
- threads has been upgraded from version 2.07 to 2.15.
- threads::shared has been upgraded from version 1.51 to 1.56.
- Tie::Hash::NamedCapture has been upgraded from version 0.09 to 0.10.
- Time::HiRes has been upgraded from version 1.9733 to 1.9741.

  It now builds on systems with C++11 compilers (such as G++ 6 and Clang++ 3.9).

  Now uses `clockid_t`.
- Time::Local has been upgraded from version 1.2300 to 1.25.
- Unicode::Collate has been upgraded from version 1.14 to 1.19.
- Unicode::UCD has been upgraded from version 0.64 to 0.68.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- version has been upgraded from version 0.9916 to 0.9917.
- VMS::DCLsym has been upgraded from version 1.06 to 1.08.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>
- warnings has been upgraded from version 1.36 to 1.37.
- XS::Typemap has been upgraded from version 0.14 to 0.15.
- XSLoader has been upgraded from version 0.21 to 0.27.

  Fixed a security hole in which binary files could be loaded from a path outside of `@INC`.

  It now uses 3−arg `open()` instead of 2−arg `open()`. [GH #15721] <https://github.com/Perl/perl5/issues/15721>

## Documentation

### New Documentation

*perldeprecation*

This file documents all upcoming deprecations, and some of the deprecations which already have been removed. The purpose of this documentation is two-fold: document what will disappear, and by which version, and serve as a guide for people dealing with code which has features that no longer work after an upgrade of their perl.

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, send email to perlbug@perl.org <mailto:perlbug@perl.org>.

Additionally, all references to Usenet have been removed, and the following selected changes have been made:

*perlfunc*

- Removed obsolete text about `defined()` on aggregates that should have been deleted earlier, when the feature was removed.
- Corrected documentation of `eval()`, and `evalbytes()`.
- Clarified documentation of `seek()`, `tell()` and `sysseek()` emphasizing that positions are in bytes and not characters. [GH #15438] <https://github.com/Perl/perl5/issues/15438>
- Clarified documentation of `sort()` concerning the variables `$a` and `$b`.
- In `split()` noted that certain pattern modifiers are legal, and added a caution about its use in Perls before v5.11.
- Removed obsolete documentation of `study()`, noting that it is now a no-op.
- Noted that `vec()` doesn't work well when the string contains characters whose code points are above 255.

*perlguts*

- Added advice on formatted printing of operands of `Size_t` and `SSize_t`

*perlhack*

- Clarify what editor tab stop rules to use, and note that we are migrating away from using tabs, replacing them with sequences of SPACE characters.

*perlhacktips*

- Give another reason to use `cBOOL` to cast an expression to boolean.
- Note that the macros `TRUE` and `FALSE` are available to express boolean values.

*perlinterp*

- perlinterp has been expanded to give a more detailed example of how to hunt around in the parser for how a given operator is handled.

*perllocale*

- Some locales aren't compatible with Perl. Note that these can cause core dumps.

*perlmod*

- Various clarifications have been added.

*perlmodlib*

- Updated the site mirror list.

*perlobj*

- Added a section on calling methods using their fully qualified names.
- Do not discourage manual `@ISA`.

*perlootut*

- Mention `Moo` more.

*perlop*

- Note that white space must be used for quoting operators if the delimiter is a word character (*i.e.*, matches `\w`).
- Clarify that in regular expression patterns delimited by single quotes, no variable interpolation is done.

*perlre*

- The first part was extensively rewritten to incorporate various basic points, that in earlier versions were mentioned in sort of an appendix on Version 8 regular expressions.

- Note that it is common to have the `/x` modifier and forget that this means that `"#"` has to be escaped.

*perlretut*

- Add introductory material.
- Note that a metacharacter occurring in a context where it can't mean that, silently loses its meta-ness and matches literally. `use re 'strict'` can catch some of these.

*perlunicode*

- Corrected the text about Unicode BYTE ORDER MARK handling.
- Updated the text to correspond with changes in Unicode UTS#18, concerning regular expressions, and Perl compatibility with what it says.

*perlvar*

- Document `@ISA`. It was documented in other places, but not in perlvar.

## Diagnostics

### New Diagnostics

*New Errors*

- A signature parameter must start with `'$'`, `'@'` or `'%'`
- Bareword in require contains ‘‘%s’’
- Bareword in require maps to empty filename
- Bareword in require maps to disallowed filename ‘‘%s’’
- Bareword in require must not start with a double-colon: ‘‘%s’’
- `%s:` command not found

  (A) You've accidentally run your script through **bash** or another shell instead of Perl. Check the `#!` line, or manually feed your script into Perl yourself. The `#!` line at the top of your file could look like:

  ```
  #!/usr/bin/perl
  ```

- `%s:` command not found: `%s`

  (A) You've accidentally run your script through **zsh** or another shell instead of Perl. Check the `#!` line, or manually feed your script into Perl yourself. The `#!` line at the top of your file could look like:

  ```
  #!/usr/bin/perl
  ```

- The experimental declared_refs feature is not enabled

  (F) To declare references to variables, as in `my \%x`, you must first enable the feature:

  ```
  no warnings "experimental::declared_refs";
  use feature "declared_refs";
  ```

  See ‘‘Declaring a reference to a variable’’.

- Illegal character following sigil in a subroutine signature
- Indentation on line `%d` of here-doc doesn't match delimiter
- Infinite recursion via empty pattern.

  Using the empty pattern (which re-executes the last successfully-matched pattern) inside a code block in another regex, as in `/(?{ s!!new! })/`, has always previously yielded a segfault. It now produces this error.

- Malformed UTF-8 string in ‘‘%s’’
- Multiple slurpy parameters not allowed

- '#' not allowed immediately following a sigil in a subroutine signature
- panic: unknown OA_*: %x
- Unescaped left brace in regex is illegal here

  Unescaped left braces are now illegal in some contexts in regular expression patterns. In other contexts, they are still just deprecated; they will be illegal in Perl 5.30.

- Version control conflict marker

  (F) The parser found a line starting with <<<<<<<, >>>>>>>, or =======. These may be left by a version control system to mark conflicts after a failed merge operation.

*New Warnings*

- Can't determine class of operator %s, assuming BASEOP
- Declaring references is experimental

  (S experimental::declared_refs) This warning is emitted if you use a reference constructor on the right-hand side of my(), state(), our(), or local(). Simply suppress the warning if you want to use the feature, but know that in doing so you are taking the risk of using an experimental feature which may change or be removed in a future Perl version:

  ```
  no warnings "experimental::declared_refs";
  use feature "declared_refs";
  $fooref = my \$foo;
  ```

  See ''Declaring a reference to a variable''.

- do ''%s'' failed, '.' is no longer in @INC

  Since "." is now removed from @INC by default, do will now trigger a warning recommending to fix the do statement.

- File::Glob::glob() will disappear in perl 5.30. Use File::Glob::bsd_glob() instead.
- Unescaped literal '%c' in regex; marked by <-- HERE in m/%s/
- Use of unassigned code point or non-standalone grapheme for a delimiter will be a fatal error starting in Perl 5.30

  See ''Deprecations''

### Changes to Existing Diagnostics

- When a require fails, we now do not provide @INC when the require is for a file instead of a module.
- When @INC is not scanned for a require call, we no longer display @INC to avoid confusion.
- Attribute ''locked'' is deprecated, and will disappear in Perl 5.28

  This existing warning has had the *and will disappear* text added in this release.

- Attribute ''unique'' is deprecated, and will disappear in Perl 5.28

  This existing warning has had the *and will disappear* text added in this release.

- Calling POSIX::%s() is deprecated

  This warning has been removed, as the deprecated functions have been removed from POSIX.

- Constants from lexical variables potentially modified elsewhere are deprecated. This will not be allowed in Perl 5.32

  This existing warning has had the *this will not be allowed* text added in this release.

- Deprecated use of my() in false conditional. This will be a fatal error in Perl 5.30

  This existing warning has had the *this will be a fatal error* text added in this release.

- `dump()` better written as `CORE::dump()`. `dump()` will no longer be available in Perl 5.30

  This existing warning has had the *no longer be available* text added in this release.

- Experimental `%s` on scalar is now forbidden

  This message is now followed by more helpful text. [GH #15291] <https://github.com/Perl/perl5/issues/15291>

- Experimental ‘‘%s’’ subs not enabled

  This warning was been removed, as lexical subs are no longer experimental.

- Having more than one /%c regexp modifier is deprecated

  This deprecation warning has been removed, since `/xx` now has a new meaning.

- %s() is deprecated on `:utf8` handles. This will be a fatal error in Perl 5.30 .

  where ‘‘%s’’ is one of `sysread`, `recv`, `syswrite`, or `send`.

  This existing warning has had the *this will be a fatal error* text added in this release.

  This warning is now enabled by default, as all `deprecated` category warnings should be.

- `$*` is no longer supported. Its use will be fatal in Perl 5.30

  This existing warning has had the *its use will be fatal* text added in this release.

- $# is no longer supported. Its use will be fatal in Perl 5.30

  This existing warning has had the *its use will be fatal* text added in this release.

- Malformed UTF-8 character%s

  Details as to the exact problem have been added at the end of this message

- Missing or undefined argument to `%s`

  This warning used to warn about `require`, even if it was actually `do` which being executed. It now gets the operation name right.

- NO-BREAK SPACE in a charnames alias definition is deprecated

  This warning has been removed as the behavior is now an error.

- Odd name/value argument for subroutine ’%s’

  This warning now includes the name of the offending subroutine.

- Opening dirhandle `%s` also as a file. This will be a fatal error in Perl 5.28

  This existing warning has had the *this will be a fatal error* text added in this release.

- Opening filehandle `%s` also as a directory. This will be a fatal error in Perl 5.28

  This existing warning has had the *this will be a fatal error* text added in this release.

- panic: ck_split, type=%u

  panic: pp_split, pm=%p, s=%p

  These panic errors have been removed.

- Passing malformed UTF-8 to ‘‘%s’’ is deprecated

  This warning has been changed to the fatal Malformed UTF-8 string in ‘‘%s’’

- Setting `$/` to a reference to `%s` as a form of slurp is deprecated, treating as undef. This will be fatal in Perl 5.28

  This existing warning has had the *this will be fatal* text added in this release.

- `${^ENCODING}` is no longer supported. Its use will be fatal in Perl 5.28

  This warning used to be: "Setting `${^ENCODING}` is deprecated".

  The special action of the variable `${^ENCODING}` was formerly used to implement the `encoding` pragma. As of Perl 5.26, rather than being deprecated, assigning to this variable now

has no effect except to issue the warning.

- Too few arguments for subroutine '%s'

  This warning now includes the name of the offending subroutine.

- Too many arguments for subroutine '%s'

  This warning now includes the name of the offending subroutine.

- Unescaped left brace in regex is deprecated here (and will be fatal in Perl 5.30), passed through in regex; marked by <-- HERE in m/%s/

  This existing warning has had the *here (and will be fatal...)* text added in this release.

- Unknown charname '' is deprecated. Its use will be fatal in Perl 5.28

  This existing warning has had the *its use will be fatal* text added in this release.

- Use of bare << to mean <<"" is deprecated. Its use will be fatal in Perl 5.28

  This existing warning has had the *its use will be fatal* text added in this release.

- Use of code point 0x%s is deprecated; the permissible max is 0x%s. This will be fatal in Perl 5.28

  This existing warning has had the *this will be fatal* text added in this release.

- Use of comma-less variable list is deprecated. Its use will be fatal in Perl 5.28

  This existing warning has had the *its use will be fatal* text added in this release.

- Use of inherited `AUTOLOAD` for non-method %s() is deprecated. This will be fatal in Perl 5.28

  This existing warning has had the *this will be fatal* text added in this release.

- Use of strings with code points over 0xFF as arguments to `%s` operator is deprecated. This will be a fatal error in Perl 5.28

  This existing warning has had the *this will be a fatal error* text added in this release.

## Utility Changes

*c2ph* **and** *pstruct*

- These old utilities have long since superceded by h2xs, and are now gone from the distribution.

*Porting/pod_lib.pl*

- Removed spurious executable bit.
- Account for the possibility of DOS file endings.

*Porting/sync-with-cpan*

- Many improvements.

*perf/benchmarks*

- Tidy file, rename some symbols.

*Porting/checkAUTHORS.pl*

- Replace obscure character range with `\w`.

*t/porting/regen.t*

- Try to be more helpful when tests fail.

*utils/h2xs.PL*

- Avoid infinite loop for enums.

### perlbug

- Long lines in the message body are now wrapped at 900 characters, to stay well within the 1000-character limit imposed by SMTP mail transfer agents. This is particularly likely to be important for the list of arguments to *Configure*, which can readily exceed the limit if, for example, it names several non-default installation paths. This change also adds the first unit tests for perlbug. [perl #128020] <https://rt.perl.org/Public/Bug/Display.html?id=128020>

## Configuration and Compilation

- `-Ddefault_inc_excludes_dot` has added, and enabled by default.

- The `dtrace` build process has further changes [GH #15718] <https://github.com/Perl/perl5/issues/15718>:
  - If the `-xnolibs` is available, use that so a *dtrace* perl can be built within a FreeBSD jail.
  - On systems that build a *dtrace* object file (FreeBSD, Solaris, and SystemTap's dtrace emulation), copy the input objects to a separate directory and process them there, and use those objects in the link, since `dtrace -G` also modifies these objects.
  - Add *libelf* to the build on FreeBSD 10.x, since *dtrace* adds references to *libelf* symbols.
  - Generate a dummy *dtrace_main.o* if `dtrace -G` fails to build it. A default build on Solaris generates probes from the unused inline functions, while they don't on FreeBSD, which causes `dtrace -G` to fail.
- You can now disable perl's use of the `PERL_HASH_SEED` and `PERL_PERTURB_KEYS` environment variables by configuring perl with `-Accflags=NO_PERL_HASH_ENV`.
- You can now disable perl's use of the `PERL_HASH_SEED_DEBUG` environment variable by configuring perl with `-Accflags=-DNO_PERL_HASH_SEED_DEBUG`.
- *Configure* now zeroes out the alignment bytes when calculating the bytes for 80-bit `NaN` and `Inf` to make builds more reproducible. [GH #15725] <https://github.com/Perl/perl5/issues/15725>
- Since v5.18, for testing purposes we have included support for building perl with a variety of non-standard, and non-recommended hash functions. Since we do not recommend the use of these functions, we have removed them and their corresponding build options. Specifically this includes the following build options:

  ```
  PERL_HASH_FUNC_SDBM
  PERL_HASH_FUNC_DJB2
  PERL_HASH_FUNC_SUPERFAST
  PERL_HASH_FUNC_MURMUR3
  PERL_HASH_FUNC_ONE_AT_A_TIME
  PERL_HASH_FUNC_ONE_AT_A_TIME_OLD
  PERL_HASH_FUNC_MURMUR_HASH_64A
  PERL_HASH_FUNC_MURMUR_HASH_64B
  ```

- Remove ''Warning: perl appears in your path''

  This install warning is more or less obsolete, since most platforms already **will** have a */usr/bin/perl* or similar provided by the OS.

- Reduce verbosity of `make install.man`

  Previously, two progress messages were emitted for each manpage: one by installman itself, and one by the function in *install_lib.pl* that it calls to actually install the file. Disabling the second of those in each case saves over 750 lines of unhelpful output.

- Cleanup for `clang -Weverything` support. [GH #15683] <https://github.com/Perl/perl5/issues/15683>
- *Configure*: signbit scan was assuming too much, stop assuming negative 0.
- Various compiler warnings have been silenced.
- Several smaller changes have been made to remove impediments to compiling under C++11.
- Builds using `USE_PAD_RESET` now work again; this configuration had bit-rotted.
- A probe for `gai_strerror` was added to *Configure* that checks if the `gai_strerror()` routine is available and can be used to translate error codes returned by `getaddrinfo()` into human readable strings.
- *Configure* now aborts if both `-Duselongdouble` and `-Dusequadmath` are requested. [GH #14944] <https://github.com/Perl/perl5/issues/14944>
- Fixed a bug in which *Configure* could append `-quadmath` to the archname even if it was already present. [GH #15423] <https://github.com/Perl/perl5/issues/15423>

- Clang builds with `-DPERL_GLOBAL_STRUCT` or `-DPERL_GLOBAL_STRUCT_PRIVATE` have been fixed (by disabling Thread Safety Analysis for these configurations).
- *make_ext.pl* no longer updates a module's *pm_to_blib* file when no files require updates. This could cause dependencies, *perlmain.c* in particular, to be rebuilt unnecessarily. [GH #15060] <https://github.com/Perl/perl5/issues/15060>
- The output of `perl -V` has been reformatted so that each configuration and compile-time option is now listed one per line, to improve readability.
- *Configure* now builds `miniperl` and `generate_uudmap` if you invoke it with `-Dusecrosscompiler` but not `-Dtargethost=somehost`. This means you can supply your target platform `config.sh`, generate the headers and proceed to build your cross-target perl. [GH #15126] <https://github.com/Perl/perl5/issues/15126>
- Perl built with `-Accflags=-DPERL_TRACE_OPS` now only dumps the operator counts when the environment variable `PERL_TRACE_OPS` is set to a non-zero integer. This allows `make test` to pass on such a build.
- When building with GCC 6 and link-time optimization (the `-flto` option to `gcc`), *Configure* was treating all probed symbols as present on the system, regardless of whether they actually exist. This has been fixed. [GH #15322] <https://github.com/Perl/perl5/issues/15322>
- The *t/test.pl* library is used for internal testing of Perl itself, and also copied by several CPAN modules. Some of those modules must work on older versions of Perl, so *t/test.pl* must in turn avoid newer Perl features. Compatibility with Perl 5.8 was inadvertently removed some time ago; it has now been restored. [GH #15302] <https://github.com/Perl/perl5/issues/15302>
- The build process no longer emits an extra blank line before building each ''simple'' extension (those with only **.pm* and **.pod* files).

## Testing

Tests were added and changed to reflect the other additions and changes in this release. Furthermore, these substantive changes were made:

- A new test script, *comp/parser_run.t*, has been added that is like *comp/parser.t* but with *test.pl* included so that `runperl()` and the like are available for use.
- Tests for locales were erroneously using locales incompatible with Perl.
- Some parts of the test suite that try to exhaustively test edge cases in the regex implementation have been restricted to running for a maximum of five minutes. On slow systems they could otherwise take several hours, without significantly improving our understanding of the correctness of the code under test.
- A new internal facility allows analysing the time taken by the individual tests in Perl's own test suite; see *Porting/harness-timer-report.pl*.
- *t/re/regexp_nonull.t* has been added to test that the regular expression engine can handle scalars that do not have a null byte just past the end of the string.
- A new test script, *t/op/decl-refs.t*, has been added to test the new feature ''Declaring a reference to a variable''.
- A new test script, *t/re/keep_tabs.t* has been added to contain tests where `\t` characters should not be expanded into spaces.
- A new test script, *t/re/anyof.t*, has been added to test that the ANYOF nodes generated by bracketed character classes are as expected.
- There is now more extensive testing of the Unicode-related API macros and functions.
- Several of the longer running API test files have been split into multiple test files so that they can be run in parallel.
- *t/harness* now tries really hard not to run tests which are located outside of the Perl source tree. [GH #14578] <https://github.com/Perl/perl5/issues/14578>
- Prevent debugger tests (*lib/perl5db.t*) from failing due to the contents of `$ENV{PERLDB_OPTS}`. [GH #15782] <https://github.com/Perl/perl5/issues/15782>

## Platform Support

### New Platforms

NetBSD/VAX

Perl now compiles under NetBSD on VAX machines. However, it's not possible for that platform to implement floating-point infinities and NaNs compatible with most modern systems, which implement the IEEE-754 floating point standard. The hexadecimal floating point (`0x...p[+-]n` literals, `printf %a`) is not implemented, either. The `make test` passes 98% of tests.

- Test fixes and minor updates.
- Account for lack of `inf`, `nan`, and `-0.0` support.

### Platform-Specific Notes

Darwin

- Don't treat `-Dprefix=/usr` as special: instead require an extra option `-Ddarwin_distribution` to produce the same results.
- OS X El Capitan doesn't implement the `clock_gettime()` or `clock_getres()` APIs; emulate them as necessary.
- Deprecated `syscall(2)` on macOS 10.12.

EBCDIC

Several tests have been updated to work (or be skipped) on EBCDIC platforms.

HP-UX

The Net::Ping UDP test is now skipped on HP-UX.

Hurd

The hints for Hurd have been improved, enabling malloc wrap and reporting the GNU libc used (previously it was an empty string when reported).

VAX

VAX floating point formats are now supported on NetBSD.

VMS

- The path separator for the `PERL5LIB` and `PERLLIB` environment entries is now a colon (`":"`) when running under a Unix shell. There is no change when running under DCL (it's still `"|"`).
- *configure.com* now recognizes the VSI-branded C compiler and no longer recognizes the ''DEC''-branded C compiler (as there hasn't been such a thing for 15 or more years).

Windows

- Support for compiling perl on Windows using Microsoft Visual Studio 2015 (containing Visual C++ 14.0) has been added.

  This version of VC++ includes a completely rewritten C run-time library, some of the changes in which mean that work done to resolve a socket `close()` bug in perl #120091 and perl #118059 is not workable in its current state with this version of VC++. Therefore, we have effectively reverted that bug fix for VS2015 onwards on the basis that being able to build with VS2015 onwards is more important than keeping the bug fix. We may revisit this in the future to attempt to fix the bug again in a way that is compatible with VS2015.

  These changes do not affect compilation with GCC or with Visual Studio versions up to and including VS2013, *i.e.*, the bug fix is retained (unchanged) for those compilers.

  Note that you may experience compatibility problems if you mix a perl built with GCC or VS <= VS2013 with XS modules built with VS2015, or if you mix a perl built with VS2015 with XS modules built with GCC or VS <= VS2013. Some incompatibility may arise because of the bug fix that has been reverted for VS2015 builds of perl, but there may well be incompatibility anyway because of the rewritten CRT in VS2015 (*e.g.*, see discussion at <http://stackoverflow.com/questions/30412951>).
- It now automatically detects GCC versus Visual C and sets the VC version number on Win32.

Linux

Drop support for Linux *a.out* executable format. Linux has used ELF for over twenty years.

OpenBSD 6

OpenBSD 6 still does not support returning `pid`, `gid`, or `uid` with `SA_SIGINFO`. Make sure to account for it.

FreeBSD

*t/uni/overload.t*: Skip hanging test on FreeBSD.

DragonFly BSD

DragonFly BSD now has support for `setproctitle()`. [GH #15703] <https://github.com/Perl/perl5/issues/15703>.

## Internal Changes

- A new API function `sv_setpv_bufsize()` allows simultaneously setting the length and the allocated size of the buffer in an SV, growing the buffer if necessary.
- A new API macro `SvPVCLEAR()` sets its SV argument to an empty string, like Perl-space `$x = ''`, but with several optimisations.
- Several new macros and functions for dealing with Unicode and UTF−8−encoded strings have been added to the API, as well as some changes in the functionality of existing functions (see ‘‘Unicode Support’’ in perlapi for more details):
  - New versions of the API macros like `isALPHA_utf8` and `toLOWER_utf8` have been added, each with the suffix `_safe`, like `isSPACE_utf8_safe`. These take an extra parameter, giving an upper limit of how far into the string it is safe to read. Using the old versions could cause attempts to read beyond the end of the input buffer if the UTF−8 is not well-formed, and their use now raises a deprecation warning. Details are at ‘‘Character classification’’ in perlapi.
  - Macros like `isALPHA_utf8` and `toLOWER_utf8` now die if they detect that their input UTF−8 is malformed. A deprecation warning had been issued since Perl 5.18.
  - Several new macros for analysing the validity of utf8 sequences. These are:

    `UTF8_GOT_ABOVE_31_BIT` `UTF8_GOT_CONTINUATION` `UTF8_GOT_EMPTY` `UTF8_GOT_LONG` `UTF8_GOT_NONCHAR` `UTF8_GOT_NON_CONTINUATION` `UTF8_GOT_OVERFLOW` `UTF8_GOT_SHORT` `UTF8_GOT_SUPER` `UTF8_GOT_SURROGATE` `UTF8_IS_INVARIANT` `UTF8_IS_NONCHAR` `UTF8_IS_SUPER` `UTF8_IS_SURROGATE` `UVCHR_IS_INVARIANT` `isUTF8_CHAR_flags` `isSTRICT_UTF8_CHAR` `isC9_STRICT_UTF8_CHAR`
  - Functions that are all extensions of the `is_utf8_string_*()` functions, that apply various restrictions to the UTF−8 recognized as valid:

    `is_strict_utf8_string`, `is_strict_utf8_string_loc`, `is_strict_utf8_string_loclen`,

    `is_c9strict_utf8_string`, `is_c9strict_utf8_string_loc`, `is_c9strict_utf8_string_loclen`,

    `is_utf8_string_flags`, `is_utf8_string_loc_flags`, `is_utf8_string_loclen_flags`,

    `is_utf8_fixed_width_buf_flags`, `is_utf8_fixed_width_buf_loc_flags`, `is_utf8_fixed_width_buf_loclen_flags`.

    `is_utf8_invariant_string`. `is_utf8_valid_partial_char`. `is_utf8_valid_partial_char_flags`.
  - The functions `utf8n_to_uvchr` and its derivatives have had several changes of behaviour.

    Calling them, while passing a string length of 0 is now asserted against in DEBUGGING builds, and otherwise, returns the Unicode REPLACEMENT CHARACTER. If you have nothing to decode, you shouldn’t call the decode function.

    They now return the Unicode REPLACEMENT CHARACTER if called with UTF−8 that has the overlong malformation and that malformation is allowed by the input parameters. This malformation is where the UTF−8 looks valid syntactically, but there is a shorter sequence

that yields the same code point. This has been forbidden since Unicode version 3.1.

They now accept an input flag to allow the overflow malformation. This malformation is when the UTF-8 may be syntactically valid, but the code point it represents is not capable of being represented in the word length on the platform. What ''allowed'' means, in this case, is that the function doesn't return an error, and it advances the parse pointer to beyond the UTF-8 in question, but it returns the Unicode REPLACEMENT CHARACTER as the value of the code point (since the real value is not representable).

They no longer abandon searching for other malformations when the first one is encountered. A call to one of these functions thus can generate multiple diagnostics, instead of just one.

- `valid_utf8_to_uvchr()` has been added to the API (although it was present in core earlier). Like `utf8_to_uvchr_buf()`, but assumes that the next character is well-formed. Use with caution.
- A new function, `utf8n_to_uvchr_error`, has been added for use by modules that need to know the details of UTF-8 malformations beyond pass/fail. Previously, the only ways to know why a sequence was ill-formed was to capture and parse the generated diagnostics or to do your own analysis.
- There is now a safer version of **utf8_hop()**, called `utf8_hop_safe()`. Unlike **utf8_hop()**, **utf8_hop_safe()** won't navigate before the beginning or after the end of the supplied buffer.
- Two new functions, `utf8_hop_forward()` and `utf8_hop_back()` are similar to `utf8_hop_safe()` but are for when you know which direction you wish to travel.
- Two new macros which return useful utf8 byte sequences:

  `BOM_UTF8`

  `REPLACEMENT_CHARACTER_UTF8`

- Perl is now built with the `PERL_OP_PARENT` compiler define enabled by default. To disable it, use the `PERL_NO_OP_PARENT` compiler define. This flag alters how the `op_sibling` field is used in `OP` structures, and has been available optionally since perl 5.22.

  See ''Internal Changes'' in perl5220delta for more details of what this build option does.

- Three new ops, `OP_ARGELEM`, `OP_ARGDEFELEM`, and `OP_ARGCHECK` have been added. These are intended principally to implement the individual elements of a subroutine signature, plus any overall checking required.
- The `OP_PUSHRE` op has been eliminated and the `OP_SPLIT` op has been changed from class `LISTOP` to `PMOP`.

  Formerly the first child of a split would be a `pushre`, which would have the `split`'s regex attached to it. Now the regex is attached directly to the `split` op, and the `pushre` has been eliminated.

- The `op_class()` API function has been added. This is like the existing `OP_CLASS()` macro, but can more accurately determine what struct an op has been allocated as. For example `OP_CLASS()` might return `OA_BASEOP_OR_UNOP` indicating that ops of this type are usually allocated as an `OP` or `UNOP`; while `op_class()` will return `OPclass_BASEOP` or `OPclass_UNOP` as appropriate.
- All parts of the internals now agree that the `sassign` op is a `BINOP`; previously it was listed as a `BASEOP` in *regen/opcodes*, which meant that several parts of the internals had to be special-cased to accommodate it. This oddity's original motivation was to handle code like `$x ||= 1`; that is now handled in a simpler way.
- The output format of the `op_dump()` function (as used by `perl -Dx`) has changed: it now displays an ''ASCII-art'' tree structure, and shows more low-level details about each op, such as its address and class.
- The `PADOFFSET` type has changed from being unsigned to signed, and several pad-related variables such as `PL_padix` have changed from being of type `I32` to type `PADOFFSET`.

- The `DEBUGGING`-mode output for regex compilation and execution has been enhanced.
- Several obscure SV flags have been eliminated, sometimes along with the macros which manipulate them: `SVpbm_VALID`, `SVpbm_TAIL`, `SvTAIL_on`, `SvTAIL_off`, `SVrepl_EVAL`, `SvEVALED`.
- An OP `op_private` flag has been eliminated: `OPpRUNTIME`. This used to often get set on `PMOP` ops, but had become meaningless over time.

## Selected Bug Fixes

- Perl no longer panics when switching into some locales on machines with buggy `strxfrm()` implementations in their *libc*. [GH #13768] <https://github.com/Perl/perl5/issues/13768>
- `$-{$name}` would leak an `AV` on each access if the regular expression had no named captures. The same applies to access to any hash tied with Tie::Hash::NamedCapture and `all => 1`. [GH #15882] <https://github.com/Perl/perl5/issues/15882>
- Attempting to use the deprecated variable `$#` as the object in an indirect object method call could cause a heap use after free or buffer overflow. [GH #15599] <https://github.com/Perl/perl5/issues/15599>
- When checking for an indirect object method call, in some rare cases the parser could reallocate the line buffer but then continue to use pointers to the old buffer. [GH #15585] <https://github.com/Perl/perl5/issues/15585>
- Supplying a glob as the format argument to `formline` would cause an assertion failure. [GH #15862] <https://github.com/Perl/perl5/issues/15862>
- Code like `$value1 =~ qr/.../ ~~ $value2` would have the match converted into a `qr//` operator, leaving extra elements on the stack to confuse any surrounding expression. [GH #15859] <https://github.com/Perl/perl5/issues/15859>
- Since v5.24 in some obscure cases, a regex which included code blocks from multiple sources (*e.g.*, via embedded via `qr//` objects) could end up with the wrong current pad and crash or give weird results. [GH #15657] <https://github.com/Perl/perl5/issues/15657>
- Occasionally `local()`s in a code block within a patterns weren't being undone when the pattern matching backtracked over the code block. [GH #15056] <https://github.com/Perl/perl5/issues/15056>
- Using `substr()` to modify a magic variable could access freed memory in some cases. [GH #15871] <https://github.com/Perl/perl5/issues/15871>
- Under `use utf8`, the entire source code is now checked for being UTF-8 well formed, not just quoted strings as before. [GH #14973] <https://github.com/Perl/perl5/issues/14973>.
- The range operator `".."` on strings now handles its arguments correctly when in the scope of the `unicode_strings` feature. The previous behaviour was sufficiently unexpected that we believe no correct program could have made use of it.
- The `split` operator did not ensure enough space was allocated for its return value in scalar context. It could then write a single pointer immediately beyond the end of the memory block allocated for the stack. [GH #15749] <https://github.com/Perl/perl5/issues/15749>
- Using a large code point with the `"W"` pack template character with the current output position aligned at just the right point could cause a write of a single zero byte immediately beyond the end of an allocated buffer. [GH #15572] <https://github.com/Perl/perl5/issues/15572>
- Supplying a format's picture argument as part of the format argument list where the picture specifies modifying the argument could cause an access to the new freed compiled format. [GH #15566] <https://github.com/Perl/perl5/issues/15566>
- The **sort()** operator's built-in numeric comparison function didn't handle large integers that weren't exactly representable by a double. This now uses the same code used to implement the `<=>` operator. [GH #15768] <https://github.com/Perl/perl5/issues/15768>
- Fix issues with `/(?{ ... <<EOF })/` that broke Method::Signatures. [GH #15779] <https://github.com/Perl/perl5/issues/15779>

- Fixed an assertion failure with `chop` and `chomp`, which could be triggered by `chop(@x =~ tr/1/1/)`. [GH #15738] <https://github.com/Perl/perl5/issues/15738>.
- Fixed a comment skipping error in patterns under `/x`; it could stop skipping a byte early, which could be in the middle of a UTF-8 character. [GH #15790] <https://github.com/Perl/perl5/issues/15790>.
- *perldb* now ignores */dev/tty* on non-Unix systems. [GH #12244] <https://github.com/Perl/perl5/issues/12244>;
- Fix assertion failure for `{}->$x` when `$x` isn't defined. [GH #15791] <https://github.com/Perl/perl5/issues/15791>.
- Fix an assertion error which could be triggered when a lookahead string in patterns exceeded a minimum length. [GH #15796] <https://github.com/Perl/perl5/issues/15796>.
- Only warn once per literal number about a misplaced `"_"`. [GH #9989] <https://github.com/Perl/perl5/issues/9989>.
- The `tr///` parse code could be looking at uninitialized data after a perse error. [GH #15624] <https://github.com/Perl/perl5/issues/15624>.
- In a pattern match, a back-reference (`\1`) to an unmatched capture could read back beyond the start of the string being matched. [GH #15634] <https://github.com/Perl/perl5/issues/15634>.
- `use re 'strict'` is supposed to warn if you use a range (such as `/(?[ [ X-Y ] ])/`) whose start and end digit aren't from the same group of 10. It didn't do that for five groups of mathematical digits starting at `U+1D7E`.
- A sub containing a ''forward'' declaration with the same name (*e.g.*, `sub c { sub c; }`) could sometimes crash or loop infinitely. [GH #15557] <https://github.com/Perl/perl5/issues/15557>
- A crash in executing a regex with a non-anchored UTF-8 substring against a target string that also used UTF-8 has been fixed. [GH #15628] <https://github.com/Perl/perl5/issues/15628>
- Previously, a shebang line like `#!perl -i u` could be erroneously interpreted as requesting the `-u` option. This has been fixed. [GH #15623] <https://github.com/Perl/perl5/issues/15623>
- The regex engine was previously producing incorrect results in some rare situations when backtracking past an alternation that matches only one thing; this showed up as capture buffers (`$1`, `$2`, *etc.*) erroneously containing data from regex execution paths that weren't actually executed for the final match. [GH #15666] <https://github.com/Perl/perl5/issues/15666>
- Certain regexes making use of the experimental `regex_sets` feature could trigger an assertion failure. This has been fixed. [GH #15620] <https://github.com/Perl/perl5/issues/15620>
- Invalid assignments to a reference constructor (*e.g.*, `\eval=time`) could sometimes crash in addition to giving a syntax error. [GH #14815] <https://github.com/Perl/perl5/issues/14815>
- The parser could sometimes crash if a bareword came after `evalbytes`. [GH #15586] <https://github.com/Perl/perl5/issues/15586>
- Autoloading via a method call would warn erroneously (''Use of inherited AUTOLOAD for non-method'') if there was a stub present in the package into which the invocant had been blessed. The warning is no longer emitted in such circumstances. [GH #9094] <https://github.com/Perl/perl5/issues/9094>
- The use of `splice` on arrays with non-existent elements could cause other operators to crash. [GH #15577] <https://github.com/Perl/perl5/issues/15577>
- A possible buffer overrun when a pattern contains a fixed utf8 substring. [GH #15534] <https://github.com/Perl/perl5/issues/15534>
- Fixed two possible use-after-free bugs in perl's lexer. [GH #15549] <https://github.com/Perl/perl5/issues/15549>
- Fixed a crash with `s///l` where it thought it was dealing with UTF-8 when it wasn't. [GH #15543] <https://github.com/Perl/perl5/issues/15543>

- Fixed a place where the regex parser was not setting the syntax error correctly on a syntactically incorrect pattern. [GH #15565] <https://github.com/Perl/perl5/issues/15565>
- The `&.` operator (and the `"&"` operator, when it treats its arguments as strings) were failing to append a trailing null byte if at least one string was marked as utf8 internally. Many code paths (system calls, regexp compilation) still expect there to be a null byte in the string buffer just past the end of the logical string. An assertion failure was the result. [GH #15606] <https://github.com/Perl/perl5/issues/15606>
- Avoid a heap-after-use error in the parser when creating an error messge for a syntactically invalid heredoc. [GH #15527] <https://github.com/Perl/perl5/issues/15527>
- Fix a segfault when run with `-DC` options on DEBUGGING builds. [GH #15563] <https://github.com/Perl/perl5/issues/15563>
- Fixed the parser error handling in subroutine attributes for an '`:attr(foo`' that does not have an ending '`")`"'.
- Fix the perl lexer to correctly handle a backslash as the last char in quoted-string context. This actually fixed two bugs, [GH #15546] <https://github.com/Perl/perl5/issues/15546> and [GH #15582] <https://github.com/Perl/perl5/issues/15582>.
- In the API function `gv_fetchmethod_pvn_flags`, rework separator parsing to prevent possible string overrun with an invalid `len` argument. [GH #15598] <https://github.com/Perl/perl5/issues/15598>
- Problems with in-place array sorts: code like `@a = sort { ... } @a`, where the source and destination of the sort are the same plain array, are optimised to do less copying around. Two side-effects of this optimisation were that the contents of `@a` as seen by sort routines were partially sorted; and under some circumstances accessing `@a` during the sort could crash the interpreter. Both these issues have been fixed, and Sort functions see the original value of `@a`. [GH #15387] <https://github.com/Perl/perl5/issues/15387>
- Non-ASCII string delimiters are now reported correctly in error messages for unterminated strings. [GH #15469] <https://github.com/Perl/perl5/issues/15469>
- `pack("p", ...)` used to emit its warning (“Attempt to pack pointer to temporary value”) erroneously in some cases, but has been fixed.
- `@DB::args` is now exempt from “used once” warnings. The warnings only occurred under **-w**, because *warnings.pm* itself uses `@DB::args` multiple times.
- The use of built-in arrays or hash slices in a double-quoted string no longer issues a warning (“Possible unintended interpolation...”) if the variable has not been mentioned before. This affected code like `qq|@DB::args|` and `qq|@SIG{'CHLD', 'HUP'}|`. (The special variables `@-` and `@+` were already exempt from the warning.)
- `gethostent` and similar functions now perform a null check internally, to avoid crashing with the torsocks library. This was a regression from v5.22. [GH #15478] <https://github.com/Perl/perl5/issues/15478>
- `defined *{'!'}`, `defined *{'['}`, and `defined *{'-'}` no longer leak memory if the typeglob in question has never been accessed before.
- Mentioning the same constant twice in a row (which is a syntax error) no longer fails an assertion under debugging builds. This was a regression from v5.20. [GH #15017] <https://github.com/Perl/perl5/issues/15017>
- Many issues relating to `printf "%a"` of hexadecimal floating point were fixed. In addition, the “subnormals” (formerly known as “denormals”) floating point numbers are now supported both with the plain IEEE 754 floating point numbers (64-bit or 128-bit) and the x86 80-bit “extended precision”. Note that subnormal hexadecimal floating point literals will give a warning about “exponent underflow”. [GH #15495] <https://github.com/Perl/perl5/issues/15495> [GH #15503] <https://github.com/Perl/perl5/issues/15503> [GH #15504] <https://github.com/Perl/perl5/issues/15504> [GH #15505] <https://github.com/Perl/perl5/issues/15505> [GH #15510] <https://github.com/Perl/perl5/issues/15510> [GH #15512] <https://github.com/Perl/perl5/issues/15512>

- A regression in v5.24 with `tr/\N{U+...}/foo/` when the code point was between 128 and 255 has been fixed. [GH #15475] <https://github.com/Perl/perl5/issues/15475>.
- Use of a string delimiter whose code point is above 2**31 now works correctly on platforms that allow this. Previously, certain characters, due to truncation, would be confused with other delimiter characters with special meaning (such as `"?"` in `m?...?`), resulting in inconsistent behaviour. Note that this is non-portable, and is based on Perl's extension to UTF-8, and is probably not displayable nor enterable by any editor. [GH #15477] <https://github.com/Perl/perl5/issues/15477>
- `@{x` followed by a newline where `"x"` represents a control or non-ASCII character no longer produces a garbled syntax error message or a crash. [GH #15518] <https://github.com/Perl/perl5/issues/15518>
- An assertion failure with `%: = 0` has been fixed. [GH #15358] <https://github.com/Perl/perl5/issues/15358>
- In Perl 5.18, the parsing of `"$foo::$bar"` was accidentally changed, such that it would be treated as `$foo."::".$bar`. The previous behavior, which was to parse it as `$foo:: . $bar`, has been restored. [GH #15408] <https://github.com/Perl/perl5/issues/15408>
- Since Perl 5.20, line numbers have been off by one when perl is invoked with the **-x** switch. This has been fixed. [GH #15413] <https://github.com/Perl/perl5/issues/15413>
- Vivifying a subroutine stub in a deleted stash (*e.g.*, `delete $My::{"Foo::"}; \&My::Foo::foo`) no longer crashes. It had begun crashing in Perl 5.18. [GH #15420] <https://github.com/Perl/perl5/issues/15420>
- Some obscure cases of subroutines and file handles being freed at the same time could result in crashes, but have been fixed. The crash was introduced in Perl 5.22. [GH #15435] <https://github.com/Perl/perl5/issues/15435>
- Code that looks for a variable name associated with an uninitialized value could cause an assertion failure in cases where magic is involved, such as `$ISA[0][0]`. This has now been fixed. [GH #15364] <https://github.com/Perl/perl5/issues/15364>
- A crash caused by code generating the warning "Subroutine STASH::NAME redefined" in cases such as `sub P::f{} undef *P::; *P::f =sub{};` has been fixed. In these cases, where the STASH is missing, the warning will now appear as "Subroutine NAME redefined". [GH #15368] <https://github.com/Perl/perl5/issues/15368>
- Fixed an assertion triggered by some code that handles deprecated behavior in formats, *e.g.*, in cases like this:

  ```
  format STDOUT =
  @
  0"$x"
  ```

  [GH #15366] <https://github.com/Perl/perl5/issues/15366>
- A possible divide by zero in string transformation code on Windows has been avoided, fixing a crash when collating an empty string. [GH #15439] <https://github.com/Perl/perl5/issues/15439>
- Some regular expression parsing glitches could lead to assertion failures with regular expressions such as `/(?<=/` and `/(?<!/`. This has now been fixed. [GH #15332] <https://github.com/Perl/perl5/issues/15332>
- `until ($x = 1) { ... }` and `... until $x = 1` now properly warn when syntax warnings are enabled. [GH #15138] <https://github.com/Perl/perl5/issues/15138>
- **socket()** now leaves the error code returned by the system in `$!` on failure. [GH #15383] <https://github.com/Perl/perl5/issues/15383>
- Assignment variants of any bitwise ops under the `bitwise` feature would crash if the left-hand side was an array or hash. [GH #15346] <https://github.com/Perl/perl5/issues/15346>
- `require` followed by a single colon (as in `foo() ? require : ...` is now parsed correctly as `require` with implicit `$_`, rather than `require ""`. [GH #15380] <https://github.com/Perl/perl5/issues/15380>

- Scalar `keys %hash` can now be assigned to consistently in all scalar lvalue contexts. Previously it worked for some contexts but not others.
- List assignment to `vec` or `substr` with an array or hash for its first argument used to result in crashes or ‘‘Can’t coerce’’ error messages at run time, unlike scalar assignment, which would give an error at compile time. List assignment now gives a compile-time error, too. [GH #15370] <https://github.com/Perl/perl5/issues/15370>
- Expressions containing an `&&` or `||` operator (or their synonyms `and` and `or`) were being compiled incorrectly in some cases. If the left-hand side consisted of either a negated bareword constant or a negated `do {}` block containing a constant expression, and the right-hand side consisted of a negated non-foldable expression, one of the negations was effectively ignored. The same was true of `if` and `unless` statement modifiers, though with the left-hand and right-hand sides swapped. This long-standing bug has now been fixed. [GH #15285] <https://github.com/Perl/perl5/issues/15285>
- `reset` with an argument no longer crashes when encountering stash entries other than globs. [GH #15314] <https://github.com/Perl/perl5/issues/15314>
- Assignment of hashes to, and deletion of, typeglobs named `*::::::` no longer causes crashes. [GH #15307] <https://github.com/Perl/perl5/issues/15307>
- Perl wasn’t correctly handling true/false values in the LHS of a list assign; specifically the truth values returned by boolean operators. This could trigger an assertion failure in something like the following:

  ```
  for ($x > $y) {
      ($_, ...) = (...); # here $_ is aliased to a truth value
  }
  ```

  This was a regression from v5.24. [GH #15690] <https://github.com/Perl/perl5/issues/15690>
- Assertion failure with user-defined Unicode-like properties. [GH #15696] <https://github.com/Perl/perl5/issues/15696>
- Fix error message for unclosed `\N{` in a regex. An unclosed `\N{` could give the wrong error message: `"\N{NAME} must be resolved by the lexer"`.
- List assignment in list context where the LHS contained aggregates and where there were not enough RHS elements, used to skip scalar lvalues. Previously, `((&#36;a,&#36;b,@c,&#36;d) = (1))` in list context returned `(&#36;a)`; now it returns `(&#36;a,&#36;b,&#36;d)`. `((&#36;a,&#36;b,&#36;c) = (1))` is unchanged: it still returns `(&#36;a,&#36;b,&#36;c)`. This can be seen in the following:

  ```
  sub inc { $_++ for @_ }
  inc(($a,$b,@c,$d) = (10))
  ```

  Formerly, the values of `(&#36;a,&#36;b,&#36;d)` would be left as `(11,undef,undef)`; now they are `(11,1,1)`.
- Code like this: `/(?{ s!!! })/` could trigger infinite recursion on the C stack (not the normal perl stack) when the last successful pattern in scope is itself. We avoid the segfault by simply forbidding the use of the empty pattern when it would resolve to the currently executing pattern. [GH #15669] <https://github.com/Perl/perl5/issues/15669>
- Avoid reading beyond the end of the line buffer in perl’s lexer when there’s a short UTF-8 character at the end. [GH #15531] <https://github.com/Perl/perl5/issues/15531>
- Alternations in regular expressions were sometimes failing to match a utf8 string against a utf8 alternate. [GH #15680] <https://github.com/Perl/perl5/issues/15680>
- Make `do "a\0b"` fail silently (and return `undef` and set `&#36;!`) instead of throwing an error. [GH #15676] <https://github.com/Perl/perl5/issues/15676>
- `chdir` with no argument didn’t ensure that there was stack space available for returning its result. [GH #15569] <https://github.com/Perl/perl5/issues/15569>
- All error messages related to `do` now refer to `do`; some formerly claimed to be from `require` instead.

- Executing `undef $x` where `$x` is tied or magical no longer incorrectly blames the variable for an uninitialized-value warning encountered by the tied/magical code.
- Code like `$x = $x . "a"` was incorrectly failing to yield a use of uninitialized value warning when `$x` was a lexical variable with an undefined value. That has now been fixed. [GH #15269] <https://github.com/Perl/perl5/issues/15269>
- `undef *_; shift` or `undef *_; pop` inside a subroutine, with no argument to `shift` or `pop`, began crashing in Perl 5.14, but has now been fixed.
- `"string$scalar->$*"` now correctly prefers concatenation overloading to string overloading if `$scalar->$*` returns an overloaded object, bringing it into consistency with `$$scalar`.
- `/@0{0*->@*/*0` and similar contortions used to crash, but no longer do, but merely produce a syntax error. [GH #15333] <https://github.com/Perl/perl5/issues/15333>
- `do` or `require` with an argument which is a reference or typeglob which, when stringified, contains a null character, started crashing in Perl 5.20, but has now been fixed. [GH #15337] <https://github.com/Perl/perl5/issues/15337>
- Improve the error message for a missing `tie()` package/method. This brings the error messages in line with the ones used for normal method calls.
- Parsing bad POSIX charclasses no longer leaks memory. [GH #15382] <https://github.com/Perl/perl5/issues/15382>

## Known Problems

- G++ 6 handles subnormal (denormal) floating point values differently than gcc 6 or g++ 5 resulting in ‘‘flush-to-zero’’. The end result is that if you specify very small values using the hexadecimal floating point format, like `0x1.fffffffffffffp-1022`, they become zeros. [GH #15990] <https://github.com/Perl/perl5/issues/15990>

## Errata From Previous Releases

- Fixed issues with recursive regexes. The behavior was fixed in Perl 5.24. [GH #14935] <https://github.com/Perl/perl5/issues/14935>

## Obituary

Jon Portnoy (AVENJ), a prolific Perl author and admired Gentoo community member, has passed away on August 10, 2016. He will be remembered and missed by all those who he came in contact with, and enriched with his intellect, wit, and spirit.

It is with great sadness that we also note Kip Hampton’s passing. Probably best known as the author of the Perl & XML column on XML.com, he was a core contributor to AxKit, an XML server platform that became an Apache Foundation project. He was a frequent speaker in the early days at OSCON, and most recently at YAPC::NA in Madison. He was frequently on irc.perl.org as ubu, generally in the #axkit−dahut community, the group responsible for YAPC::NA Asheville in 2011.

Kip and his constant contributions to the community will be greatly missed.

## Acknowledgements

Perl 5.26.0 represents approximately 13 months of development since Perl 5.24.0 and contains approximately 360,000 lines of changes across 2,600 files from 86 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 230,000 lines of changes to 1,800 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.26.0:

Aaron Crane, Abigail, Ævar Arnfjörð Bjarmason, Alex Vandiver, Andreas König, Andreas Voegele, Andrew Fresh, Andy Lester, Aristotle Pagaltzis, Chad Granum, Chase Whitener, Chris ’BinGOs’ Williams, Chris Lamb, Christian Hansen, Christian Millour, Colin Newell, Craig A. Berry, Dagfinn Ilmari Mannsåker, Dan Collins, Daniel Dragan, Dave Cross, Dave Rolsky, David Golden, David H. Gutteridge, David Mitchell, Dominic Hargreaves, Doug Bell, E. Choroba, Ed Avis, Father Chrysostomos, François Perrad, Hauke D, H.Merijn Brand, Hugo van der Sanden, Ivan Pozdeev, James E Keenan, James Raspass, Jarkko Hietaniemi, Jerry D. Hedden, Jim Cromie, J. Nick Koston, John Lightsey, Karen Etheridge, Karl Williamson, Leon Timmermans, Lukas Mai, Matthew Horsfall,

Maxwell Carey, Misty De Meo, Neil Bowers, Nicholas Clark, Nicolas R., Niko Tyni, Pali, Paul Marquess, Peter Avalos, Petr PísaX, Pino Toscano, Rafael Garcia-Suarez, Reini Urban, Renee Baecker, Ricardo Signes, Richard Levitte, Rick Delaney, Salvador Fandiño, Samuel Thibault, Sawyer X, Sébastien Aperghis-Tramoni, Sergey Aleynikov, Shlomi Fish, Smylers, Stefan Seifert, Steffen Müller, Stevan Little, Steve Hay, Steven Humphrey, Sullivan Beck, Theo Buehler, Thomas Sibley, Todd Rinaldo, Tomasz Konojacki, Tony Cook, Unicode Consortium, Yaroslav Kuzmin, Yves Orton, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to `perlbug@perl.org` to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5261delta − what is new for perl v5.26.1

## DESCRIPTION

This document describes differences between the 5.26.0 release and the 5.26.1 release.

If you are upgrading from an earlier release such as 5.24.0, first read perl5260delta, which describes differences between 5.24.0 and 5.26.0.

## Security

### [CVE−2017−12837] Heap buffer overflow in regular expression compiler

Compiling certain regular expression patterns with the case-insensitive modifier could cause a heap buffer overflow and crash perl. This has now been fixed. [GH #16021] <https://github.com/Perl/perl5/issues/16021>

### [CVE−2017−12883] Buffer over-read in regular expression parser

For certain types of syntax error in a regular expression pattern, the error message could either contain the contents of a random, possibly large, chunk of memory, or could crash perl. This has now been fixed. [GH #16025] <https://github.com/Perl/perl5/issues/16025>

### [CVE−2017−12814] `$ENV{$key}` stack buffer overflow on Windows

A possible stack buffer overflow in the `%ENV` code on Windows has been fixed by removing the buffer completely since it was superfluous anyway. [GH #16051] <https://github.com/Perl/perl5/issues/16051>

## Incompatible Changes

There are no changes intentionally incompatible with 5.26.0. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- base has been upgraded from version 2.25 to 2.26.

  The effects of dotless `@INC` on this module have been limited by the introduction of a more refined and accurate solution for removing `'.'` from `@INC` while reducing the false positives.

- charnames has been upgraded from version 1.44 to 1.45.

- Module::CoreList has been upgraded from version 5.20170530 to 5.20170922_26.

## Platform Support

### Platform-Specific Notes

FreeBSD

- Building with **g++** on FreeBSD−11.0 has been fixed. [GH #15984] <https://github.com/Perl/perl5/issues/15984>

Windows

- Support for compiling perl on Windows using Microsoft Visual Studio 2017 (containing Visual C++ 14.1) has been added.

- Building XS modules with GCC 6 in a 64−bit build of Perl failed due to incorrect mapping of `strtoll` and `strtoull`. This has now been fixed. [GH #16074] <https://github.com/Perl/perl5/issues/16074> [cpan #121683] <https://rt.cpan.org/Public/Bug/Display.html?id=121683> [cpan #122353] <https://rt.cpan.org/Public/Bug/Display.html?id=122353>

## Selected Bug Fixes

- Several built-in functions previously had bugs that could cause them to write to the internal stack without allocating room for the item being written. In rare situations, this could have led to a crash. These bugs have now been fixed, and if any similar bugs are introduced in future, they will be detected automatically in debugging builds. [GH #16076] <https://github.com/Perl/perl5/issues/16076>

- Using a symbolic ref with postderef syntax as the key in a hash lookup was yielding an assertion failure on debugging builds. [GH #16029] <https://github.com/Perl/perl5/issues/16029>

- List assignment (`aassign`) could in some rare cases allocate an entry on the mortal stack and leave the entry uninitialized. [GH #16017] <https://github.com/Perl/perl5/issues/16017>
- Attempting to apply an attribute to an `our` variable where a function of that name already exists could result in a NULL pointer being supplied where an SV was expected, crashing perl. [perl #131597] <https://rt.perl.org/Public/Bug/Display.html?id=131597>
- The code that vivifies a typeglob out of a code ref made some false assumptions that could lead to a crash in cases such as `$::{"A"} = sub {}; \&{"A"}`. This has now been fixed. [GH #15937] <https://github.com/Perl/perl5/issues/15937>
- `my_atof2` no longer reads beyond the terminating NUL, which previously occurred if the decimal point is immediately before the NUL. [GH #16002] <https://github.com/Perl/perl5/issues/16002>
- Occasional ‘‘Malformed UTF-8 character’’ crashes in `s//` on utf8 strings have been fixed. [GH #16019] <https://github.com/Perl/perl5/issues/16019>
- `perldoc -f s` now finds `s///`. [GH #15989] <https://github.com/Perl/perl5/issues/15989>
- Some erroneous warnings after utf8 conversion have been fixed. [GH #15958] <https://github.com/Perl/perl5/issues/15958>
- The `jmpenv` frame to catch Perl exceptions is set up lazily, and this used to be a bit too lazy. The catcher is now set up earlier, preventing some possible crashes. [GH #11804] <https://github.com/Perl/perl5/issues/11804>
- Spurious ‘‘Assuming NOT a POSIX class’’ warnings have been removed. [GH #16001] <https://github.com/Perl/perl5/issues/16001>

## Acknowledgements

Perl 5.26.1 represents approximately 4 months of development since Perl 5.26.0 and contains approximately 8,900 lines of changes across 85 files from 23 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 990 lines of changes to 38 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.26.1:

Aaron Crane, Andy Dougherty, Aristotle Pagaltzis, Chris ’BinGOs’ Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, David Mitchell, E. Choroba, Eric Herman, Father Chrysostomos, Jacques Germishuys, James E Keenan, John SJ Anderson, Karl Williamson, Ken Brown, Lukas Mai, Matthew Horsfall, Ricardo Signes, Sawyer X, Steve Hay, Tony Cook, Yves Orton, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ‘‘SECURITY VULNERABILITY CONTACT INFORMATION’’ in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5262delta − what is new for perl v5.26.2

## DESCRIPTION

This document describes differences between the 5.26.1 release and the 5.26.2 release.

If you are upgrading from an earlier release such as 5.26.0, first read perl5261delta, which describes differences between 5.26.0 and 5.26.1.

## Security

### [CVE−2018−6797] heap-buffer-overflow (WRITE of size 1) in S_regatom (regcomp.c)

A crafted regular expression could cause a heap buffer write overflow, with control over the bytes written. [GH #16185] <https://github.com/Perl/perl5/issues/16185>

### [CVE−2018−6798] Heap-buffer-overflow in Perl_ _byte_dump_string (utf8.c)

Matching a crafted locale dependent regular expression could cause a heap buffer read overflow and potentially information disclosure. [GH #16143] <https://github.com/Perl/perl5/issues/16143>

### [CVE−2018−6913] heap-buffer-overflow in S_pack_rec

`pack()` could cause a heap buffer write overflow with a large item count. [GH #16098] <https://github.com/Perl/perl5/issues/16098>

### Assertion failure in Perl_ _core_swash_init (utf8.c)

Control characters in a supposed Unicode property name could cause perl to crash. This has been fixed. [perl #132055] <https://rt.perl.org/Public/Bug/Display.html?id=132055> [perl #132553] <https://rt.perl.org/Public/Bug/Display.html?id=132553> [perl #132658] <https://rt.perl.org/Public/Bug/Display.html?id=132658>

## Incompatible Changes

There are no changes intentionally incompatible with 5.26.1. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20170922_26 to 5.20180414_26.
- PerlIO::via has been upgraded from version 0.16 to 0.17.
- Term::ReadLine has been upgraded from version 1.16 to 1.17.
- Unicode::UCD has been upgraded from version 0.68 to 0.69.

## Documentation

### Changes to Existing Documentation

*perluniprops*

- This has been updated to note that `\p{Word}` now includes code points matching the `\p{Join_Control}` property. The change to the property was made in Perl 5.18, but not documented until now. There are currently only two code points that match this property: U+200C (ZERO WIDTH NON-JOINER) and U+200D (ZERO WIDTH JOINER).

## Platform Support

### Platform-Specific Notes

Windows

Visual C++ compiler version detection has been improved to work on non-English language systems. [GH #16235] <https://github.com/Perl/perl5/issues/16235>

We now set `$Config{libpth}` correctly for 64−bit builds using Visual C++ versions earlier than 14.1. [GH #16269] <https://github.com/Perl/perl5/issues/16269>

## Selected Bug Fixes

- The `readpipe()` built-in function now checks at compile time that it has only one parameter expression, and puts it in scalar context, thus ensuring that it doesn’t corrupt the stack at runtime. [GH #2793] <https://github.com/Perl/perl5/issues/2793>
- Fixed a use after free bug in `pp_list` introduced in Perl 5.27.1. [GH #16124] <https://github.com/Perl/perl5/issues/16124>

- Parsing a `sub` definition could cause a use after free if the `sub` keyword was followed by whitespace including newlines (and comments). [GH #16097] <https://github.com/Perl/perl5/issues/16097>
- The tokenizer now correctly adjusts a parse pointer when skipping whitespace in an `${identifier}` construct. [perl #131949] <https://rt.perl.org/Public/Bug/Display.html?id=131949>
- Accesses to `${^LAST_FH}` no longer assert after using any of a variety of I/O operations on a non-glob. [GH #15372] <https://github.com/Perl/perl5/issues/15372>
- `sort` now performs correct reference counting when aliasing `$a` and `$b`, thus avoiding premature destruction and leakage of scalars if they are re-aliased during execution of the sort comparator. [GH #11422] <https://github.com/Perl/perl5/issues/11422>
- Some convoluted kinds of regexp no longer cause an arithmetic overflow when compiled. [GH #16113] <https://github.com/Perl/perl5/issues/16113>
- Fixed a duplicate symbol failure with **-flto -mieee-fp** builds. *pp.c* defined `_LIB_VERSION` which **-lieee** already defines. [GH #16086] <https://github.com/Perl/perl5/issues/16086>
- A NULL pointer dereference in the `S_regmatch()` function has been fixed. [perl #132017] <https://rt.perl.org/Public/Bug/Display.html?id=132017>
- Failures while compiling code within other constructs, such as with string interpolation and the right part of `s///e` now cause compilation to abort earlier.

  Previously compilation could continue in order to report other errors, but the failed sub-parse could leave partly parsed constructs on the parser shift-reduce stack, confusing the parser, leading to perl crashes. [GH #14739] <https://github.com/Perl/perl5/issues/14739>

## Acknowledgements

Perl 5.26.2 represents approximately 7 months of development since Perl 5.26.1 and contains approximately 3,300 lines of changes across 82 files from 17 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,800 lines of changes to 36 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.26.2:

Aaron Crane, Abigail, Chris 'BinGOs' Williams, H.Merijn Brand, James E Keenan, Jarkko Hietaniemi, John SJ Anderson, Karen Etheridge, Karl Williamson, Lukas Mai, Renee Baecker, Sawyer X, Steve Hay, Todd Rinaldo, Tony Cook, Yves Orton, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5263delta − what is new for perl v5.26.3

## DESCRIPTION

This document describes differences between the 5.26.2 release and the 5.26.3 release.

If you are upgrading from an earlier release such as 5.26.1, first read perl5262delta, which describes differences between 5.26.1 and 5.26.2.

## Security

### [CVE−2018−12015] Directory traversal in module Archive::Tar

By default, Archive::Tar doesn't allow extracting files outside the current working directory. However, this secure extraction mode could be bypassed by putting a symlink and a regular file with the same name into the tar file.

[GH #16580] <https://github.com/Perl/perl5/issues/16580> [cpan #125523] <https://rt.cpan.org/Ticket/Display.html?id=125523>

### [CVE−2018−18311] Integer overflow leading to buffer overflow and segmentation fault

Integer arithmetic in `Perl_my_setenv()` could wrap when the combined length of the environment variable name and value exceeded around 0x7fffffff. This could lead to writing beyond the end of an allocated buffer with attacker supplied data.

[GH #16560] <https://github.com/Perl/perl5/issues/16560>

### [CVE−2018−18312] Heap-buffer-overflow write in S_regatom (regcomp.c)

A crafted regular expression could cause heap-buffer-overflow write during compilation, potentially allowing arbitrary code execution.

[GH #16649] <https://github.com/Perl/perl5/issues/16649>

### [CVE−2018−18313] Heap-buffer-overflow read in S_grok_bslash_N (regcomp.c)

A crafted regular expression could cause heap-buffer-overflow read during compilation, potentially leading to sensitive information being leaked.

[GH #16554] <https://github.com/Perl/perl5/issues/16554>

### [CVE−2018−18314] Heap-buffer-overflow write in S_regatom (regcomp.c)

A crafted regular expression could cause heap-buffer-overflow write during compilation, potentially allowing arbitrary code execution.

[GH #16041] <https://github.com/Perl/perl5/issues/16041>

## Incompatible Changes

There are no changes intentionally incompatible with 5.26.2. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.24 to 2.24_01.
- Module::CoreList has been upgraded from version 5.20180414_26 to 5.20181129_26.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Unexpected ’]’ with no following ’)’ in (?[... in regex; marked by <−− HERE in m/%s/

  (F) While parsing an extended character class a ’]’ character was encountered at a point in the definition where the only legal use of ’]’ is to close the character class definition as part of a ’])’, you may have forgotten the close paren, or otherwise confused the parser.

- Expecting close paren for nested extended charclass in regex; marked by <−− HERE in m/%s/

  (F) While parsing a nested extended character class like:

```
(?[ ... (?flags:(?[ ... ])) ... ])
                           ^
```

we expected to see a close paren ')' (marked by ^) but did not.

- Expecting close paren for wrapper for nested extended charclass in regex; marked by <-- HERE in m/%s/

  (F) While parsing a nested extended character class like:

```
(?[ ... (?flags:(?[ ... ])) ... ])
                           ^
```

  we expected to see a close paren ')' (marked by ^) but did not.

### Changes to Existing Diagnostics

- Syntax error in (?[...]) in regex; marked by <-- HERE in m/%s/

  This fatal error message has been slightly expanded (from "Syntax error in (?[...]) in regex m/%s/") for greater clarity.

## Acknowledgements

Perl 5.26.3 represents approximately 8 months of development since Perl 5.26.2 and contains approximately 4,500 lines of changes across 51 files from 15 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 770 lines of changes to 10 .pm, .t, .c and .h files.

Perl continues to flourish into its third decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.26.3:

Aaron Crane, Abigail, Chris 'BinGOs' Williams, Dagfinn Ilmari Mannsåker, David Mitchell, H.Merijn Brand, James E Keenan, John SJ Anderson, Karen Etheridge, Karl Williamson, Sawyer X, Steve Hay, Todd Rinaldo, Tony Cook, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see "SECURITY VULNERABILITY CONTACT INFORMATION" in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5280delta − what is new for perl v5.28.0

## DESCRIPTION

This document describes differences between the 5.26.0 release and the 5.28.0 release.

If you are upgrading from an earlier release such as 5.24.0, first read perl5260delta, which describes differences between 5.24.0 and 5.26.0.

## Core Enhancements

### Unicode 10.0 is supported

A list of changes is at <http://www.unicode.org/versions/Unicode10.0.0>.

### `delete` on key/value hash slices

`delete` can now be used on key/value hash slices, returning the keys along with the deleted values. [GH #15982] <https://github.com/Perl/perl5/issues/15982>

### Experimentally, there are now alphabetic synonyms for some regular expression assertions

If you find it difficult to remember how to write certain of the pattern assertions, there are now alphabetic synonyms.

```
CURRENT                 NEW SYNONYMS
------                  ------------
(?=...)         (*pla:...) or (*positive_lookahead:...)
(?!...)         (*nla:...) or (*negative_lookahead:...)
(?<=...)        (*plb:...) or (*positive_lookbehind:...)
(?<!...)        (*nlb:...) or (*negative_lookbehind:...)
(?>...)         (*atomic:...)
```

These are considered experimental, so using any of these will raise (unless turned off) a warning in the `experimental::alpha_assertions` category.

### Mixed Unicode scripts are now detectable

A mixture of scripts, such as Cyrillic and Latin, in a string is often the sign of a spoofing attack. A new regular expression construct now allows for easy detection of these. For example, you can say

```
qr/(*script_run: \d+ \b )/x
```

And the digits matched will all be from the same set of 10. You won't get a look-alike digit from a different script that has a different value than what it appears to be.

Or:

```
qr/(*sr: \b \w+ \b )/x
```

makes sure that all the characters come from the same script.

You can also combine script runs with `(?>...)` (or `*atomic:...`).

Instead of writing:

```
(*sr:(?<...))
```

you can now run:

```
(*asr:...)
# or
(*atomic_script_run:...)
```

This is considered experimental, so using it will raise (unless turned off) a warning in the `experimental::script_run` category.

See ''Script Runs'' in perlre.

### In-place editing with `perl -i` is now safer

Previously in-place editing (`perl -i`) would delete or rename the input file as soon as you started working on a new file.

Without backups this would result in loss of data if there was an error, such as a full disk, when writing to the output file.

This has changed so that the input file isn't replaced until the output file has been completely written and successfully closed.

This works by creating a work file in the same directory, which is renamed over the input file once the output file is complete.

Incompatibilities:

- Since this renaming needs to only happen once, if you create a thread or child process, that renaming will only happen in the original thread or process.
- If you change directories while processing a file, and your operating system doesn't provide the `unlinkat()`, `renameat()` and `fchmodat()` functions, the final rename step may fail.

[GH #15216] <https://github.com/Perl/perl5/issues/15216>

### Initialisation of aggregate state variables

A persistent lexical array or hash variable can now be initialized, by an expression such as `state @a = qw(x y z)`. Initialization of a list of persistent lexical variables is still not possible.

### Full-size inode numbers

On platforms where inode numbers are of a type larger than perl's native integer numerical types, stat will preserve the full content of large inode numbers by returning them in the form of strings of decimal digits. Exact comparison of inode numbers can thus be achieved by comparing with `eq` rather than `==`. Comparison with `==`, and other numerical operations (which are usually meaningless on inode numbers), work as well as they did before, which is to say they fall back to floating point, and ultimately operate on a fairly useless rounded inode number if the real inode number is too big for the floating point format.

### The `sprintf %j` format size modifier is now available with pre−C99 compilers

The actual size used depends on the platform, so remains unportable.

### Close-on-exec flag set atomically

When opening a file descriptor, perl now generally opens it with its close-on-exec flag already set, on platforms that support doing so. This improves thread safety, because it means that an `exec` initiated by one thread can no longer cause a file descriptor in the process of being opened by another thread to be accidentally passed to the executed program.

Additionally, perl now sets the close-on-exec flag more reliably, whether it does so atomically or not. Most file descriptors were getting the flag set, but some were being missed.

### String− and number-specific bitwise ops are no longer experimental

The new string-specific (`&. |. ^. ~.`) and number-specific (`& | ^ ~`) bitwise operators introduced in Perl 5.22 that are available within the scope of `use feature 'bitwise'` are no longer experimental. Because the number-specific ops are spelled the same way as the existing operators that choose their behaviour based on their operands, these operators must still be enabled via the ''bitwise'' feature, in either of these two ways:

```
use feature "bitwise";

use v5.28; # "bitwise" now included
```

They are also now enabled by the **−E** command-line switch.

The ''bitwise'' feature no longer emits a warning. Existing code that disables the ''experimental::bitwise'' warning category that the feature previously used will continue to work.

One caveat that module authors ought to be aware of is that the numeric operators now pass a fifth TRUE argument to overload methods. Any methods that check the number of operands may croak if they do not expect so many. XS authors in particular should be aware that this:

```
SV *
bitop_handler (lobj, robj, swap)
```

may need to be changed to this:

```
SV *
bitop_handler (lobj, robj, swap, ...)
```

### Locales are now thread-safe on systems that support them

These systems include Windows starting with Visual Studio 2005, and in POSIX 2008 systems.

The implication is that you are now free to use locales and change them in a threaded environment. Your changes affect only your thread. See ''Multi-threaded operation'' in perllocale

### New read-only predefined variable `${^SAFE_LOCALES}`

This variable is 1 if the Perl interpreter is operating in an environment where it is safe to use and change locales (see perllocale.) This variable is true when the perl is unthreaded, or compiled in a platform that supports thread-safe locale operation (see previous item).

## Security

### [CVE−2017−12837] Heap buffer overflow in regular expression compiler

Compiling certain regular expression patterns with the case-insensitive modifier could cause a heap buffer overflow and crash perl. This has now been fixed. [GH #16021] <https://github.com/Perl/perl5/issues/16021>

### [CVE−2017−12883] Buffer over-read in regular expression parser

For certain types of syntax error in a regular expression pattern, the error message could either contain the contents of a random, possibly large, chunk of memory, or could crash perl. This has now been fixed. [GH #16025] <https://github.com/Perl/perl5/issues/16025>

### [CVE−2017−12814] `$ENV{$key}` stack buffer overflow on Windows

A possible stack buffer overflow in the `%ENV` code on Windows has been fixed by removing the buffer completely since it was superfluous anyway. [GH #16051] <https://github.com/Perl/perl5/issues/16051>

### Default Hash Function Change

Perl 5.28.0 retires various older hash functions which are not viewed as sufficiently secure for use in Perl. We now support four general purpose hash functions, Siphash (2−4 and 1−3 variants), and Zaphod32, and StadtX hash. In addition we support SBOX32 (a form of tabular hashing) for hashing short strings, in conjunction with any of the other hash functions provided.

By default Perl is configured to support SBOX hashing of strings up to 24 characters, in conjunction with StadtX hashing on 64 bit builds, and Zaphod32 hashing for 32 bit builds.

You may control these settings with the following options to Configure:

```
−DPERL_HASH_FUNC_SIPHASH
−DPERL_HASH_FUNC_SIPHASH13
−DPERL_HASH_FUNC_STADTX
−DPERL_HASH_FUNC_ZAPHOD32
```

To disable SBOX hashing you can use

```
−DPERL_HASH_USE_SBOX32_ALSO=0
```

And to set the maximum length to use SBOX32 hashing on with:

```
−DSBOX32_MAX_LEN=16
```

The maximum length allowed is 256. There probably isn't much point in setting it higher than the default.

## Incompatible Changes

### Subroutine attribute and signature order

The experimental subroutine signatures feature has been changed so that subroutine attributes must now come before the signature rather than after. This is because attributes like `:lvalue` can affect the compilation of code within the signature, for example:

```
sub f :lvalue ($a = do { $x = "abc"; return substr($x,0,1)}) { ...}
```

Note that this the second time they have been flipped:

```
sub f :lvalue ($a, $b) { ... }; # 5.20; 5.28 onwards
sub f ($a, $b) :lvalue { ... }; # 5.22 − 5.26
```

### Comma-less variable lists in formats are no longer allowed

Omitting the commas between variables passed to formats is no longer allowed. This has been deprecated since Perl 5.000.

**The `:locked` and `:unique` attributes have been removed**
These have been no-ops and deprecated since Perl 5.12 and 5.10, respectively.

**`\N{}` with nothing between the braces is now illegal**
This has been deprecated since Perl 5.24.

**Opening the same symbol as both a file and directory handle is no longer allowed**
Using `open()` and `opendir()` to associate both a filehandle and a dirhandle to the same symbol (glob or scalar) has been deprecated since Perl 5.10.

**Use of bare `<<` to mean `<<''` is no longer allowed**
Use of a bare terminator has been deprecated since Perl 5.000.

**Setting $/ to a reference to a non-positive integer no longer allowed**
This used to work like setting it to `undef`, but has been deprecated since Perl 5.20.

**Unicode code points with values exceeding `IV_MAX` are now fatal**
This was deprecated since Perl 5.24.

**The `B::OP::terse` method has been removed**
Use `B::Concise::b_terse` instead.

**Use of inherited AUTOLOAD for non-methods is no longer allowed**
This was deprecated in Perl 5.004.

**Use of strings with code points over 0xFF is not allowed for bitwise string operators**
Code points over `0xFF` do not make sense for bitwise operators and such an operation will now croak, except for a few remaining cases. See perldeprecation.

This was deprecated in Perl 5.24.

**Setting `${^ENCODING}` to a defined value is now illegal**
This has been deprecated since Perl 5.22 and a no-op since Perl 5.26.

**Backslash no longer escapes colon in PATH for the `-S` switch**
Previously the `-S` switch incorrectly treated backslash (‘‘\’’) as an escape for colon when traversing the `PATH` environment variable. [GH #15584] <https://github.com/Perl/perl5/issues/15584>

**the `-DH` (DEBUG_H) misfeature has been removed**
On a perl built with debugging support, the `H` flag to the `-D` debugging option has been removed. This was supposed to dump hash values, but has been broken for many years.

**Yada-yada is now strictly a statement**
By the time of its initial stable release in Perl 5.12, the `...` (yada-yada) operator was explicitly intended to serve as a statement, not an expression. However, the original implementation was confused on this point, leading to inconsistent parsing. The operator was accidentally accepted in a few situations where it did not serve as a complete statement, such as

```
... . "foo";
... if $a < $b;
```

The parsing has now been made consistent, permitting yada-yada only as a statement. Affected code can use `do{...}` to put a yada-yada into an arbitrary expression context.

**Sort algorithm can no longer be specified**
Since Perl 5.8, the sort pragma has had subpragmata `_mergesort`, `_quicksort`, and `_qsort` that can be used to specify which algorithm perl should use to implement the sort builtin. This was always considered a dubious feature that might not last, hence the underscore spellings, and they were documented as not being portable beyond Perl 5.8. These subpragmata have now been deleted, and any attempt to use them is an error. The sort pragma otherwise remains, and the algorithm-neutral `stable` subpragma can be used to control sorting behaviour. [GH #13234] <https://github.com/Perl/perl5/issues/13234>

**Over-radix digits in floating point literals**
Octal and binary floating point literals used to permit any hexadecimal digit to appear after the radix point. The digits are now restricted to those appropriate for the radix, as digits before the radix point always were.

### Return type of `unpackstring()`

The return types of the C API functions `unpackstring()` and `unpack_str()` have changed from `I32` to `SSize_t`, in order to accommodate datasets of more than two billion items.

## Deprecations

### Use of `vec` on strings with code points above 0xFF is deprecated

Such strings are represented internally in UTF-8, and `vec` is a bit-oriented operation that will likely give unexpected results on those strings.

### Some uses of unescaped `''{''` in regexes are no longer fatal

Perl 5.26.0 fatalized some uses of an unescaped left brace, but an exception was made at the last minute, specifically crafted to be a minimal change to allow GNU Autoconf to work. That tool is heavily depended upon, and continues to use the deprecated usage. Its use of an unescaped left brace is one where we have no intention of repurposing `"{"` to be something other than itself.

That exception is now generalized to include various other such cases where the `"{"` will not be repurposed.

Note that these uses continue to raise a deprecation message.

### Use of unescaped `''{''` immediately after a `''('')` in regular expression patterns is deprecated

Using unescaped left braces is officially deprecated everywhere, but it is not enforced in contexts where their use does not interfere with expected extensions to the language. A deprecation is added in this release when the brace appears immediately after an opening parenthesis. Before this, even if the brace was part of a legal quantifier, it was not interpreted as such, but as the literal characters, unlike other quantifiers that follow a `"("` which are considered errors. Now, their use will raise a deprecation message, unless turned off.

### Assignment to `$[` will be fatal in Perl 5.30

Assigning a non-zero value to `$[` has been deprecated since Perl 5.12, but was never given a deadline for removal. This has now been scheduled for Perl 5.30.

### hostname() won't accept arguments in Perl 5.32

Passing arguments to `Sys::Hostname::hostname()` was already deprecated, but didn't have a removal date. This has now been scheduled for Perl 5.32. [GH #14662] <https://github.com/Perl/perl5/issues/14662>

### Module removals

The following modules will be removed from the core distribution in a future release, and will at that time need to be installed from CPAN. Distributions on CPAN which require these modules will need to list them as prerequisites.

The core versions of these modules will now issue `"deprecated"`-category warnings to alert you to this fact. To silence these deprecation warnings, install the modules in question from CPAN.

Note that these are (with rare exceptions) fine modules that you are encouraged to continue to use. Their disinclusion from core primarily hinges on their necessity to bootstrapping a fully functional, CPAN-capable Perl installation, not usually on concerns over their design.

B::Debug
Locale::Codes and its associated Country, Currency and Language modules

## Performance Enhancements

- The start up overhead for creating regular expression patterns with Unicode properties (`\p{...}`) has been greatly reduced in most cases.
- Many string concatenation expressions are now considerably faster, due to the introduction internally of a `multiconcat` opcode which combines multiple concatenations, and optionally a `=` or `.=`, into a single action. For example, apart from retrieving `$s`, `$a` and `$b`, this whole expression is now handled as a single op:

  ```
  $s .= "a=$a b=$b\n"
  ```

  As a special case, if the LHS of an assignment is a lexical variable or `my $s`, the op itself handles retrieving the lexical variable, which is faster.

  In general, the more the expression includes a mix of constant strings and variable expressions, the

longer the expression, and the more it mixes together non−utf8 and utf8 strings, the more marked the performance improvement. For example on a `x86_64` system, this code has been benchmarked running four times faster:

```
my $s;
my $a = "ab\x{100}cde";
my $b = "fghij";
my $c = "\x{101}klmn";

for my $i (1..10_000_000) {
    $s = "\x{100}wxyz";
    $s .= "foo=$a bar=$b baz=$c";
}
```

In addition, `sprintf` expressions which have a constant format containing only `%s` and `%%` format elements, and which have a fixed number of arguments, are now also optimised into a `multiconcat` op.

- The `ref()` builtin is now much faster in boolean context, since it no longer bothers to construct a temporary string like `Foo=ARRAY(0x134af48)`.
- `keys()` in void and scalar contexts is now more efficient.
- The common idiom of comparing the result of **index()** with −1 is now specifically optimised, e.g.

  ```
  if (index(...) != -1) { ... }
  ```
- `for()` loops and similar constructs are now more efficient in most cases.
- File::Glob has been modified to remove unnecessary backtracking and recursion, thanks to Russ Cox. See <https://research.swtch.com/glob> for more details.
- The XS-level `SvTRUE()` API function is now more efficient.
- Various integer-returning ops are now more efficient in scalar/boolean context.
- Slightly improved performance when parsing stash names. [GH #15689] <https://github.com/Perl/perl5/issues/15689>
- Calls to `require` for an already loaded module are now slightly faster. [GH #16175] <https://github.com/Perl/perl5/issues/16175>
- The performance of pattern matching `[[:ascii:]]` and `[[:^ascii:]]` has been improved significantly except on EBCDIC platforms.
- Various optimizations have been applied to matching regular expression patterns, so under the right circumstances, significant performance gains may be noticed. But in an application with many varied patterns, little overall improvement likely will be seen.
- Other optimizations have been applied to UTF−8 handling, but these are not typically a major factor in most applications.

## Modules and Pragmata

Key highlights in this release across several modules:

### Removal of use vars

The usage of `use vars` has been discouraged since the introduction of `our` in Perl 5.6.0. Where possible the usage of this pragma has now been removed from the Perl source code.

This had a slight effect (for the better) on the output of WARNING_BITS in B::Deparse.

### Use of DynaLoader changed to XSLoader in many modules

XSLoader is more modern, and most modules already require perl 5.6 or greater, so no functionality is lost by switching. In some cases, we have also made changes to the local implementation that may not be reflected in the version on CPAN due to a desire to maintain more backwards compatibility.

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.24 to 2.30.

  This update also handled CVE−2018−12015: directory traversal vulnerability. [cpan #125523] <https://rt.cpan.org/Ticket/Display.html?id=125523>

- arybase has been upgraded from version 0.12 to 0.15.
- Attribute::Handlers has been upgraded from version 0.99 to 1.01.
- attributes has been upgraded from version 0.29 to 0.33.
- B has been upgraded from version 1.68 to 1.74.
- B::Concise has been upgraded from version 0.999 to 1.003.
- B::Debug has been upgraded from version 1.24 to 1.26.

  NOTE: B::Debug is deprecated and may be removed from a future version of Perl.
- B::Deparse has been upgraded from version 1.40 to 1.48.

  It includes many bug fixes, and in particular, it now deparses variable attributes correctly:

  ```
  my $x :foo;  # used to deparse as
               # 'attributes'->import('main', \$x, 'foo'), my $x;
  ```
- base has been upgraded from version 2.25 to 2.27.
- bignum has been upgraded from version 0.47 to 0.49.
- blib has been upgraded from version 1.06 to 1.07.
- bytes has been upgraded from version 1.05 to 1.06.
- Carp has been upgraded from version 1.42 to 1.50.

  If a package on the call stack contains a constant named `ISA`, Carp no longer throws a ‘‘Not a GLOB reference’’ error.

  Carp, when generating stack traces, now attempts to work around longstanding bugs resulting from Perl’s non-reference-counted stack. [GH #9282] <https://github.com/Perl/perl5/issues/9282>

  Carp has been modified to avoid assuming that objects cannot be overloaded without the overload module loaded (this can happen with objects created by XS modules). Previously, infinite recursion would result if an XS-defined overload method itself called Carp. [GH #16407] <https://github.com/Perl/perl5/issues/16407>

  Carp now avoids using `overload::StrVal`, partly because older versions of overload (included with perl 5.14 and earlier) load Scalar::Util at run time, which will fail if Carp has been invoked after a syntax error.
- charnames has been upgraded from version 1.44 to 1.45.
- Compress::Raw::Zlib has been upgraded from version 2.074 to 2.076.

  This addresses a security vulnerability in older versions of the ’zlib’ library (which is bundled with Compress-Raw-Zlib).
- Config::Extensions has been upgraded from version 0.01 to 0.02.
- Config::Perl::V has been upgraded from version 0.28 to 0.29.
- CPAN has been upgraded from version 2.18 to 2.20.
- Data::Dumper has been upgraded from version 2.167 to 2.170.

  Quoting of glob names now obeys the Useqq option [GH #13274] <https://github.com/Perl/perl5/issues/13274>.

  Attempts to set an option to `undef` through a combined getter/setter method are no longer mistaken for getter calls [GH #12135] <https://github.com/Perl/perl5/issues/12135>.
- Devel::Peek has been upgraded from version 1.26 to 1.27.
- Devel::PPPort has been upgraded from version 3.35 to 3.40.

  Devel::PPPort has moved from cpan-first to perl-first maintenance

  Primary responsibility for the code in Devel::PPPort has moved into core perl. In a practical sense there should be no change except that hopefully it will stay more up to date with changes made to symbols in perl, rather than needing to be updated after the fact.

- Digest::SHA has been upgraded from version 5.96 to 6.01.
- DirHandle has been upgraded from version 1.04 to 1.05.
- DynaLoader has been upgraded from version 1.42 to 1.45.

  Its documentation now shows the use of `__PACKAGE__` and direct object syntax [GH #16190] <https://github.com/Perl/perl5/issues/16190>.
- Encode has been upgraded from version 2.88 to 2.97.
- encoding has been upgraded from version 2.19 to 2.22.
- Errno has been upgraded from version 1.28 to 1.29.
- experimental has been upgraded from version 0.016 to 0.019.
- Exporter has been upgraded from version 5.72 to 5.73.
- ExtUtils::CBuilder has been upgraded from version 0.280225 to 0.280230.
- ExtUtils::Constant has been upgraded from version 0.23 to 0.25.
- ExtUtils::Embed has been upgraded from version 1.34 to 1.35.
- ExtUtils::Install has been upgraded from version 2.04 to 2.14.
- ExtUtils::MakeMaker has been upgraded from version 7.24 to 7.34.
- ExtUtils::Miniperl has been upgraded from version 1.06 to 1.08.
- ExtUtils::ParseXS has been upgraded from version 3.34 to 3.39.
- ExtUtils::Typemaps has been upgraded from version 3.34 to 3.38.
- ExtUtils::XSSymSet has been upgraded from version 1.3 to 1.4.
- feature has been upgraded from version 1.47 to 1.52.
- fields has been upgraded from version 2.23 to 2.24.
- File::Copy has been upgraded from version 2.32 to 2.33.

  It will now use the sub-second precision variant of **utime()** supplied by Time::HiRes where available. [GH #16225] <https://github.com/Perl/perl5/issues/16225>.
- File::Fetch has been upgraded from version 0.52 to 0.56.
- File::Glob has been upgraded from version 1.28 to 1.31.
- File::Path has been upgraded from version 2.12_01 to 2.15.
- File::Spec and Cwd have been upgraded from version 3.67 to 3.74.
- File::stat has been upgraded from version 1.07 to 1.08.
- FileCache has been upgraded from version 1.09 to 1.10.
- Filter::Simple has been upgraded from version 0.93 to 0.95.
- Filter::Util::Call has been upgraded from version 1.55 to 1.58.
- GDBM_File has been upgraded from version 1.15 to 1.17.

  Its documentation now explains that `each` and `delete` don't mix in hashes tied to this module [GH #12894] <https://github.com/Perl/perl5/issues/12894>.

  It will now retry opening with an acceptable block size if asking gdbm to default the block size failed [GH #13232] <https://github.com/Perl/perl5/issues/13232>.
- Getopt::Long has been upgraded from version 2.49 to 2.5.
- Hash::Util::FieldHash has been upgraded from version 1.19 to 1.20.
- I18N::Langinfo has been upgraded from version 0.13 to 0.17.

  This module is now available on all platforms, emulating the system **nl_langinfo** (3) on systems that lack it. Some caveats apply, as detailed in its documentation, the most severe being that, except for MS Windows, the `CODESET` item is not implemented on those systems, always returning `""`.

It now sets the UTF-8 flag in its returned scalar if the string contains legal non-ASCII UTF-8, and the locale is UTF-8 [GH #15131] <https://github.com/Perl/perl5/issues/15131>.

This update also fixes a bug in which the underlying locale was ignored for the `RADIXCHAR` (always was returned as a dot) and the `THOUSEP` (always empty). Now the locale-appropriate values are returned.

- I18N::LangTags has been upgraded from version 0.42 to 0.43.
- if has been upgraded from version 0.0606 to 0.0608.
- IO has been upgraded from version 1.38 to 1.39.
- IO::Socket::IP has been upgraded from version 0.38 to 0.39.
- IPC::Cmd has been upgraded from version 0.96 to 1.00.
- JSON::PP has been upgraded from version 2.27400_02 to 2.97001.
- The `libnet` distribution has been upgraded from version 3.10 to 3.11.
- List::Util has been upgraded from version 1.46_02 to 1.49.
- Locale::Codes has been upgraded from version 3.42 to 3.56.

  **NOTE**: Locale::Codes scheduled to be removed from core in Perl 5.30.
- Locale::Maketext has been upgraded from version 1.28 to 1.29.
- Math::BigInt has been upgraded from version 1.999806 to 1.999811.
- Math::BigInt::FastCalc has been upgraded from version 0.5005 to 0.5006.
- Math::BigRat has been upgraded from version 0.2611 to 0.2613.
- Module::CoreList has been upgraded from version 5.20170530 to 5.20180622.
- mro has been upgraded from version 1.20 to 1.22.
- Net::Ping has been upgraded from version 2.55 to 2.62.
- NEXT has been upgraded from version 0.67 to 0.67_01.
- ODBM_File has been upgraded from version 1.14 to 1.15.
- Opcode has been upgraded from version 1.39 to 1.43.
- overload has been upgraded from version 1.28 to 1.30.
- PerlIO::encoding has been upgraded from version 0.25 to 0.26.
- PerlIO::scalar has been upgraded from version 0.26 to 0.29.
- PerlIO::via has been upgraded from version 0.16 to 0.17.
- Pod::Functions has been upgraded from version 1.11 to 1.13.
- Pod::Html has been upgraded from version 1.2202 to 1.24.

  A title for the HTML document will now be automatically generated by default from a ''NAME'' section in the POD document, as it used to be before the module was rewritten to use Pod::Simple::XHTML to do the core of its job [GH #11954] <https://github.com/Perl/perl5/issues/11954>.
- Pod::Perldoc has been upgraded from version 3.28 to 3.2801.
- The `podlators` distribution has been upgraded from version 4.09 to 4.10.

  Man page references and function names now follow the Linux man page formatting standards, instead of the Solaris standard.
- POSIX has been upgraded from version 1.76 to 1.84.

  Some more cautions were added about using locale-specific functions in threaded applications.
- re has been upgraded from version 0.34 to 0.36.

- Scalar::Util has been upgraded from version 1.46_02 to 1.50.
- SelfLoader has been upgraded from version 1.23 to 1.25.
- Socket has been upgraded from version 2.020_03 to 2.027.
- sort has been upgraded from version 2.02 to 2.04.
- Storable has been upgraded from version 2.62 to 3.08.
- Sub::Util has been upgraded from version 1.48 to 1.49.
- subs has been upgraded from version 1.02 to 1.03.
- Sys::Hostname has been upgraded from version 1.20 to 1.22.
- Term::ReadLine has been upgraded from version 1.16 to 1.17.
- Test has been upgraded from version 1.30 to 1.31.
- Test::Harness has been upgraded from version 3.38 to 3.42.
- Test::Simple has been upgraded from version 1.302073 to 1.302133.
- threads has been upgraded from version 2.15 to 2.22.

  The documentation now better describes the problems that arise when returning values from threads, and no longer warns about creating threads in `BEGIN` blocks. [GH #11563] <https://github.com/Perl/perl5/issues/11563>
- threads::shared has been upgraded from version 1.56 to 1.58.
- Tie::Array has been upgraded from version 1.06 to 1.07.
- Tie::StdHandle has been upgraded from version 4.4 to 4.5.
- Time::gmtime has been upgraded from version 1.03 to 1.04.
- Time::HiRes has been upgraded from version 1.9741 to 1.9759.
- Time::localtime has been upgraded from version 1.02 to 1.03.
- Time::Piece has been upgraded from version 1.31 to 1.3204.
- Unicode::Collate has been upgraded from version 1.19 to 1.25.
- Unicode::Normalize has been upgraded from version 1.25 to 1.26.
- Unicode::UCD has been upgraded from version 0.68 to 0.70.

  The function `num` now accepts an optional parameter to help in diagnosing error returns.
- User::grent has been upgraded from version 1.01 to 1.02.
- User::pwent has been upgraded from version 1.00 to 1.01.
- utf8 has been upgraded from version 1.19 to 1.21.
- vars has been upgraded from version 1.03 to 1.04.
- version has been upgraded from version 0.9917 to 0.9923.
- VMS::DCLsym has been upgraded from version 1.08 to 1.09.
- VMS::Stdio has been upgraded from version 2.41 to 2.44.
- warnings has been upgraded from version 1.37 to 1.42.

  It now includes new functions with names ending in `_at_level`, allowing callers to specify the exact call frame. [GH #16257] <https://github.com/Perl/perl5/issues/16257>
- XS::Typemap has been upgraded from version 0.15 to 0.16.
- XSLoader has been upgraded from version 0.27 to 0.30.

  Its documentation now shows the use of `__PACKAGE__`, and direct object syntax for example `DynaLoader` usage [GH #16190] <https://github.com/Perl/perl5/issues/16190>.

  Platforms that use `mod2fname` to edit the names of loadable libraries now look for bootstrap (.bs) files under the correct, non-edited name.

### Removed Modules and Pragmata

- The `VMS::stdio` compatibility shim has been removed.

## Documentation

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, send email to perlbug@perl.org <mailto:perlbug@perl.org>.

Additionally, the following selected changes have been made:

*perlapi*

- The API functions `perl_parse()`, `perl_run()`, and `perl_destruct()` are now documented comprehensively, where previously the only documentation was a reference to the perlembed tutorial.
- The documentation of `newGIVENOP()` has been belatedly updated to account for the removal of lexical `$_`.
- The API functions `newCONSTSUB()` and `newCONSTSUB_flags()` are documented much more comprehensively than before.

*perldata*

- The section ‘‘Truth and Falsehood’’ in perlsyn has been moved into perldata.

*perldebguts*

- The description of the conditions under which `DB::sub()` will be called has been clarified. [GH #16055] <https://github.com/Perl/perl5/issues/16055>

*perldiag*

- ‘‘Variable length lookbehind not implemented in regex m/%s/’’ in perldiag

  This now gives more ideas as to workarounds to the issue that was introduced in Perl 5.18 (but not documented explicitly in its perldelta) for the fact that some Unicode `/i` rules cause a few sequences such as

  ```
  (?<!st)
  ```

  to be considered variable length, and hence disallowed.

- ‘‘Use of state `$_` is experimental’’ in perldiag

  This entry has been removed, as the experimental support of this construct was removed in perl 5.24.0.

- The diagnostic `Initialization of state variables in list context currently forbidden` has changed to `Initialization of state variables in list currently forbidden`, because list-context initialization of single aggregate state variables is now permitted.

*perlembed*

- The examples in perlembed have been made more portable in the way they exit, and the example that gets an exit code from the embedded Perl interpreter now gets it from the right place. The examples that pass a constructed argv to Perl now show the mandatory null `argv[argc]`.
- An example in perlembed used the string value of ERRSV as a format string when calling **croak()**. If that string contains format codes such as `%s` this could crash the program.

  This has been changed to a call to **croak_sv()**.

  An alternative could have been to supply a trivial format string:

  ```
  croak("%s", SvPV_nolen(ERRSV));
  ```

  or as a special case for ERRSV simply:

  ```
  croak(NULL);
  ```

*perlfunc*

- There is now a note that warnings generated by built-in functions are documented in perldiag and warnings. [GH #12642] <https://github.com/Perl/perl5/issues/12642>
- The documentation for the `exists` operator no longer says that autovivification behaviour ''may be fixed in a future release''. We've determined that we're not going to change the default behaviour. [GH #15231] <https://github.com/Perl/perl5/issues/15231>
- A couple of small details in the documentation for the `bless` operator have been clarified. [GH #14684] <https://github.com/Perl/perl5/issues/14684>
- The description of `@INC` hooks in the documentation for `require` has been corrected to say that filter subroutines receive a useless first argument. [GH #12569] <https://github.com/Perl/perl5/issues/12569>
- The documentation of `ref` has been rewritten for clarity.
- The documentation of `use` now explains what syntactically qualifies as a version number for its module version checking feature.
- The documentation of `warn` has been updated to reflect that since Perl 5.14 it has treated complex exception objects in a manner equivalent to `die`. [GH #13641] <https://github.com/Perl/perl5/issues/13641>
- The documentation of `die` and `warn` has been revised for clarity.
- The documentation of `each` has been improved, with a slightly more explicit description of the sharing of iterator state, and with caveats regarding the fragility of while-each loops. [GH #16334] <https://github.com/Perl/perl5/issues/16334>
- Clarification to `require` was added to explain the differences between

```
require Foo::Bar;
require "Foo/Bar.pm";
```

*perlgit*

- The precise rules for identifying `smoke-me` branches are now stated.

*perlguts*

- The section on reference counting in perlguts has been heavily revised, to describe references in the way a programmer needs to think about them rather than in terms of the physical data structures.
- Improve documentation related to UTF-8 multibytes.

*perlintern*

- The internal functions `newXS_len_flags()` and `newATTRSUB_x()` are now documented.

*perlobj*

- The documentation about `DESTROY` methods has been corrected, updated, and revised, especially in regard to how they interact with exceptions. [GH #14083] <https://github.com/Perl/perl5/issues/14083>

*perlop*

- The description of the `x` operator in perlop has been clarified. [GH #16253] <https://github.com/Perl/perl5/issues/16253>
- perlop has been updated to note that `qw`'s whitespace rules differ from that of `split`'s in that only ASCII whitespace is used.
- The general explanation of operator precedence and associativity has been corrected and clarified. [GH #15153] <https://github.com/Perl/perl5/issues/15153>
- The documentation for the `\` referencing operator now explains the unusual context that it supplies to its operand. [GH #15932] <https://github.com/Perl/perl5/issues/15932>

*perlrequick*

- Clarifications on metacharacters and character classes

*perlretut*

- Clarify metacharacters.

*perlrun*

- Clarify the differences between **-M** and **-m**. [GH #15998] <https://github.com/Perl/perl5/issues/15998>

*perlsec*

- The documentation about set-id scripts has been updated and revised. [GH #10289] <https://github.com/Perl/perl5/issues/10289>
- A section about using `sudo` to run Perl scripts has been added.

*perlsyn*

- The section ‘‘Truth and Falsehood’’ in perlsyn has been removed from that document, where it didn’t belong, and merged into the existing paragraph on the same topic in perldata.
- The means to disambiguate between code blocks and hash constructors, already documented in perlref, are now documented in perlsyn too. [GH #15918] <https://github.com/Perl/perl5/issues/15918>

*perluniprops*

- perluniprops has been updated to note that `\p{Word}` now includes code points matching the `\p{Join_Control}` property. The change to the property was made in Perl 5.18, but not documented until now. There are currently only two code points that match this property U+200C (ZERO WIDTH NON-JOINER) and U+200D (ZERO WIDTH JOINER).
- For each binary table or property, the documentation now includes which characters in the range `\x00-\xFF` it matches, as well as a list of the first few ranges of code points matched above that.

*perlvar*

- The entry for `$+` in perlvar has been expanded upon to describe handling of multiply-named capturing groups.

*perlfunc, perlop, perlsyn*

- In various places, improve the documentation of the special cases in the condition expression of a while loop, such as implicit `defined` and assignment to `$_`. [GH #16334] <https://github.com/Perl/perl5/issues/16334>

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Can’t ‘‘goto’’ into a ‘‘given’’ block

  (F) A ‘‘goto’’ statement was executed to jump into the middle of a `given` block. You can’t get there from here. See ‘‘goto’’ in perlfunc.

- Can’t ‘‘goto’’ into a binary or list expression

  Use of `goto` to jump into the parameter of a binary or list operator has been prohibited, to prevent crashes and stack corruption. [GH #15914] <https://github.com/Perl/perl5/issues/15914>

  You may only enter the *first* argument of an operator that takes a fixed number of arguments, since this is a case that will not cause stack corruption. [GH #16415] <https://github.com/Perl/perl5/issues/16415>

*New Warnings*

- Old package separator used in string

  (W syntax) You used the old package separator, ‘’’’, in a variable named inside a double-quoted

string; e.g., `"In $name's house"`. This is equivalent to `"In $name::s house"`. If you meant the former, put a backslash before the apostrophe (`"In $name\'s house"`).

- ‘‘Locale ’%s’ contains (at least) the following characters which have unexpected meanings: `%s` The Perl program will use the expected meanings’’ in perldiag

### Changes to Existing Diagnostics

- A false-positive warning that was issued when using a numerically-quantified sub-pattern in a recursive regex has been silenced. [GH #16106] <https://github.com/Perl/perl5/issues/16106>
- The warning about useless use of a concatenation operator in void context is now generated for expressions with multiple concatenations, such as `$a.$b.$c`, which used to mistakenly not warn. [GH #3990] <https://github.com/Perl/perl5/issues/3990>
- Warnings that a variable or subroutine ‘‘masks earlier declaration in same ...’’, or that an `our` variable has been redeclared, have been moved to a new warnings category ‘‘shadow’’. Previously they were in category ‘‘misc’’.
- The deprecation warning from `Sys::Hostname::hostname()` saying that it doesn’t accept arguments now states the Perl version in which the warning will be upgraded to an error. [GH #14662] <https://github.com/Perl/perl5/issues/14662>
- The perldiag entry for the error regarding a set-id script has been expanded to make clear that the error is reporting a specific security vulnerability, and to advise how to fix it.
- The `Unable to flush stdout` error message was missing a trailing newline. [debian #875361]

## Utility Changes

### perlbug

- `--help` and `--version` options have been added.

## Configuration and Compilation

- C89 requirement

  Perl has been documented as requiring a C89 compiler to build since October 1998. A variety of simplifications have now been made to Perl’s internals to rely on the features specified by the C89 standard. We believe that this internal change hasn’t altered the set of platforms that Perl builds on, but please report a bug if Perl now has new problems building on your platform.

- On GCC, `-Werror=pointer-arith` is now enabled by default, disallowing arithmetic on void and function pointers.
- Where an HTML version of the documentation is installed, the HTML documents now use relative links to refer to each other. Links from the index page of perlipc to the individual section documents are now correct. [GH #11941] <https://github.com/Perl/perl5/issues/11941>
- *lib/unicore/mktables* now correctly canonicalizes the names of the dependencies stored in the files it generates.

  *regen/mk_invlists.pl*, unlike the other *regen/*.pl* scripts, used `$0` to name itself in the dependencies stored in the files it generates. It now uses a literal so that the path stored in the generated files doesn’t depend on how *regen/mk_invlists.pl* is invoked.

  This lack of canonical names could cause test failures in *t/porting/regen.t*. [GH #16446] <https://github.com/Perl/perl5/issues/16446>

- New probes

  HAS_BUILTIN_ADD_OVERFLOW
  HAS_BUILTIN_MUL_OVERFLOW
  HAS_BUILTIN_SUB_OVERFLOW
  HAS_THREAD_SAFE_NL_LANGINFO_L
  HAS_LOCALECONV_L
  HAS_MBRLEN
  HAS_MBRTOWC
  HAS_MEMRCHR

HAS_NANOSLEEP
HAS_STRNLEN
HAS_STRTOLD_L
I_WCHAR

## Testing

- Testing of the XS-APItest directory is now done in parallel, where applicable.
- Perl now includes a default *.travis.yml* file for Travis CI testing on github mirrors. [GH #14558] <https://github.com/Perl/perl5/issues/14558>
- The watchdog timer count in *re/pat_psycho.t* can now be overridden.

  This test can take a long time to run, so there is a timer to keep this in check (currently, 5 minutes). This commit adds checking the environment variable `PERL_TEST_TIME_OUT_FACTOR`; if set, the time out setting is multiplied by its value.
- *harness* no longer waits for 30 seconds when running *t/io/openpid.t*. [GH #13535] <https://github.com/Perl/perl5/issues/13535> [GH #16420] <https://github.com/Perl/perl5/issues/16420>

## Packaging

For the past few years we have released perl using three different archive formats: bzip (`.bz2`), LZMA2 (`.xz`) and gzip (`.gz`). Since xz compresses better and decompresses faster, and gzip is more compatible and uses less memory, we have dropped the `.bz2` archive format with this release. (If this poses a problem, do let us know; see ''Reporting Bugs'', below.)

## Platform Support

### Discontinued Platforms

PowerUX / Power MAX OS
: Compiler hints and other support for these apparently long-defunct platforms has been removed.

### Platform-Specific Notes

CentOS
: Compilation on CentOS 5 is now fixed.

Cygwin
: A build with the quadmath library can now be done on Cygwin.

Darwin
: Perl now correctly uses reentrant functions, like `asctime_r`, on versions of Darwin that have support for them.

FreeBSD
: FreeBSD's */usr/share/mk/sys.mk* specifies `-O2` for architectures other than ARM and MIPS. By default, perl is now compiled with the same optimization levels.

VMS
: Several fix-ups for *configure.com*, marking function VMS has (or doesn't have).

  CRTL features can now be set by embedders before invoking Perl by using the `decc$feature_set` and `decc$feature_set_value` functions. Previously any attempt to set features after image initialization were ignored.

Windows
: - Support for compiling perl on Windows using Microsoft Visual Studio 2017 (containing Visual C++ 14.1) has been added.
  - Visual C++ compiler version detection has been improved to work on non-English language systems.
  - We now set `$Config{libpth}` correctly for 64-bit builds using Visual C++ versions earlier than 14.1.

## Internal Changes

- A new optimisation phase has been added to the compiler, `optimize_optree()`, which does a top-down scan of a complete optree just before the peephole optimiser is run. This phase is not currently hookable.

- An `OP_MULTICONCAT` op has been added. At `optimize_optree()` time, a chain of `OP_CONCAT` and `OP_CONST` ops, together optionally with an `OP_STRINGIFY` and/or `OP_SASSIGN`, are combined into a single `OP_MULTICONCAT` op. The op is of type `UNOP_AUX`, and the aux array contains the argument count, plus a pointer to a constant string and a set of segment lengths. For example with

  ```
  my $x = "foo=$foo, bar=$bar\n";
  ```

  the constant string would be `"foo=, bar=\n"` and the segment lengths would be (4,6,1). If the string contains characters such as `\x80`, whose representation changes under utf8, two sets of strings plus lengths are precomputed and stored.

- Direct access to `PL_keyword_plugin` is not safe in the presence of multithreading. A new `wrap_keyword_plugin` function has been added to allow XS modules to safely define custom keywords even when loaded from a thread, analogous to `PL_check` / `wrap_op_checker`.

- The `PL_statbuf` interpreter variable has been removed.

- The deprecated function `to_utf8_case()`, accessible from XS code, has been removed.

- A new function `is_utf8_invariant_string_loc()` has been added that is like `is_utf8_invariant_string()` but takes an extra pointer parameter into which is stored the location of the first variant character, if any are found.

- A new function, `Perl_langinfo()` has been added. It is an (almost) drop-in replacement for the system `nl_langinfo(3)`, but works on platforms that lack that; as well as being more thread-safe, and hiding some gotchas with locale handling from the caller. Code that uses this, needn't use `localeconv(3)` (and be affected by the gotchas) to find the decimal point, thousands separator, or currency symbol. See ''Perl_langinfo'' in perlapi.

- A new API function `sv_rvunweaken()` has been added to complement `sv_rvweaken()`. The implementation was taken from ''unweaken'' in Scalar::Util.

- A new flag, `SORTf_UNSTABLE`, has been added. This will allow a future commit to make mergesort unstable when the user specifies Xno sort stableX, since it has been decided that mergesort should remain stable by default.

- XS modules can now automatically get reentrant versions of system functions on threaded perls.

  By adding

  ```
  #define PERL_REENTRANT
  ```

  near the beginning of an `XS` file, it will be compiled so that whatever reentrant functions perl knows about on that system will automatically and invisibly be used instead of the plain, non-reentrant versions. For example, if you write `getpwnam()` in your code, on a system that has `getpwnam_r()` all calls to the former will be translated invisibly into the latter. This does not happen except on threaded perls, as they aren't needed otherwise. Be aware that which functions have reentrant versions varies from system to system.

- The `PERL_NO_OP_PARENT` build define is no longer supported, which means that perl is now always built with `PERL_OP_PARENT` enabled.

- The format and content of the non−utf8 transliteration table attached to the `op_pv` field of `OP_TRANS`/`OP_TRANSR` ops has changed. It's now a `struct OPtrans_map`.

- A new compiler `#define`, `dTHX_DEBUGGING`. has been added. This is useful for XS or C code that only need the thread context because their debugging statements that get compiled only under `−DDEBUGGING` need one.

- A new API function ''Perl_setlocale'' in perlapi has been added.

- ''sync_locale'' in perlapi has been revised to return a boolean as to whether the system was using the global locale or not.

- A new kind of magic scalar, called a ''nonelem'' scalar, has been introduced. It is stored in an array to denote a non-existent element, whenever such an element is accessed in a potential lvalue context. It replaces the existing ''defelem'' (deferred element) magic wherever this is possible, being significantly more efficient. This means that `some_sub($sparse_array[$nonelem])` no longer has to create a new magic defelem

scalar each time, as long as the element is within the array.

It partially fixes the rare bug of deferred elements getting out of synch with their arrays when the array is shifted or unshifted. [GH #16364] <https://github.com/Perl/perl5/issues/16364>

## Selected Bug Fixes

- List assignment (`aassign`) could in some rare cases allocate an entry on the mortals stack and leave the entry uninitialized, leading to possible crashes. [GH #16017] <https://github.com/Perl/perl5/issues/16017>
- Attempting to apply an attribute to an `our` variable where a function of that name already exists could result in a NULL pointer being supplied where an SV was expected, crashing perl. [perl #131597] <https://rt.perl.org/Ticket/Display.html?id=131597>
- `split ' '` now correctly handles the argument being split when in the scope of the `unicode_strings` feature. Previously, when a string using the single-byte internal representation contained characters that are whitespace by Unicode rules but not by ASCII rules, it treated those characters as part of fields rather than as field separators. [GH #15904] <https://github.com/Perl/perl5/issues/15904>
- Several built-in functions previously had bugs that could cause them to write to the internal stack without allocating room for the item being written. In rare situations, this could have led to a crash. These bugs have now been fixed, and if any similar bugs are introduced in future, they will be detected automatically in debugging builds.

  These internal stack usage checks introduced are also done by the `entersub` operator when calling XSUBs. This means we can report which XSUB failed to allocate enough stack space. [GH #16126] <https://github.com/Perl/perl5/issues/16126>
- Using a symbolic ref with postderef syntax as the key in a hash lookup was yielding an assertion failure on debugging builds. [GH #16029] <https://github.com/Perl/perl5/issues/16029>
- Array and hash variables whose names begin with a caret now admit indexing inside their curlies when interpolated into strings, as in `"${^CAPTURE[0]}"` to index `@{^CAPTURE}`. [GH #16050] <https://github.com/Perl/perl5/issues/16050>
- Fetching the name of a glob that was previously UTF-8 but wasn't any longer would return that name flagged as UTF-8. [GH #15971] <https://github.com/Perl/perl5/issues/15971>
- The perl `sprintf()` function (via the underlying C function `Perl_sv_vcatpvfn_flags()`) has been heavily reworked to fix many minor bugs, including the integer wrapping of large width and precision specifiers and potential buffer overruns. It has also been made faster in many cases.
- Exiting from an `eval`, whether normally or via an exception, now always frees temporary values (possibly calling destructors) *before* setting `$@`. For example:

  ```
  sub DESTROY { eval { die "died in DESTROY"; } }
  eval { bless []; };
  # $@ used to be equal to "died in DESTROY" here; it's now "".
  ```
- Fixed a duplicate symbol failure with `-flto -mieee-fp` builds. *pp.c* defined `_LIB_VERSION` which `-lieee` already defines. [GH #16086] <https://github.com/Perl/perl5/issues/16086>
- The tokenizer no longer consumes the exponent part of a floating point number if it's incomplete. [GH #16073] <https://github.com/Perl/perl5/issues/16073>
- On non-threaded builds, for `m/$null/` where `$null` is an empty string is no longer treated as if the `/o` flag was present when the previous matching match operator included the `/o` flag. The rewriting used to implement this behavior could confuse the interpreter. This matches the behaviour of threaded builds. [GH #14668] <https://github.com/Perl/perl5/issues/14668>
- Parsing a `sub` definition could cause a use after free if the `sub` keyword was followed by whitespace including newlines (and comments.) [GH #16097] <https://github.com/Perl/perl5/issues/16097>

- The tokenizer now correctly adjusts a parse pointer when skipping whitespace in a `${identifier}` construct. [perl #131949] <https://rt.perl.org/Public/Bug/Display.html?id=131949>
- Accesses to `${^LAST_FH}` no longer assert after using any of a variety of I/O operations on a non-glob. [GH #15372] <https://github.com/Perl/perl5/issues/15372>
- The XS-level `Copy()`, `Move()`, `Zero()` macros and their variants now assert if the pointers supplied are `NULL`. ISO C considers supplying NULL pointers to the functions these macros are built upon as undefined behaviour even when their count parameters are zero. Based on these assertions and the original bug report three macro calls were made conditional. [GH #16079] <https://github.com/Perl/perl5/issues/16079> [GH #16112] <https://github.com/Perl/perl5/issues/16112>
- Only the `=` operator is permitted for defining defaults for parameters in subroutine signatures. Previously other assignment operators, e.g. `+=`, were also accidentally permitted. [GH #16084] <https://github.com/Perl/perl5/issues/16084>
- Package names are now always included in `:prototype` warnings [perl #131833] <https://rt.perl.org/Public/Bug/Display.html?id=131833>
- The `je_old_stack_hwm` field, previously only found in the `jmpenv` structure on debugging builds, has been added to non-debug builds as well. This fixes an issue with some CPAN modules caused by the size of this structure varying between debugging and non-debugging builds. [GH #16122] <https://github.com/Perl/perl5/issues/16122>
- The arguments to the `ninstr()` macro are now correctly parenthesized.
- A NULL pointer dereference in the `S_regmatch()` function has been fixed. [perl #132017] <https://rt.perl.org/Public/Bug/Display.html?id=132017>
- Calling exec PROGRAM LIST with an empty `LIST` has been fixed. This should call `execvp()` with an empty `argv` array (containing only the terminating `NULL` pointer), but was instead just returning false (and not setting `$!`). [GH #16075] <https://github.com/Perl/perl5/issues/16075>
- The `gv_fetchmeth_sv` C function stopped working properly in Perl 5.22 when fetching a constant with a UTF-8 name if that constant subroutine was stored in the stash as a simple scalar reference, rather than a full typeglob. This has been corrected.
- Single-letter debugger commands followed by an argument which starts with punctuation (e.g. `p$^V` and `x@ARGV`) now work again. They had been wrongly requiring a space between the command and the argument. [GH #13342] <https://github.com/Perl/perl5/issues/13342>
- splice now throws an exception (''Modification of a read-only value attempted'') when modifying a read-only array. Until now it had been silently modifying the array. The new behaviour is consistent with the behaviour of push and unshift. [GH #15923] <https://github.com/Perl/perl5/issues/15923>
- `stat()`, `lstat()`, and file test operators now fail if given a filename containing a nul character, in the same way that `open()` already fails.
- `stat()`, `lstat()`, and file test operators now reliably set `$!` when failing due to being applied to a closed or otherwise invalid file handle.
- File test operators for Unix permission bits that don't exist on a particular platform, such as `-k` (sticky bit) on Windows, now check that the file being tested exists before returning the blanket false result, and yield the appropriate errors if the argument doesn't refer to a file.
- Fixed a 'read before buffer' overrun when parsing a range starting with `\N{}` at the beginning of the character set for the transliteration operator. [GH #16189] <https://github.com/Perl/perl5/issues/16189>
- Fixed a leaked scalar when parsing an empty `\N{}` at compile-time. [GH #16189] <https://github.com/Perl/perl5/issues/16189>
- Calling `do $path` on a directory or block device now yields a meaningful error code in `$!`. [GH #14841] <https://github.com/Perl/perl5/issues/14841>

- Regexp substitution using an overloaded replacement value that provides a tainted stringification now correctly taints the resulting string. [GH #12495] <https://github.com/Perl/perl5/issues/12495>
- Lexical sub declarations in `do` blocks such as `do { my sub lex; 123 }` could corrupt the stack, erasing items already on the stack in the enclosing statement. This has been fixed. [GH #16243] <https://github.com/Perl/perl5/issues/16243>
- `pack` and `unpack` can now handle repeat counts and lengths that exceed two billion. [GH #13179] <https://github.com/Perl/perl5/issues/13179>
- Digits past the radix point in octal and binary floating point literals now have the correct weight on platforms where a floating point significand doesn't fit into an integer type.
- The canonical truth value no longer has a spurious special meaning as a callable subroutine. It used to be a magic placeholder for a missing `import` or `unimport` method, but is now treated like any other string `1`. [GH #14902] <https://github.com/Perl/perl5/issues/14902>
- `system` now reduces its arguments to strings in the parent process, so any effects of stringifying them (such as overload methods being called or warnings being emitted) are visible in the way the program expects. [GH #13561] <https://github.com/Perl/perl5/issues/13561>
- The `readpipe()` built-in function now checks at compile time that it has only one parameter expression, and puts it in scalar context, thus ensuring that it doesn't corrupt the stack at runtime. [GH #2793] <https://github.com/Perl/perl5/issues/2793>
- `sort` now performs correct reference counting when aliasing `$a` and `$b`, thus avoiding premature destruction and leakage of scalars if they are re-aliased during execution of the sort comparator. [GH #11422] <https://github.com/Perl/perl5/issues/11422>
- `reverse` with no operand, reversing `$_` by default, is no longer in danger of corrupting the stack. [GH #16291] <https://github.com/Perl/perl5/issues/16291>
- `exec`, `system`, et al are no longer liable to have their argument lists corrupted by reentrant calls and by magic such as tied scalars. [GH #15660] <https://github.com/Perl/perl5/issues/15660>
- Perl's own `malloc` no longer gets confused by attempts to allocate more than a gigabyte on a 64−bit platform. [GH #13273] <https://github.com/Perl/perl5/issues/13273>
- Stacked file test operators in a sort comparator expression no longer cause a crash. [GH #15626] <https://github.com/Perl/perl5/issues/15626>
- An identity `tr///` transformation on a reference is no longer mistaken for that reference for the purposes of deciding whether it can be assigned to. [GH #15812] <https://github.com/Perl/perl5/issues/15812>
- Lengthy hexadecimal, octal, or binary floating point literals no longer cause undefined behaviour when parsing digits that are of such low significance that they can't affect the floating point value. [GH #16114] <https://github.com/Perl/perl5/issues/16114>
- `open $$scalarref...` and similar invocations no longer leak the file handle. [GH #12593] <https://github.com/Perl/perl5/issues/12593>
- Some convoluted kinds of regexp no longer cause an arithmetic overflow when compiled. [GH #16113] <https://github.com/Perl/perl5/issues/16113>
- The default typemap, by avoiding `newGVgen`, now no longer leaks when XSUBs return file handles (`PerlIO *` or `FILE *`). [GH #12593] <https://github.com/Perl/perl5/issues/12593>
- Creating a `BEGIN` block as an XS subroutine with a prototype no longer crashes because of the early freeing of the subroutine.
- The `printf` format specifier `%.0f` no longer rounds incorrectly [GH #9125] <https://github.com/Perl/perl5/issues/9125>, and now shows the correct sign for a negative zero.
- Fixed an issue where the error `Scalar value @arrayname[0] better written as $arrayname` would give an error `Cannot printf Inf with 'c'` when arrayname starts with `Inf`. [GH #16335] <https://github.com/Perl/perl5/issues/16335>

- The Perl implementation of `getcwd()` in `Cwd` in the PathTools distribution now behaves the same as XS implementation on errors: it returns an error, and sets `$!`. [GH #16338] <https://github.com/Perl/perl5/issues/16338>
- Vivify array elements when putting them on the stack. Fixes [GH #5310] <https://github.com/Perl/perl5/issues/5310> (reported in April 2002).
- Fixed parsing of braced subscript after parens. Fixes [GH #4688] <https://github.com/Perl/perl5/issues/4688> (reported in December 2001).
- `tr/non_utf8/long_non_utf8/c` could give the wrong results when the length of the replacement character list was greater than 0x7fff.
- `tr/non_utf8/non_utf8/cd` failed to add the implied `\x{100}-\x{7fffffff}` to the search character list.
- Compilation failures within ‘‘perl-within-perl’’ constructs, such as with string interpolation and the right part of `s///e`, now cause compilation to abort earlier.

  Previously compilation could continue in order to report other errors, but the failed sub-parse could leave partly parsed constructs on the parser shift-reduce stack, confusing the parser, leading to perl crashes. [GH #14739] <https://github.com/Perl/perl5/issues/14739>
- On threaded perls where the decimal point (radix) character is not a dot, it has been possible for a race to occur between threads when one needs to use the real radix character (such as with `sprintf`). This has now been fixed by use of a mutex on systems without thread-safe locales, and the problem just doesn’t come up on those with thread-safe locales.
- Errors while compiling a regex character class could sometime trigger an assertion failure. [GH #16172] <https://github.com/Perl/perl5/issues/16172>

## Acknowledgements

Perl 5.28.0 represents approximately 13 months of development since Perl 5.26.0 and contains approximately 730,000 lines of changes across 2,200 files from 77 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 580,000 lines of changes to 1,300 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.28.0:

Aaron Crane, Abigail, Ævar Arnfjörð Bjarmason, Alberto Simões, Alexandr Savca, Andrew Fresh, Andy Dougherty, Andy Lester, Aristotle Pagaltzis, Ask Bjørn Hansen, Chris ’BinGOs’ Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, Dan Collins, Daniel Dragan, David Cantrell, David Mitchell, Dmitry Ulanov, Dominic Hargreaves, E. Choroba, Eric Herman, Eugen Konkov, Father Chrysostomos, Gene Sullivan, George Hartzell, Graham Knop, Harald Jörg, H.Merijn Brand, Hugo van der Sanden, Jacques Germishuys, James E Keenan, Jarkko Hietaniemi, Jerry D. Hedden, J. Nick Koston, John Lightsey, John Peacock, John P. Linderman, John SJ Anderson, Karen Etheridge, Karl Williamson, Ken Brown, Ken Cotterill, Leon Timmermans, Lukas Mai, Marco Fontani, Marc-Philip Werner, Matthew Horsfall, Neil Bowers, Nicholas Clark, Nicolas R., Niko Tyni, Pali, Paul Marquess, Peter John Acklam, Reini Urban, Renee Baecker, Ricardo Signes, Robin Barker, Sawyer X, Scott Lanning, Sergey Aleynikov, Shirakata Kentaro, Shoichi Kaji, Slaven Rezic, Smylers, Steffen Müller, Steve Hay, Sullivan Beck, Thomas Sibley, Todd Rinaldo, Tomasz Konojacki, Tom Hukins, Tom Wyant, Tony Cook, Vitali Peil, Yves Orton, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5281delta − what is new for perl v5.28.1

## DESCRIPTION

This document describes differences between the 5.28.0 release and the 5.28.1 release.

If you are upgrading from an earlier release such as 5.26.0, first read perl5280delta, which describes differences between 5.26.0 and 5.28.0.

## Security

### [CVE−2018−18311] Integer overflow leading to buffer overflow and segmentation fault

Integer arithmetic in `Perl_my_setenv()` could wrap when the combined length of the environment variable name and value exceeded around 0x7fffffff. This could lead to writing beyond the end of an allocated buffer with attacker supplied data.

[GH #16560] <https://github.com/Perl/perl5/issues/16560>

### [CVE−2018−18312] Heap-buffer-overflow write in S_regatom (regcomp.c)

A crafted regular expression could cause heap-buffer-overflow write during compilation, potentially allowing arbitrary code execution.

[GH #16649] <https://github.com/Perl/perl5/issues/16649>

## Incompatible Changes

There are no changes intentionally incompatible with 5.28.0. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20180622 to 5.20181129_28.

## Selected Bug Fixes

- Perl 5.28 introduced an `index()` optimization when comparing to −1 (or indirectly, e.g. >= 0). When this optimization was triggered inside a `when` clause it caused a warning (‘‘Argument `%s` isn’t numeric in smart match’’). This has now been fixed. [GH #16626] <https://github.com/Perl/perl5/issues/16626>
- Matching of decimal digits in script runs, introduced in Perl 5.28, had a bug that led to `"1\N{THAI DIGIT FIVE}"` matching `/^(*sr:\d+)$/` when it should not. This has now been fixed.
- The new in-place editing code no longer leaks directory handles. [GH #16602] <https://github.com/Perl/perl5/issues/16602>

## Acknowledgements

Perl 5.28.1 represents approximately 5 months of development since Perl 5.28.0 and contains approximately 6,100 lines of changes across 44 files from 12 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 700 lines of changes to 12 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.28.1:

Aaron Crane, Abigail, Chris ’BinGOs’ Williams, Dagfinn Ilmari Mannsåker, David Mitchell, James E Keenan, John SJ Anderson, Karen Etheridge, Karl Williamson, Sawyer X, Steve Hay, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5282delta − what is new for perl v5.28.2

## DESCRIPTION

This document describes differences between the 5.28.1 release and the 5.28.2 release.

If you are upgrading from an earlier release such as 5.28.0, first read perl5281delta, which describes differences between 5.28.0 and 5.28.1.

## Incompatible Changes

### Any set of digits in the Common script are legal in a script run of another script

There are several sets of digits in the Common script. `[0−9]` is the most familiar. But there are also `[\x{FF10}−\x{FF19}]` (FULLWIDTH DIGIT ZERO − FULLWIDTH DIGIT NINE), and several sets for use in mathematical notation, such as the MATHEMATICAL DOUBLE-STRUCK DIGITs. Any of these sets should be able to appear in script runs of, say, Greek. But the previous design overlooked all but the ASCII digits `[0−9]`, so the design was flawed. This has been fixed, so is both a bug fix and an incompatibility.

All digits in a run still have to come from the same set of ten digits.

[GH #16704] <https://github.com/Perl/perl5/issues/16704>

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20181129_28 to 5.20190419.
- PerlIO::scalar has been upgraded from version 0.29 to 0.30.
- Storable has been upgraded from version 3.08 to 3.08_01.

## Platform Support

### Platform-Specific Notes

Windows

The Windows Server 2003 SP1 Platform SDK build, with its early x64 compiler and tools, was accidentally broken in Perl 5.27.9. This has now been fixed.

Mac OS X

Perl's build and testing process on Mac OS X for `−Duseshrplib` builds is now compatible with Mac OS X System Integrity Protection (SIP).

SIP prevents binaries in */bin* (and a few other places) being passed the `DYLD_LIBRARY_PATH` environment variable. For our purposes this prevents `DYLD_LIBRARY_PATH` from being passed to the shell, which prevents that variable being passed to the testing or build process, so running `perl` couldn't find *libperl.dylib*.

To work around that, the initial build of the *perl* executable expects to find *libperl.dylib* in the build directory, and the library path is then adjusted during installation to point to the installed library.

[GH #15057] <https://github.com/Perl/perl5/issues/15057>

## Selected Bug Fixes

- If an in-place edit is still in progress during global destruction and the process exit code (as stored in `$?`) is zero, perl will now treat the in-place edit as successful, replacing the input file with any output produced.

  This allows code like:

  ```
  perl −i −ne 'print "Foo"; last'
  ```

  to replace the input file, while code like:

  ```
  perl −i −ne 'print "Foo"; die'
  ```

  will not. Partly resolves [perl #133659].

  [GH #16748] <https://github.com/Perl/perl5/issues/16748>

- A regression in Perl 5.28 caused the following code to fail

  ```
  close(STDIN); open(CHILD, "|wc -l")'
  ```

  because the child's stdin would be closed on exec. This has now been fixed.

- `pack "u", "invalid uuencoding"` now properly NUL terminates the zero-length SV produced.

  [GH #16343] <https://github.com/Perl/perl5/issues/16343>

- Failing to compile a format now aborts compilation. Like other errors in sub-parses this could leave the parser in a strange state, possibly crashing perl if compilation continued.

  [GH #16169] <https://github.com/Perl/perl5/issues/16169>

- See ''Any set of digits in the Common script are legal in a script run of another script''.

## Acknowledgements

Perl 5.28.2 represents approximately 4 months of development since Perl 5.28.1 and contains approximately 2,500 lines of changes across 75 files from 13 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,200 lines of changes to 29 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.28.2:

Aaron Crane, Abigail, Andy Dougherty, David Mitchell, Karen Etheridge, Karl Williamson, Leon Timmermans, Nicolas R., Sawyer X, Steve Hay, Tina Müller, Tony Cook, Zak B. Elep.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/> . There may also be information at <http://www.perl.org/> , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5283delta − what is new for perl v5.28.3

## DESCRIPTION

This document describes differences between the 5.28.2 release and the 5.28.3 release.

If you are upgrading from an earlier release such as 5.28.1, first read perl5282delta, which describes differences between 5.28.1 and 5.28.2.

## Security

### [CVE−2020−10543] Buffer overflow caused by a crafted regular expression

A signed `size_t` integer overflow in the storage space calculations for nested regular expression quantifiers could cause a heap buffer overflow in Perl's regular expression compiler that overwrites memory allocated after the regular expression storage space with attacker supplied data.

The target system needs a sufficient amount of memory to allocate partial expansions of the nested quantifiers prior to the overflow occurring. This requirement is unlikely to be met on 64−bit systems.

Discovered by: ManhND of The Tarantula Team, VinCSS (a member of Vingroup).

### [CVE−2020−10878] Integer overflow via malformed bytecode produced by a crafted regular expression

Integer overflows in the calculation of offsets between instructions for the regular expression engine could cause corruption of the intermediate language state of a compiled regular expression. An attacker could abuse this behaviour to insert instructions into the compiled form of a Perl regular expression.

Discovered by: Hugo van der Sanden and Slaven Rezic.

### [CVE−2020−12723] Buffer overflow caused by a crafted regular expression

Recursive calls to `S_study_chunk()` by Perl's regular expression compiler to optimize the intermediate language representation of a regular expression could cause corruption of the intermediate language state of a compiled regular expression.

Discovered by: Sergey Aleynikov.

### Additional Note

An application written in Perl would only be vulnerable to any of the above flaws if it evaluates regular expressions supplied by the attacker. Evaluating regular expressions in this fashion is known to be dangerous since the regular expression engine does not protect against denial of service attacks in this usage scenario.

## Incompatible Changes

There are no changes intentionally incompatible with Perl 5.28.2. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20190419 to 5.20200601_28.

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

## Acknowledgements

Perl 5.28.3 represents approximately 13 months of development since Perl 5.28.2 and contains approximately 3,100 lines of changes across 48 files from 16 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,700 lines of changes to 9 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.28.3:

Chris 'BinGOs' Williams, Dan Book, Hugo van der Sanden, James E Keenan, John Lightsey, Karen Etheridge, Karl Williamson, Matthew Horsfall, Max Maischein, Nicolas R., Renee Baecker, Sawyer X, Steve Hay, Tom Hukins, Tony Cook, Zak B. Elep.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who

reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <https://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see "SECURITY VULNERABILITY CONTACT INFORMATION" in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5300delta − what is new for perl v5.30.0

## DESCRIPTION

This document describes differences between the 5.28.0 release and the 5.30.0 release.

If you are upgrading from an earlier release such as 5.26.0, first read perl5280delta, which describes differences between 5.26.0 and 5.28.0.

## Notice

sv_utf8_(downgrade|decode) are no longer marked as experimental. [GH #16822] <https://github.com/Perl/perl5/issues/16822>.

## Core Enhancements

### Limited variable length lookbehind in regular expression pattern matching is now experimentally supported

Using a lookbehind assertion (like `(?<=foo?)` or `(?<!ba{1,9}r)` previously would generate an error and refuse to compile. Now it compiles (if the maximum lookbehind is at most 255 characters), but raises a warning in the new `experimental::vlb` warnings category. This is to caution you that the precise behavior is subject to change based on feedback from use in the field.

See ‘‘(?<=pattern)’’ in perlre and ‘‘(?<!pattern)’’ in perlre.

### The upper limit ‘‘n’’ specifiable in a regular expression quantifier of the form ‘‘{m,n}’’ has been doubled to 65534

The meaning of an unbounded upper quantifier `"{m,}"` remains unchanged. It matches 2**31 − 1 times on most platforms, and more on ones where a C language short variable is more than 4 bytes long.

### Unicode 12.1 is supported

Because of a change in Unicode release cycles, Perl jumps from Unicode 10.0 in Perl 5.28 to Unicode 12.1 in Perl 5.30.

For details on the Unicode changes, see <https://www.unicode.org/versions/Unicode11.0.0/> for 11.0; <https://www.unicode.org/versions/Unicode12.0.0/> for 12.0; and <https://www.unicode.org/versions/Unicode12.1.0/> for 12.1. (Unicode 12.1 differs from 12.0 only in the addition of a single character, that for the new Japanese era name.)

The Word_Break property, as in past Perl releases, remains tailored to behave more in line with expectations of Perl users. This means that sequential runs of horizontal white space characters are not broken apart, but kept as a single run. Unicode 11 changed from past versions to be more in line with Perl, but it left several white space characters as causing breaks: TAB, NO BREAK SPACE, and FIGURE SPACE (U+2007). We have decided to continue to use the previous Perl tailoring with regards to these.

### Wildcards in Unicode property value specifications are now partially supported

You can now do something like this in a regular expression pattern

```
qr! \p{nv= /(?x) \A [0−5] \z / }!
```

which matches all Unicode code points whose numeric value is between 0 and 5 inclusive. So, it could match the Thai or Bengali digits whose numeric values are 0, 1, 2, 3, 4, or 5.

This marks another step in implementing the regular expression features the Unicode Consortium suggests.

Most properties are supported, with the remainder planned for 5.32. Details are in ‘‘Wildcards in Property Values’’ in perlunicode.

### qr'\N{name}' is now supported

Previously it was an error to evaluate a named character `\N{...}` within a single quoted regular expression pattern (whose evaluation is deferred from the normal place). This restriction is now removed.

### Turkic UTF−8 locales are now seamlessly supported

Turkic languages have different casing rules than other languages for the characters `"i"` and `"I"`. The uppercase of `"i"` is LATIN CAPITAL LETTER I WITH DOT ABOVE (U+0130); and the lowercase of `"I"` is LATIN SMALL LETTER DOTLESS I (U+0131). Unicode furnishes alternate casing rules for use with Turkic languages. Previously, Perl ignored these, but now, it uses them when it detects that it is

operating under a Turkic UTF-8 locale.

**It is now possible to compile perl to always use thread-safe locale operations.**
Previously, these calls were only used when the perl was compiled to be multi-threaded. To always enable them, add

```
-Accflags='-DUSE_THREAD_SAFE_LOCALE'
```

to your *Configure* flags.

**Eliminate opASSIGN macro usage from core**
This macro is still defined but no longer used in core

**`-Drv` now means something on `-DDEBUGGING` builds**
Now, adding the verbose flag (`-Dv`) to the `-Dr` flag turns on all possible regular expression debugging.

## Incompatible Changes

**Assigning non-zero to `$[` is fatal**
Setting `$[` to a non-zero value has been deprecated since Perl 5.12 and now throws a fatal error. See "Assigning non-zero to `$[` is fatal" in perldeprecation.

**Delimiters must now be graphemes**
See ‘‘Use of unassigned code point or non-standalone grapheme for a delimiter.’’ in perldeprecation

**Some formerly deprecated uses of an unescaped left brace ‘‘{’’ in regular expression patterns are now illegal**
But to avoid breaking code unnecessarily, most instances that issued a deprecation warning, remain legal and now have a non-deprecation warning raised. See ‘‘Unescaped left braces in regular expressions’’ in perldeprecation.

**Previously deprecated sysread()/syswrite() on :utf8 handles is now fatal**
Calling **sysread()**, **syswrite()**, **send()** or **recv()** on a `:utf8` handle, whether applied explicitly or implicitly, is now fatal. This was deprecated in perl 5.24.

There were two problems with calling these functions on `:utf8` handles:

- All four functions only paid attention to the `:utf8` flag. Other layers were completely ignored, so a handle with `:encoding(UTF-16LE)` layer would be treated as UTF-8. Other layers, such as compression are completely ignored with or without the `:utf8` flag.
- **sysread()** and **recv()** would read from the handle, skipping any validation by the layers, and do no validation of their own. This could lead to invalidly encoded perl scalars.

[GH #14839] <https://github.com/Perl/perl5/issues/14839>.

**my() in false conditional prohibited**
Declarations such as `my $x if 0` are no longer permitted.

[GH #16702] <https://github.com/Perl/perl5/issues/16702>.

**Fatalize $* and $#**
These special variables, long deprecated, now throw exceptions when used.

[GH #16718] <https://github.com/Perl/perl5/issues/16718>.

**Fatalize unqualified use of dump()**
The `dump()` function, long discouraged, may no longer be used unless it is fully qualified, *i.e.*, `CORE::dump()`.

[GH #16719] <https://github.com/Perl/perl5/issues/16719>.

**Remove File::Glob::glob()**
The `File::Glob::glob()` function, long deprecated, has been removed and now throws an exception which advises use of `File::Glob::bsd_glob()` instead.

[GH #16721] <https://github.com/Perl/perl5/issues/16721>.

**`pack()` no longer can return malformed UTF-8**
It croaks if it would otherwise return a UTF-8 string that contains malformed UTF-8. This protects against potential security threats. This is considered a bug fix as well. [GH #16035] <https://github.com/Perl/perl5/issues/16035>.

### Any set of digits in the Common script are legal in a script run of another script

There are several sets of digits in the Common script. `[0-9]` is the most familiar. But there are also `[\x{FF10}-\x{FF19}]` (FULLWIDTH DIGIT ZERO − FULLWIDTH DIGIT NINE), and several sets for use in mathematical notation, such as the MATHEMATICAL DOUBLE-STRUCK DIGITs. Any of these sets should be able to appear in script runs of, say, Greek. But the design of 5.30 overlooked all but the ASCII digits `[0-9]`, so the design was flawed. This has been fixed, so is both a bug fix and an incompatibility. [GH #16704] <https://github.com/Perl/perl5/issues/16704>.

All digits in a run still have to come from the same set of ten digits.

### JSON::PP enables allow_nonref by default

As JSON::XS 4.0 changed its policy and enabled allow_nonref by default, JSON::PP also enabled allow_nonref by default.

## Deprecations

### In XS code, use of various macros dealing with UTF-8.

This deprecation was scheduled to become fatal in 5.30, but has been delayed to 5.32 due to problems that showed up with some CPAN modules. For details of what's affected, see perldeprecation.

## Performance Enhancements

- Translating from UTF-8 into the code point it represents now is done via a deterministic finite automaton, speeding it up. As a typical example, `ord("\x7fff")` now requires 12% fewer instructions than before. The performance of checking that a sequence of bytes is valid UTF-8 is similarly improved, again by using a DFA.
- Eliminate recursion from **finalize_op()**. [GH #11866] <https://github.com/Perl/perl5/issues/11866>.
- A handful of small optimizations related to character folding and character classes in regular expressions.
- Optimization of `IV` to `UV` conversions. [GH #16761] <https://github.com/Perl/perl5/issues/16761>.
- Speed up of the integer stringification algorithm by processing two digits at a time instead of one. [GH #16769] <https://github.com/Perl/perl5/issues/16769>.
- Improvements based on LGTM analysis and recommendation. (<https://lgtm.com/projects/g/Perl/perl5/alerts/?mode=tree>). [GH #16765] <https://github.com/Perl/perl5/issues/16765>. [GH #16773] <https://github.com/Perl/perl5/issues/16773>.
- Code optimizations in *regcomp.c*, *regcomp.h*, *regexec.c*.
- Regular expression pattern matching of things like `qr/[^a]/` is significantly sped up, where *a* is any ASCII character. Other classes can get this speed up, but which ones is complicated and depends on the underlying bit patterns of those characters, so differs between ASCII and EBCDIC platforms, but all case pairs, like `qr/[Gg]/` are included, as is `[^01]`.

## Modules and Pragmata

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.30 to 2.32.
- B has been upgraded from version 1.74 to 1.76.
- B::Concise has been upgraded from version 1.003 to 1.004.
- B::Deparse has been upgraded from version 1.48 to 1.49.
- bignum has been upgraded from version 0.49 to 0.51.
- bytes has been upgraded from version 1.06 to 1.07.
- Carp has been upgraded from version 1.38 to 1.50
- Compress::Raw::Bzip2 has been upgraded from version 2.074 to 2.084.
- Compress::Raw::Zlib has been upgraded from version 2.076 to 2.084.

- Config::Extensions has been upgraded from version 0.02 to 0.03.
- Config::Perl::V. has been upgraded from version 0.29 to 0.32. This was due to a new configuration variable that has influence on binary compatibility: `USE_THREAD_SAFE_LOCALE`.
- CPAN has been upgraded from version 2.20 to 2.22.
- Data::Dumper has been upgraded from version 2.170 to 2.174

  Data::Dumper now avoids leaking when `croak`ing.
- DB_File has been upgraded from version 1.840 to 1.843.
- deprecate has been upgraded from version 0.03 to 0.04.
- Devel::Peek has been upgraded from version 1.27 to 1.28.
- Devel::PPPort has been upgraded from version 3.40 to 3.52.
- Digest::SHA has been upgraded from version 6.01 to 6.02.
- Encode has been upgraded from version 2.97 to 3.01.
- Errno has been upgraded from version 1.29 to 1.30.
- experimental has been upgraded from version 0.019 to 0.020.
- ExtUtils::CBuilder has been upgraded from version 0.280230 to 0.280231.
- ExtUtils::Manifest has been upgraded from version 1.70 to 1.72.
- ExtUtils::Miniperl has been upgraded from version 1.08 to 1.09.
- ExtUtils::ParseXS has been upgraded from version 3.39 to 3.40. `OUTLIST` parameters are no longer incorrectly included in the automatically generated function prototype. [GH #16746] <https://github.com/Perl/perl5/issues/16746>.
- feature has been upgraded from version 1.52 to 1.54.
- File::Copy has been upgraded from version 2.33 to 2.34.
- File::Find has been upgraded from version 1.34 to 1.36.

  `$File::Find::dont_use_nlink` now defaults to 1 on all platforms. [GH #16759] <https://github.com/Perl/perl5/issues/16759>.

  Variables `$Is_Win32` and `$Is_VMS` are being initialized.
- File::Glob has been upgraded from version 1.31 to 1.32.
- File::Path has been upgraded from version 2.15 to 2.16.
- File::Spec has been upgraded from version 3.74 to 3.78.

  Silence Cwd warning on Android builds if `targetsh` is not defined.
- File::Temp has been upgraded from version 0.2304 to 0.2309.
- Filter::Util::Call has been upgraded from version 1.58 to 1.59.
- GDBM_File has been upgraded from version 1.17 to 1.18.
- HTTP::Tiny has been upgraded from version 0.070 to 0.076.
- I18N::Langinfo has been upgraded from version 0.17 to 0.18.
- IO has been upgraded from version 1.39 to 1.40.
- IO-Compress has been upgraded from version 2.074 to 2.084.

  Adds support for `IO::Uncompress::Zstd` and `IO::Uncompress::UnLzip`.

  The `BinModeIn` and `BinModeOut` options are now no-ops. ALL files will be read/written in binmode.
- IPC::Cmd has been upgraded from version 1.00 to 1.02.
- JSON::PP has been upgraded from version 2.97001 to 4.02.

  JSON::PP as JSON::XS 4.0 enables `allow_nonref` by default.

- lib has been upgraded from version 0.64 to 0.65.
- Locale::Codes has been upgraded from version 3.56 to 3.57.
- Math::BigInt has been upgraded from version 1.999811 to 1.999816.

  `bnok()` now supports the full Kronenburg extension. [cpan #95628] <https://rt.cpan.org/Ticket/Display.html?id=95628>.
- Math::BigInt::FastCalc has been upgraded from version 0.5006 to 0.5008.
- Math::BigRat has been upgraded from version 0.2613 to 0.2614.
- Module::CoreList has been upgraded from version 5.20180622 to 5.20190520.

  Changes to B::Op_private and Config
- Module::Load has been upgraded from version 0.32 to 0.34.
- Module::Metadata has been upgraded from version 1.000033 to 1.000036.

  Properly clean up temporary directories after testing.
- NDBM_File has been upgraded from version 1.14 to 1.15.
- Net::Ping has been upgraded from version 2.62 to 2.71.
- ODBM_File has been upgraded from version 1.15 to 1.16.
- PathTools has been upgraded from version 3.74 to 3.78.
- parent has been upgraded from version 0.236 to 0.237.
- perl5db.pl has been upgraded from version 1.54 to 1.55.

  Debugging threaded code no longer deadlocks in `DB::sub` nor `DB::lsub`.
- perlfaq has been upgraded from version 5.021011 to 5.20190126.
- PerlIO::encoding has been upgraded from version 0.26 to 0.27.

  Warnings enabled by setting the `WARN_ON_ERR` flag in `$PerlIO::encoding::fallback` are now only produced if warnings are enabled with `use warnings "utf8";` or setting `$^W`.
- PerlIO::scalar has been upgraded from version 0.29 to 0.30.
- podlators has been upgraded from version 4.10 to 4.11.
- POSIX has been upgraded from version 1.84 to 1.88.
- re has been upgraded from version 0.36 to 0.37.
- SDBM_File has been upgraded from version 1.14 to 1.15.
- sigtrap has been upgraded from version 1.08 to 1.09.
- Storable has been upgraded from version 3.08 to 3.15.

  Storable no longer probes for recursion limits at build time. [GH #16780] <https://github.com/Perl/perl5/issues/16780> and others.

  Metasploit exploit code was included to test for CVE-2015-1592 detection, this caused anti-virus detections on at least one AV suite. The exploit code has been removed and replaced with a simple functional test. [GH #16778] <https://github.com/Perl/perl5/issues/16778>
- Test::Simple has been upgraded from version 1.302133 to 1.302162.
- Thread::Queue has been upgraded from version 3.12 to 3.13.
- threads::shared has been upgraded from version 1.58 to 1.60.

  Added support for extra tracing of locking, this requires a `-DDEBUGGING` and extra compilation flags.
- Time::HiRes has been upgraded from version 1.9759 to 1.9760.
- Time::Local has been upgraded from version 1.25 to 1.28.

- Time::Piece has been upgraded from version 1.3204 to 1.33.
- Unicode::Collate has been upgraded from version 1.25 to 1.27.
- Unicode::UCD has been upgraded from version 0.70 to 0.72.
- User::grent has been upgraded from version 1.02 to 1.03.
- utf8 has been upgraded from version 1.21 to 1.22.
- vars has been upgraded from version 1.04 to 1.05.

  `vars.pm` no longer disables non-vars strict when checking if strict vars is enabled. [GH #15851] <https://github.com/Perl/perl5/issues/15851>.
- version has been upgraded from version 0.9923 to 0.9924.
- warnings has been upgraded from version 1.42 to 1.44.
- XS::APItest has been upgraded from version 0.98 to 1.00.
- XS::Typemap has been upgraded from version 0.16 to 0.17.

### Removed Modules and Pragmata

The following modules will be removed from the core distribution in a future release, and will at that time need to be installed from CPAN. Distributions on CPAN which require these modules will need to list them as prerequisites.

The core versions of these modules will now issue `"deprecated"`-category warnings to alert you to this fact. To silence these deprecation warnings, install the modules in question from CPAN.

Note that these are (with rare exceptions) fine modules that you are encouraged to continue to use. Their disinclusion from core primarily hinges on their necessity to bootstrapping a fully functional, CPAN-capable Perl installation, not usually on concerns over their design.

- B::Debug is no longer distributed with the core distribution. It continues to be available on CPAN as `B::Debug <https://metacpan.org/pod/B::Debug>`.
- Locale::Codes has been removed at the request of its author. It continues to be available on CPAN as `Locale::Codes <https://metacpan.org/pod/Locale::Codes>` [GH #16660] <https://github.com/Perl/perl5/issues/16660>.

## Documentation

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, send email to perlbug@perl.org <mailto:perlbug@perl.org>.

*perlapi*

- `AvFILL()` was wrongly listed as deprecated. This has been corrected. [GH #16586] <https://github.com/Perl/perl5/issues/16586>

*perlop*

- We no longer have null (empty line) here doc terminators, so perlop should not refer to them.
- The behaviour of `tr` when the delimiter is an apostrophe has been clarified. In particular, hyphens aren't special, and `\x{}` isn't interpolated. [GH #15853] <https://github.com/Perl/perl5/issues/15853>

*perlreapi, perlvar*

- Improve docs for lastparen, lastcloseparen.

*perlfunc*

- The entry for "−X" in perlfunc has been clarified to indicate that symbolic links are followed for most tests.
- Clarification of behaviour of `reset EXPR`.
- Try to clarify that `ref(qr/xx/)` returns `Regexp` rather than `REGEXP` and why. [GH #16801] <https://github.com/Perl/perl5/issues/16801>.

*perlreref*

- Clarification of the syntax of /(?(cond)yes)/.

*perllocale*

- There are actually two slightly different types of UTF-8 locales: one for Turkic languages and one for everything else. Starting in Perl v5.30, Perl seamlessly handles both types.

*perlrecharclass*

- Added a note for the ::xdigit:: character class.

*perlvar*

- More specific documentation of paragraph mode. [GH #16787] <https://github.com/Perl/perl5/issues/16787>.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### Changes to Existing Diagnostics

- As noted under ''Incompatible Changes'' above, the deprecation warning ''Unescaped left brace in regex is deprecated here (and will be fatal in Perl 5.30), passed through in regex; marked by <-- HERE in m/%s/'' has been changed to the non-deprecation warning ''Unescaped left brace in regex is passed through in regex; marked by <-- HERE in m/%s/''.
- Specifying `\o{}` without anything between the braces now yields the fatal error message ''Empty \o{}''. Previously it was ''Number with no digits''. This means the same wording is used for this kind of error as with similar constructs such as `\p{}`.
- Within the scope of the experimental feature `use re 'strict'`, specifying `\x{}` without anything between the braces now yields the fatal error message ''Empty \x{}''. Previously it was ''Number with no digits''. This means the same wording is used for this kind of error as with similar constructs such as `\p{}`. It is legal, though not wise to have an empty `\x` outside of `re 'strict'`; it silently generates a NUL character.
- Type of arg `%d` to `%s` must be `%s` (not `%s`)

  Attempts to push, pop, etc on a hash or glob now produce this message rather than complaining that they no longer work on scalars. [GH #15774] <https://github.com/Perl/perl5/issues/15774>.
- Prototype not terminated

  The file and line number is now reported for this error. [GH #16697] <https://github.com/Perl/perl5/issues/16697>
- Under `-Dr` (or `use re 'Debug'`) the compiled regex engine program is displayed. It used to use two different spellings for *infinity*, `INFINITY`, and `INFTY`. It now uses the latter exclusively, as that spelling has been around the longest.

## Utility Changes

### xsubpp

- The generated prototype (with `PROTOTYPES: ENABLE`) would include `OUTLIST` parameters, but these aren't arguments to the perl function. This has been rectified. [GH #16746] <https://github.com/Perl/perl5/issues/16746>.

## Configuration and Compilation

- Normally the thread-safe locale functions are used only on threaded builds. It is now possible to force their use on unthreaded builds on systems that have them available, by including the `-Accflags='-DUSE_THREAD_SAFE_LOCALE'` option to *Configure*.
- Improve detection of memrchr, strlcat, and strlcpy
- Improve Configure detection of **memmem()**. [GH #16807] <https://github.com/Perl/perl5/issues/16807>.
- Multiple improvements and fixes for -DPERL_GLOBAL_STRUCT build option.
- Fix -DPERL_GLOBAL_STRUCT_PRIVATE build option.

## Testing

- *t/lib/croak/op* [GH #15774] <https://github.com/Perl/perl5/issues/15774>.

  separate error for `push`, etc. on hash/glob.

- *t/op/svleak.t* [GH #16749] <https://github.com/Perl/perl5/issues/16749>.

  Add test for `goto &sub` in overload leaking.

- Split *t/re/fold_grind.t* into multiple test files.

- Fix intermittent tests which failed due to race conditions which surface during parallel testing. [GH #16795] <https://github.com/Perl/perl5/issues/16795>.

- Thoroughly test paragraph mode, using a new test file, *t/io/paragraph_mode.t*. [GH #16787] <https://github.com/Perl/perl5/issues/16787>.

- Some tests in *t/io/eintr.t* caused the process to hang on pre−16 Darwin. These tests are skipped for those version of Darwin.

## Platform Support

### Platform-Specific Notes

HP-UX 11.11

An obscure problem in `pack()` when compiling with HP C−ANSI-C has been fixed by disabling optimizations in *pp_pack.c*.

Mac OS X

Perl's build and testing process on Mac OS X for `-Duseshrplib` builds is now compatible with Mac OS X System Integrity Protection (SIP).

SIP prevents binaries in */bin* (and a few other places) being passed the `DYLD_LIBRARY_PATH` environment variable. For our purposes this prevents `DYLD_LIBRARY_PATH` from being passed to the shell, which prevents that variable being passed to the testing or build process, so running `perl` couldn't find *libperl.dylib*.

To work around that, the initial build of the *perl* executable expects to find *libperl.dylib* in the build directory, and the library path is then adjusted during installation to point to the installed library.

[GH #15057] <https://github.com/Perl/perl5/issues/15057>.

Minix3

Some support for Minix3 has been re-added.

Cygwin

Cygwin doesn't make `cuserid` visible.

Win32 Mingw

C99 math functions are now available.

Windows

- The `USE_CPLUSPLUS` build option which has long been available in *win32/Makefile* (for **nmake**) and *win32/makefile.mk* (for **dmake**) is now also available in *win32/GNUmakefile* (for **gmake**).

- The **nmake** makefile no longer defaults to Visual C++ 6.0 (a very old version which is unlikely to be widely used today). As a result, it is now a requirement to specify the `CCTYPE` since there is no obvious choice of which modern version to default to instead. Failure to specify `CCTYPE` will result in an error being output and the build will stop.

  (The **dmake** and **gmake** makefiles will automatically detect which compiler is being used, so do not require `CCTYPE` to be set. This feature has not yet been added to the **nmake** makefile.)

- `sleep()` with warnings enabled for a `USE_IMP_SYS` build no longer warns about the sleep timeout being too large. [GH #16631] <https://github.com/Perl/perl5/issues/16631>.

- Support for compiling perl on Windows using Microsoft Visual Studio 2019 (containing Visual C++ 14.2) has been added.

- **socket()** now sets `$!` if the protocol, address family and socket type combination is not found. [GH #16849] <https://github.com/Perl/perl5/issues/16849>.
- The Windows Server 2003 SP1 Platform SDK build, with its early x64 compiler and tools, was accidentally broken in Perl 5.27.9. This has now been fixed.

## Internal Changes

- The sizing pass has been eliminated from the regular expression compiler. An extra pass may instead be needed in some cases to count the number of parenthetical capture groups.
- A new function "`my_strtod`" in perlapi or its synonym, **Strtod()**, is now available with the same signature as the libc **strtod()**. It provides **strotod()** equivalent behavior on all platforms, using the best available precision, depending on platform capabilities and *Configure* options, while handling locale-related issues, such as if the radix character should be a dot or comma.
- Added `newSVsv_nomg()` to copy a SV without processing get magic on the source. [GH #16461] <https://github.com/Perl/perl5/issues/16461>.
- It is now forbidden to malloc more than `PTRDIFF_T_MAX` bytes. Much code (including C optimizers) assumes that all data structures will not be larger than this, so this catches such attempts before overflow happens.
- Two new regnodes have been introduced `EXACT_ONLY8`, and `EXACTFU_ONLY8`. They're equivalent to `EXACT` and `EXACTFU`, except that they contain a code point which requires UTF-8 to represent/match. Hence, if the target string isn't UTF-8, we know it can't possibly match, without needing to try.
- `print_bytes_for_locale()` is now defined if `DEBUGGING`, Prior, it didn't get defined unless `LC_COLLATE` was defined on the platform.

## Selected Bug Fixes

- Compilation under `-DPERL_MEM_LOG` and `-DNO_LOCALE` have been fixed.
- Perl 5.28 introduced an `index()` optimization when comparing to -1 (or indirectly, e.g. >= 0). When this optimization was triggered inside a `when` clause it caused a warning (''Argument `%s` isn't numeric in smart match''). This has now been fixed. [GH #16626] <https://github.com/Perl/perl5/issues/16626>
- The new in-place editing code no longer leaks directory handles. [GH #16602] <https://github.com/Perl/perl5/issues/16602>.
- Warnings produced from constant folding operations on overloaded values no longer produce spurious ''Use of uninitialized value'' warnings. [GH #16349] <https://github.com/Perl/perl5/issues/16349>.
- Fix for ''mutator not seen in (lex = ...) .= ...'' [GH #16655] <https://github.com/Perl/perl5/issues/16655>.
- `pack "u", "invalid uuencoding"` now properly NUL terminates the zero-length SV produced. [GH #16343] <https://github.com/Perl/perl5/issues/16343>.
- Improve the debugging output for **calloc()** calls with `-Dm`. [GH #16653] <https://github.com/Perl/perl5/issues/16653>.
- Regexp script runs were failing to permit ASCII digits in some cases. [GH #16704] <https://github.com/Perl/perl5/issues/16704>.
- On Unix-like systems supporting a platform-specific technique for determining `$^X`, Perl failed to fall back to the generic technique when the platform-specific one fails (for example, a Linux system with /proc not mounted). This was a regression in Perl 5.28.0. [GH #16715] <https://github.com/Perl/perl5/issues/16715>.
- SDBM_File is now more robust with corrupt database files. The improvements do not make SDBM files suitable as an interchange format. [GH #16164] <https://github.com/Perl/perl5/issues/16164>.
- `binmode($fh);` or `binmode($fh, ':raw');` now properly removes the `:utf8` flag from the default `:crlf` I/O layer on Win32. [GH #16730] <https://github.com/Perl/perl5/issues/16730>.

- The experimental reference aliasing feature was misinterpreting array and hash slice assignment as being localised, e.g.

  ```
  \(@a[3,5,7]) = \(....);
  ```

  was being interpreted as:

  ```
  local \(@a[3,5,7]) = \(....);
  ```

  [GH #16701] <https://github.com/Perl/perl5/issues/16701>.

- `sort SUBNAME` within an `eval EXPR` when `EXPR` was UTF-8 upgraded could panic if the `SUBNAME` was non-ASCII. [GH #16979] <https://github.com/Perl/perl5/issues/16979>.

- Correctly handle **realloc()** modifying `errno` on success so that the modification isn't visible to the perl user, since **realloc()** is called implicitly by the interpreter. This modification is permitted by the C standard, but has only been observed on FreeBSD 13.0-CURRENT. [GH #16907] <https://github.com/Perl/perl5/issues/16907>.

- Perl now exposes POSIX `getcwd` as `Internals::getcwd()` if available. This is intended for use by `Cwd.pm` during bootstrapping and may be removed or changed without notice. This fixes some bootstrapping issues while building perl in a directory where some ancestor directory isn't readable. [GH #16903] <https://github.com/Perl/perl5/issues/16903>.

- `pack()` no longer can return malformed UTF-8. It croaks if it would otherwise return a UTF-8 string that contains malformed UTF-8. This protects against potential security threats. [GH #16035] <https://github.com/Perl/perl5/issues/16035>.

- See ''Any set of digits in the Common script are legal in a script run of another script''.

- Regular expression matching no longer leaves stale UTF-8 length magic when updating `$^R`. This could result in `length($^R)` returning an incorrect value.

- Reduce recursion on ops [GH #11866] <https://github.com/Perl/perl5/issues/11866>.

  This can prevent stack overflow when processing extremely deep op trees.

- Avoid leak in multiconcat with overloading. [GH #16823] <https://github.com/Perl/perl5/issues/16823>.

- The handling of user-defined `\p{}` properties (see ''User-Defined Character Properties'' in perlunicode) has been rewritten to be in C (instead of Perl). This speeds things up, but in the process several inconsistencies and bug fixes are made.

  1. A few error messages have minor wording changes. This is essentially because the new way is integrated into the regex error handling mechanism that marks the position in the input at which the error occurred. That was not possible previously. The messages now also contain additional back-trace-like information in case the error occurs deep in nested calls.

  2. A user-defined property is implemented as a perl subroutine with certain highly constrained naming conventions. It was documented previously that the sub would be in the current package if the package was unspecified. This turned out not to be true in all cases, but now it is.

  3. All recursive calls are treated as infinite recursion. Previously they would cause the interpreter to panic. Now, they cause the regex pattern to fail to compile.

  4. Similarly, any other error likely would lead to a panic; now to just the pattern failing to compile.

  5. The old mechanism did not detect illegal ranges in the definition of the property. Now, the range max must not be smaller than the range min. Otherwise, the pattern fails to compile.

  6. The intention was to have each sub called only once during the lifetime of the program, so that a property's definition is immutable. This was relaxed so that it could be called once for all /i compilations, and potentially a second time for non-/i (the sub is passed a parameter indicating which). However, in practice there were instances when this was broken, and multiple calls were possible. Those have been fixed. Now (besides the /i,non-/i cases) the only way a sub can be called multiple times is if some component of it has not been defined yet. For example, suppose we have sub **IsA()** whose definition is known at compile time, and

it in turn calls **isB()** whose definition is not yet known. **isA()** will be called each time a pattern it appears in is compiled. If **isA()** also calls **isC()** and that definition is known, **isC()** will be called just once.

7. There were some races and very long hangs should one thread be compiling the same property as another simultaneously. These have now been fixed.

- Fixed a failure to match properly.

  An EXACTFish regnode has a finite length it can hold for the string being matched. If that length is exceeded, a second node is used for the next segment of the string, for as many regnodes as are needed. Care has to be taken where to break the string, in order to deal multi-character folds in Unicode correctly. If we want to break a string at a place which could potentially be in the middle of a multi-character fold, we back off one (or more) characters, leaving a shorter EXACTFish regnode. This backing off mechanism contained an off-by-one error. [GH #16806] <https://github.com/Perl/perl5/issues/16806>.

- A bare `eof` call with no previous file handle now returns true. [GH #16786] <https://github.com/Perl/perl5/issues/16786>

- Failing to compile a format now aborts compilation. Like other errors in sub-parses this could leave the parser in a strange state, possibly crashing perl if compilation continued. [GH #16169] <https://github.com/Perl/perl5/issues/16169>

- If an in-place edit is still in progress during global destruction and the process exit code (as stored in `$?`) is zero, perl will now treat the in-place edit as successful, replacing the input file with any output produced.

  This allows code like:

  ```
  perl -i -ne 'print "Foo"; last'
  ```

  to replace the input file, while code like:

  ```
  perl -i -ne 'print "Foo"; die'
  ```

  will not. Partly resolves [GH #16748] <https://github.com/Perl/perl5/issues/16748>.

- A regression in 5.28 caused the following code to fail

  ```
  close(STDIN); open(CHILD, "|wc -l")'
  ```

  because the child's stdin would be closed on exec. This has now been fixed.

- Fixed an issue where compiling a regexp containing both compile-time and run-time code blocks could lead to trying to compile something which is invalid syntax.

- Fixed build failures with `-DNO_LOCALE_NUMERIC` and `-DNO_LOCALE_COLLATE`. [GH #16771] <https://github.com/Perl/perl5/issues/16771>.

- Prevent the tests in *ext/B/t/strict.t* from being skipped. [GH #16783] <https://github.com/Perl/perl5/issues/16783>.

- `/di` nodes ending or beginning in *s* are now `EXACTF`. We do not want two `EXACTFU` to be joined together during optimization, and to form a `ss`, `sS`, `Ss` or `SS` sequence; they are the only multi-character sequences which may match differently under `/ui` and `/di`.

## Acknowledgements

Perl 5.30.0 represents approximately 11 months of development since Perl 5.28.0 and contains approximately 620,000 lines of changes across 1,300 files from 58 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 510,000 lines of changes to 750 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.30.0:

Aaron Crane, Abigail, Alberto Simões, Alexandr Savca, Andreas König, Andy Dougherty, Aristotle Pagaltzis, Brian Greenfield, Chad Granum, Chris 'BinGOs' Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, Dan Book, Dan Dedrick, Daniel Dragan, Dan Kogai, David Cantrell, David Mitchell, Dominic Hargreaves, E. Choroba, Ed J, Eugen Konkov, François Perrad, Graham Knop, Hauke D,

H.Merijn Brand, Hugo van der Sanden, Jakub Wilk, James Clarke, James E Keenan, Jerry D. Hedden, Jim Cromie, John SJ Anderson, Karen Etheridge, Karl Williamson, Leon Timmermans, Matthias Bethke, Nicholas Clark, Nicolas R., Niko Tyni, Pali, Petr PísaX, Phil Pearl (Lobbes), Richard Leach, Ryan Voots, Sawyer X, Shlomi Fish, Sisyphus, Slaven Rezic, Steve Hay, Sullivan Beck, Tina Müller, Tomasz Konojacki, Tom Wyant, Tony Cook, Unicode Consortium, Yves Orton, Zak B. Elep.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of most of the (very much appreciated) contributors who reported issues to the Perl bug tracker. Noteworthy in this release were the large number of bug fixes made possible by Sergey Aleynikov's high quality perlbug reports for issues he discovered by fuzzing with AFL.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5301delta − what is new for perl v5.30.1

## DESCRIPTION

This document describes differences between the 5.30.0 release and the 5.30.1 release.

If you are upgrading from an earlier release such as 5.28.0, first read perl5300delta, which describes differences between 5.28.0 and 5.30.0.

## Incompatible Changes

There are no changes intentionally incompatible with 5.30.1. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20190522 to 5.20191110.

## Documentation

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, send email to perlbug@perl.org <mailto:perlbug@perl.org>.

Additionally, documentation has been updated to reference GitHub as the new canonical repository and to describe the new GitHub pull request workflow.

## Configuration and Compilation

- The `ECHO` macro is now defined. This is used in a `dtrace` rule that was originally changed for FreeBSD, and the FreeBSD make apparently predefines it. The Solaris make does not predefine `ECHO` which broke this rule on Solaris. [perl #17057] <https://github.com/perl/perl5/issues/17057>

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

## Platform Support

### Platform-Specific Notes

Win32

The locale tests could crash on Win32 due to a Windows bug, and separately due to the CRT throwing an exception if the locale name wasn’t validly encoded in the current code page.

For the second we now decode the locale name ourselves, and always decode it as UTF−8.

[perl #16922] <https://github.com/perl/perl5/issues/16922>

## Selected Bug Fixes

- Setting `$)` now properly sets supplementary group ids, if you have the necessary privileges. [perl #17031] <https://github.com/perl/perl5/issues/17031>
- `readline @foo` now evaluates `@foo` in scalar context. Previously, it would be evaluated in list context, and since **readline()** pops only one argument from the stack, the stack could underflow, or be left with unexpected values on it. [perl #16929] <https://github.com/perl/perl5/issues/16929>
- **sv_gets()** now recovers better if the target SV is modified by a signal handler. [perl #16960] <https://github.com/perl/perl5/issues/16960>
- Matching a non−`SVf_UTF8` string against a regular expression containing Unicode literals could leak an SV on each match attempt. [perl #17140] <https://github.com/perl/perl5/issues/17140>
- `sprintf("%.*a", -10000, $x)` would cause a buffer overflow due to mishandling of the negative precision value. [perl #16942] <https://github.com/perl/perl5/issues/16942>
- `scalar()` on a reference could cause an erroneous assertion failure during compilation. [perl #16969] <https://github.com/perl/perl5/issues/16969>

## Acknowledgements

Perl 5.30.1 represents approximately 6 months of development since Perl 5.30.0 and contains approximately 4,700 lines of changes across 67 files from 14 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 910 lines of

changes to 20 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.30.1:

Chris 'BinGOs' Williams, Dan Book, David Mitchell, Hugo van der Sanden, James E Keenan, Karen Etheridge, Karl Williamson, Manuel Mausz, Max Maischein, Nicolas R., Sawyer X, Steve Hay, Tom Hukins, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications which make it inappropriate to send to a publicly archived mailing list, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5302delta − what is new for perl v5.30.2

## DESCRIPTION

This document describes differences between the 5.30.1 release and the 5.30.2 release.

If you are upgrading from an earlier release such as 5.30.0, first read perl5301delta, which describes differences between 5.30.0 and 5.30.1.

## Incompatible Changes

There are no changes intentionally incompatible with 5.30.0. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Compress::Raw::Bzip2 has been upgraded from version 2.084 to 2.089.
- Module::CoreList has been upgraded from version 5.20191110 to 5.20200314.

## Documentation

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, send email to <https://github.com/Perl/perl5/issues>.

## Configuration and Compilation

- GCC 10 is now supported by *Configure*.

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

## Platform Support

### Platform-Specific Notes

Windows

The MYMALLOC (PERL_MALLOC) build on Windows has been fixed.

## Selected Bug Fixes

- **printf()** or **sprintf()** with the `%n` format no longer cause a panic on debugging builds, or report an incorrectly cached length value when producing `SVfUTF8` flagged strings.

  [GH #17221 <https://github.com/Perl/perl5/issues/17221>]

- A memory leak in regular expression patterns has been fixed.

  [GH #17218 <https://github.com/Perl/perl5/issues/17218>]

- A read beyond buffer in grok_infnan has been fixed.

  [GH #17370 <https://github.com/Perl/perl5/issues/17370>]

- An assertion failure in the regular expression engine has been fixed.

  [GH #17372 <https://github.com/Perl/perl5/issues/17372>]

- `(?{...})` eval groups in regular expressions no longer unintentionally trigger ''EVAL without pos change exceeded limit in regex''.

  [GH #17490 <https://github.com/Perl/perl5/issues/17490>]

## Acknowledgements

Perl 5.30.2 represents approximately 4 months of development since Perl 5.30.1 and contains approximately 2,100 lines of changes across 110 files from 15 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 920 lines of changes to 30 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.30.2:

Chris 'BinGOs' Williams, Dan Book, David Mitchell, Hugo van der Sanden, Karen Etheridge, Karl Williamson, Matthew Horsfall, Nicolas R., Petr PísaX, Renee Baecker, Sawyer X, Steve Hay, Tomasz Konojacki, Tony Cook, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control

history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://rt.perl.org/>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see "SECURITY VULNERABILITY CONTACT INFORMATION" in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5303delta − what is new for perl v5.30.3

## DESCRIPTION

This document describes differences between the 5.30.2 release and the 5.30.3 release.

If you are upgrading from an earlier release such as 5.30.1, first read perl5302delta, which describes differences between 5.30.1 and 5.30.2.

## Security

### [CVE−2020−10543] Buffer overflow caused by a crafted regular expression

A signed `size_t` integer overflow in the storage space calculations for nested regular expression quantifiers could cause a heap buffer overflow in Perl's regular expression compiler that overwrites memory allocated after the regular expression storage space with attacker supplied data.

The target system needs a sufficient amount of memory to allocate partial expansions of the nested quantifiers prior to the overflow occurring. This requirement is unlikely to be met on 64−bit systems.

Discovered by: ManhND of The Tarantula Team, VinCSS (a member of Vingroup).

### [CVE−2020−10878] Integer overflow via malformed bytecode produced by a crafted regular expression

Integer overflows in the calculation of offsets between instructions for the regular expression engine could cause corruption of the intermediate language state of a compiled regular expression. An attacker could abuse this behaviour to insert instructions into the compiled form of a Perl regular expression.

Discovered by: Hugo van der Sanden and Slaven Rezic.

### [CVE−2020−12723] Buffer overflow caused by a crafted regular expression

Recursive calls to `S_study_chunk()` by Perl's regular expression compiler to optimize the intermediate language representation of a regular expression could cause corruption of the intermediate language state of a compiled regular expression.

Discovered by: Sergey Aleynikov.

### Additional Note

An application written in Perl would only be vulnerable to any of the above flaws if it evaluates regular expressions supplied by the attacker. Evaluating regular expressions in this fashion is known to be dangerous since the regular expression engine does not protect against denial of service attacks in this usage scenario.

## Incompatible Changes

There are no changes intentionally incompatible with Perl 5.30.2. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Module::CoreList has been upgraded from version 5.20200314 to 5.20200601_30.

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

## Acknowledgements

Perl 5.30.3 represents approximately 3 months of development since Perl 5.30.2 and contains approximately 1,100 lines of changes across 42 files from 7 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 350 lines of changes to 8 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.30.3:

Chris 'BinGOs' Williams, Hugo van der Sanden, John Lightsey, Karl Williamson, Nicolas R., Sawyer X, Steve Hay.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <https://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see "SECURITY VULNERABILITY CONTACT INFORMATION" in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5320delta − what is new for perl v5.32.0

## DESCRIPTION

This document describes differences between the 5.30.0 release and the 5.32.0 release.

If you are upgrading from an earlier release such as 5.28.0, first read perl5300delta, which describes differences between 5.28.0 and 5.30.0.

## Core Enhancements

### The isa Operator

A new experimental infix operator called `isa` tests whether a given object is an instance of a given class or a class derived from it:

```
if( $obj isa Package::Name ) { ... }
```

For more detail see ‘‘Class Instance Operator’’ in perlop.

### Unicode 13.0 is supported

See <https://www.unicode.org/versions/Unicode13.0.0/> for details.

### Chained comparisons capability

Some comparison operators, as their associativity, *chain* with some operators of the same precedence (but never with operators of different precedence).

```
if ( $x < $y <= $z ) {...}
```

behaves exactly like:

```
if ( $x < $y && $y <= $z ) {...}
```

(assuming that `"$y"` is as simple a scalar as it looks.)

You can read more about this in perlop under ‘‘Operator Precedence and Associativity’’ in perlop.

### New Unicode properties `Identifier_Status` and `Identifier_Type` supported

Unicode has revised its regular expression requirements: <https://www.unicode.org/reports/tr18/tr18−21.html>. As part of that they are wanting more properties to be exposed, ones that aren’t part of the strict UCD (Unicode character database). These two are used for examining inputs for security purposes. Details on their usage is at <https://www.unicode.org/reports/tr39/>.

### It is now possible to write `qr/\p{Name=...}/`, or `qr!\p{na=/(SMILING|GRINNING) FACE/}!`

The Unicode Name property is now accessible in regular expression patterns, as an alternative to `\N{...}`. A comparison of the two methods is given in ‘‘Comparison of \N{...} and \p{name=...}’’ in perlunicode.

The second example above shows that wildcard subpatterns are also usable in this property. See ‘‘Wildcards in Property Values’’ in perlunicode.

### Improvement of `POSIX::mblen()`, `mbtowc`, and `wctomb`

The `POSIX::mblen()`, `mbtowc`, and `wctomb` functions now work on shift state locales and are thread-safe on C99 and above compilers when executed on a platform that has locale thread-safety; the length parameters are now optional.

These functions are always executed under the current C language locale. (See perllocale.) Most locales are stateless, but a few, notably the very rarely encountered ISO 2022, maintain a state between calls to these functions. Previously the state was cleared on every call, but now the state is not reset unless the appropriate parameter is `undef`.

On threaded perls, the C99 functions **mbrlen** (3), **mbrtowc** (3), and **wcrtomb** (3), when available, are substituted for the plain functions. This makes these functions thread-safe when executing on a locale thread-safe platform.

The string length parameters in `mblen` and `mbtowc` are now optional; useful only if you wish to restrict the length parsed in the source string to less than the actual length.

### Alpha assertions are no longer experimental

See ‘‘(*pla:pattern)’’ in perlre, ‘‘(*plb:pattern)’’ in perlre, ‘‘(*nla:pattern)’’ in perlre>, and ‘‘(*nlb:pattern)’’ in perlre. Use of these no longer generates a warning; existing code that disables the

warning category `experimental::alpha_assertions` will continue to work without any changes needed. Enabling the category has no effect.

### Script runs are no longer experimental

See ''Script Runs'' in perlre. Use of these no longer generates a warning; existing code that disables the warning category `experimental::script_run` will continue to work without any changes needed. Enabling the category has no effect.

### Feature checks are now faster

Previously feature checks in the parser required a hash lookup when features were set outside of a feature bundle, this has been optimized to a bit mask check. [GH #17229 <https://github.com/Perl/perl5/issues/17229>]

### Perl is now developed on GitHub

Perl is now developed on GitHub. You can find us at <https://github.com/Perl/perl5>.

Non-security bugs should now be reported via GitHub. Security issues should continue to be reported as documented in perlsec.

### Compiled patterns can now be dumped before optimization

This is primarily useful for tracking down bugs in the regular expression compiler. This dump happens on `-DDEBUGGING` perls, if you specify `-Drv` on the command line; or on any perl if the pattern is compiled within the scope of `use re qw(Debug DUMP_PRE_OPTIMIZE)` or `use re qw(Debug COMPILE EXTRA)`. (All but the second case display other information as well.)

## Security

### [CVE-2020-10543] Buffer overflow caused by a crafted regular expression

A signed `size_t` integer overflow in the storage space calculations for nested regular expression quantifiers could cause a heap buffer overflow in Perl's regular expression compiler that overwrites memory allocated after the regular expression storage space with attacker supplied data.

The target system needs a sufficient amount of memory to allocate partial expansions of the nested quantifiers prior to the overflow occurring. This requirement is unlikely to be met on 64-bit systems.

Discovered by: ManhND of The Tarantula Team, VinCSS (a member of Vingroup).

### [CVE-2020-10878] Integer overflow via malformed bytecode produced by a crafted regular expression

Integer overflows in the calculation of offsets between instructions for the regular expression engine could cause corruption of the intermediate language state of a compiled regular expression. An attacker could abuse this behaviour to insert instructions into the compiled form of a Perl regular expression.

Discovered by: Hugo van der Sanden and Slaven Rezic.

### [CVE-2020-12723] Buffer overflow caused by a crafted regular expression

Recursive calls to `S_study_chunk()` by Perl's regular expression compiler to optimize the intermediate language representation of a regular expression could cause corruption of the intermediate language state of a compiled regular expression.

Discovered by: Sergey Aleynikov.

### Additional Note

An application written in Perl would only be vulnerable to any of the above flaws if it evaluates regular expressions supplied by the attacker. Evaluating regular expressions in this fashion is known to be dangerous since the regular expression engine does not protect against denial of service attacks in this usage scenario.

## Incompatible Changes

### Certain pattern matching features are now prohibited in compiling Unicode property value wildcard subpatterns

These few features are either inappropriate or interfere with the algorithm used to accomplish this task. The complete list is in ''Wildcards in Property Values'' in perlunicode.

### Unused functions `POSIX::mbstowcs` and `POSIX::wcstombs` are removed

These functions could never have worked due to a defective interface specification. There is clearly no demand for them, given that no one has ever complained in the many years the functions were claimed

to be available, hence so-called ‘‘support’’ for them is now dropped.

**A bug fix for `(?[...])` may have caused some patterns to no longer compile**

See ‘‘Selected Bug Fixes’’. The heuristics previously used may have let some constructs compile (perhaps not with the programmer’s intended effect) that should have been errors. None are known, but it is possible that some erroneous constructs no longer compile.

**`\p{`*`user-defined`*`}` properties now always override official Unicode ones**

Previously, if and only if a user-defined property was declared prior to the compilation of the regular expression pattern that contains it, its definition was used instead of any official Unicode property with the same name. Now, it always overrides the official property. This change could break existing code that relied (likely unwittingly) on the previous behavior. Without this fix, if Unicode released a new version with a new property that happens to have the same name as the one you had long been using, your program would break when you upgraded to a perl that used that new Unicode version. See ‘‘User-Defined Character Properties’’ in perlunicode. [GH #17205 <https://github.com/Perl/perl5/issues/17205>]

**Modifiable variables are no longer permitted in constants**

Code like:

```
my $var;
$sub = sub () { $var };
```

where `$var` is referenced elsewhere in some sort of modifiable context now produces an exception when the sub is defined.

This error can be avoided by adding a return to the sub definition:

```
$sub = sub () { return $var };
```

This has been deprecated since Perl 5.22. [GH #17020] <https://github.com/Perl/perl5/issues/17020>

**Use of `vec` on strings with code points above 0xFF is forbidden**

Such strings are represented internally in UTF-8, and `vec` is a bit-oriented operation that will likely give unexpected results on those strings. This was deprecated in perl 5.28.0.

**Use of code points over 0xFF in string bitwise operators**

Some uses of these were already illegal after a previous deprecation cycle. The remaining uses are now prohibited, having been deprecated in perl 5.28.0. See perldeprecation.

**`Sys::Hostname::hostname()` does not accept arguments**

This usage was deprecated in perl 5.28.0 and is now fatal.

**Plain “0” string now treated as a number for range operator**

Previously a range `"0" .. "-1"` would produce a range of numeric strings from ‘‘0’’ through ‘‘99’’; this now produces an empty list, just as `0 .. -1` does. This also means that `"0" .. "9"` now produces a list of integers, where previously it would produce a list of strings.

This was due to a special case that treated strings starting with ‘‘0’’ as strings so ranges like `"00" .. "03"` produced `"00", "01", "02", "03"`, but didn’t specially handle the string `"0"`. [GH #16770] <https://github.com/Perl/perl5/issues/16770>

**`\K` now disallowed in look-ahead and look-behind assertions**

This was disallowed because it causes unexpected behaviour, and no-one could define what the desired behaviour should be. [GH #14638] <https://github.com/Perl/perl5/issues/14638>

## Performance Enhancements

- `my_strnlen` has been sped up for systems that don’t have their own `strnlen` implementation.
- `grok_bin_oct_hex` (and so, `grok_bin`, `grok_oct`, and `grok_hex`) have been sped up.
- `grok_number_flags` has been sped up.
- `sort` is now noticeably faster in cases such as `sort {$a <=> $b}` or `sort {$b <=> $a}`. [GH #17608 <https://github.com/Perl/perl5/pull/17608>]

## Modules and Pragmata

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.32 to 2.36.
- autodie has been upgraded from version 2.29 to 2.32.
- B has been upgraded from version 1.76 to 1.80.
- B::Deparse has been upgraded from version 1.49 to 1.54.
- Benchmark has been upgraded from version 1.22 to 1.23.
- charnames has been upgraded from version 1.45 to 1.48.
- Class::Struct has been upgraded from version 0.65 to 0.66.
- Compress::Raw::Bzip2 has been upgraded from version 2.084 to 2.093.
- Compress::Raw::Zlib has been upgraded from version 2.084 to 2.093.
- CPAN has been upgraded from version 2.22 to 2.27.
- DB_File has been upgraded from version 1.843 to 1.853.
- Devel::PPPort has been upgraded from version 3.52 to 3.57.

  The test files generated on Win32 are now identical to when they are generated on POSIX-like systems.
- diagnostics has been upgraded from version 1.36 to 1.37.
- Digest::MD5 has been upgraded from version 2.55 to 2.55_01.
- Dumpvalue has been upgraded from version 1.18 to 1.21.

  Previously, when dumping elements of an array and encountering an undefined value, the string printed would have been `empty array`. This has been changed to what was apparently originally intended: `empty slot`.
- DynaLoader has been upgraded from version 1.45 to 1.47.
- Encode has been upgraded from version 3.01 to 3.06.
- encoding has been upgraded from version 2.22 to 3.00.
- English has been upgraded from version 1.10 to 1.11.
- Exporter has been upgraded from version 5.73 to 5.74.
- ExtUtils::CBuilder has been upgraded from version 0.280231 to 0.280234.
- ExtUtils::MakeMaker has been upgraded from version 7.34 to 7.44.
- feature has been upgraded from version 1.54 to 1.58.

  A new `indirect` feature has been added, which is enabled by default but allows turning off indirect object syntax.
- File::Find has been upgraded from version 1.36 to 1.37.

  On Win32, the tests no longer require either a file in the drive root directory, or a writable root directory.
- File::Glob has been upgraded from version 1.32 to 1.33.
- File::stat has been upgraded from version 1.08 to 1.09.
- Filter::Simple has been upgraded from version 0.95 to 0.96.
- Getopt::Long has been upgraded from version 2.5 to 2.51.
- Hash::Util has been upgraded from version 0.22 to 0.23.

  The Synopsis has been updated as the example code stopped working with newer perls. [GH #17399 <https://github.com/Perl/perl5/issues/17399>]
- I18N::Langinfo has been upgraded from version 0.18 to 0.19.
- I18N::LangTags has been upgraded from version 0.43 to 0.44.

  Document the `IGNORE_WIN32_LOCALE` environment variable.

- IO has been upgraded from version 1.40 to 1.43.

  IO::Socket no longer caches a zero protocol value, since this indicates that the implementation will select a protocol. This means that on platforms that don't implement `SO_PROTOCOL` for a given socket type the protocol method may return `undef`.

  The supplied *TO* is now always honoured on calls to the `send()` method. [GH #16891] <https://github.com/Perl/perl5/issues/16891>
- IO-Compress has been upgraded from version 2.084 to 2.093.
- IPC::Cmd has been upgraded from version 1.02 to 1.04.
- IPC::Open3 has been upgraded from version 1.20 to 1.21.
- JSON::PP has been upgraded from version 4.02 to 4.04.
- Math::BigInt has been upgraded from version 1.999816 to 1.999818.
- Math::BigInt::FastCalc has been upgraded from version 0.5008 to 0.5009.
- Module::CoreList has been upgraded from version 5.20190522 to 5.20200620.
- Module::Load::Conditional has been upgraded from version 0.68 to 0.70.
- Module::Metadata has been upgraded from version 1.000036 to 1.000037.
- mro has been upgraded from version 1.22 to 1.23.
- Net::Ping has been upgraded from version 2.71 to 2.72.
- Opcode has been upgraded from version 1.43 to 1.47.
- open has been upgraded from version 1.11 to 1.12.
- overload has been upgraded from version 1.30 to 1.31.
- parent has been upgraded from version 0.237 to 0.238.
- perlfaq has been upgraded from version 5.20190126 to 5.20200523.
- PerlIO has been upgraded from version 1.10 to 1.11.
- PerlIO::encoding has been upgraded from version 0.27 to 0.28.
- PerlIO::via has been upgraded from version 0.17 to 0.18.
- Pod::Html has been upgraded from version 1.24 to 1.25.
- Pod::Simple has been upgraded from version 3.35 to 3.40.
- podlators has been upgraded from version 4.11 to 4.14.
- POSIX has been upgraded from version 1.88 to 1.94.
- re has been upgraded from version 0.37 to 0.40.
- Safe has been upgraded from version 2.40 to 2.41.
- Scalar::Util has been upgraded from version 1.50 to 1.55.
- SelfLoader has been upgraded from version 1.25 to 1.26.
- Socket has been upgraded from version 2.027 to 2.029.
- Storable has been upgraded from version 3.15 to 3.21.

  Use of `note()` from Test::More is now optional in tests. This works around a circular dependency with Test::More when installing on very old perls from CPAN.

  Vstring magic strings over 2GB are now disallowed.

  Regular expressions objects weren't properly counted for object id purposes on retrieve. This would corrupt the resulting structure, or cause a runtime error in some cases. [GH #17037] <https://github.com/Perl/perl5/issues/17037>
- Sys::Hostname has been upgraded from version 1.22 to 1.23.

- Sys::Syslog has been upgraded from version 0.35 to 0.36.
- Term::ANSIColor has been upgraded from version 4.06 to 5.01.
- Test::Simple has been upgraded from version 1.302162 to 1.302175.
- Thread has been upgraded from version 3.04 to 3.05.
- Thread::Queue has been upgraded from version 3.13 to 3.14.
- threads has been upgraded from version 2.22 to 2.25.
- threads::shared has been upgraded from version 1.60 to 1.61.
- Tie::File has been upgraded from version 1.02 to 1.06.
- Tie::Hash::NamedCapture has been upgraded from version 0.10 to 0.13.
- Tie::Scalar has been upgraded from version 1.04 to 1.05.
- Tie::StdHandle has been upgraded from version 4.5 to 4.6.
- Time::HiRes has been upgraded from version 1.9760 to 1.9764.

  Removed obsolete code such as support for pre−5.6 perl and classic MacOS. [GH #17096] <https://github.com/Perl/perl5/issues/17096>
- Time::Piece has been upgraded from version 1.33 to 1.3401.
- Unicode::Normalize has been upgraded from version 1.26 to 1.27.
- Unicode::UCD has been upgraded from version 0.72 to 0.75.
- VMS::Stdio has been upgraded from version 2.44 to 2.45.
- warnings has been upgraded from version 1.44 to 1.47.
- Win32 has been upgraded from version 0.52 to 0.53.
- Win32API::File has been upgraded from version 0.1203 to 0.1203_01.
- XS::APItest has been upgraded from version 1.00 to 1.09.

### Removed Modules and Pragmata

- Pod::Parser has been removed from the core distribution. It still is available for download from CPAN. This resolves [#13194 <https://github.com/Perl/perl5/issues/13194>].

## Documentation

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, open an issue at <https://github.com/Perl/perl5/issues>.

Additionally, the following selected changes have been made:

*perldebguts*

- Simplify a few regnode definitions

  Update `BOUND` and `NBOUND` definitions.
- Add ANYOFHs regnode

  This node is like `ANYOFHb`, but is used when more than one leading byte is the same in all the matched code points.

  `ANYOFHb` is used to avoid having to convert from UTF−8 to code point for something that won't match. It checks that the first byte in the UTF−8 encoded target is the desired one, thus ruling out most of the possible code points.

*perlapi*

- `sv_2pvbyte` updated to mention it will croak if the SV cannot be downgraded.
- `sv_setpvn` updated to mention that the UTF−8 flag will not be changed by this function, and a terminating NUL byte is guaranteed.

- Documentation for `PL_phase` has been added.
- The documentation for `grok_bin`, `grok_oct`, and `grok_hex` has been updated and clarified.

*perldiag*

- Add documentation for experimental 'isa' operator

  (S experimental::isa) This warning is emitted if you use the (`isa`) operator. This operator is currently experimental and its behaviour may change in future releases of Perl.

*perlfunc*

`caller`
: Like `__FILE__` and `__LINE__`, the filename and line number returned here may be altered by the mechanism described at ''Plain Old Comments (Not!)'' in perlsyn.

`__FILE__`
: It can be altered by the mechanism described at ''Plain Old Comments (Not!)'' in perlsyn.

`__LINE__`
: It can be altered by the mechanism described at ''Plain Old Comments (Not!)'' in perlsyn.

`return`
: Now mentions that you cannot return from `do BLOCK`.

`open`
: The `open()` section had been renovated significantly.

*perlguts*

- No longer suggesting using perl's `malloc`. Modern system `malloc` is assumed to be much better than perl's implementation now.
- Documentation about *embed.fnc* flags has been removed. *embed.fnc* now has sufficient comments within it. Anyone changing that file will see those comments first, so entries here are now redundant.
- Updated documentation for `UTF8f`
- Added missing `=for apidoc` lines

*perlhacktips*

- The differences between Perl strings and C strings are now detailed.

*perlintro*

- The documentation for the repetition operator `x` have been clarified. [GH #17335 <https://github.com/Perl/perl5/issues/17335>]

*perlipc*

- The documentation surrounding `open` and handle usage has been modernized to prefer 3−arg open and lexical variables instead of barewords.
- Various updates and fixes including making all examples strict-safe and replacing `−w` with `use warnings`.

*perlop*

- 'isa' operator is experimental

  This is an experimental feature and is available when enabled by `use feature 'isa'`. It emits a warning in the `experimental::isa` category.

*perlpod*

- Details of the various stacks within the perl interpreter are now explained here.
- Advice has been added regarding the usage of `Z<>`.

*perlport*

- Update `timegm` example to use the correct year format *1970* instead of *70*. [GH #16431 <https://github.com/Perl/perl5/issues/16431>]

*perlreref*

- Fix some typos.

*perlvar*

- Now recommends stringifying `$]` and comparing it numerically.

*perlapi, perlintern*

- Documentation has been added for several functions that were lacking it before.

*perlxs*

- Suggest using `libffi` for simple library bindings via CPAN modules like FFI::Platypus or FFI::Raw.

*POSIX*

- `setlocale` warning about threaded builds updated to note it does not apply on Perl 5.28.X and later.
- `Posix::SigSet->new(...)` updated to state it throws an error if any of the supplied signals cannot be added to the set.

Additionally, the following selected changes have been made:

*Updating of links*

- Links to the now defunct <https://search.cpan.org> site now point at the equivalent <https://metacpan.org> URL. [GH #17393 <https://github.com/Perl/perl5/issues/17393>]
- The man page for ExtUtils::XSSymSet is now only installed on VMS, which is the only platform the module is installed on. [GH #17424 <https://github.com/Perl/perl5/issues/17424>]
- URLs have been changed to `https://` and stale links have been updated.

  Where applicable, the URLs in the documentation have been moved from using the `http://` protocol to `https://`. This also affects the location of the bug tracker at <https://rt.perl.org>.
- Some links to OS/2 libraries, Address Sanitizer and other system tools had gone stale. These have been updated with working links.
- Some links to old email addresses on perl5−porters had gone stale. These have been updated with working links.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Expecting interpolated extended charclass in regex; marked by <−− HERE in m/%s/

  This is a replacement for several error messages listed under ‘‘Changes to Existing Diagnostics’’.
- `No digits found for %s literal`

  (F) No hexadecimal digits were found following `0x` or no binary digits were found following `0b`.

*New Warnings*

- Code point 0x%X is not Unicode, and not portable

  This is actually not a new message, but it is now output when the warnings category `portable` is enabled.

  When raised during regular expression pattern compilation, the warning has extra text added at the end marking where precisely in the pattern it occurred.

- Non-hex character ’%c’ terminates \x early. Resolved as ‘‘%s’’

  This replaces a warning that was much less specific, and which gave false information. This new warning parallels the similar already-existing one raised for `\o{}`.

### Changes to Existing Diagnostics

- Character following ‘‘\c’’ must be printable ASCII

  ...now has extra text added at the end, when raised during regular expression pattern compilation, marking where precisely in the pattern it occurred.

- Use ‘‘%s’’ instead of ‘‘%s’’

  ...now has extra text added at the end, when raised during regular expression pattern compilation, marking where precisely in the pattern it occurred.

- Sequence ‘‘\c{’’ invalid

  ...now has extra text added at the end, when raised during regular expression pattern compilation, marking where precisely in the pattern it occurred.

- ‘‘\c%c’’ is more clearly written simply as ‘‘%s’’

  ...now has extra text added at the end, when raised during regular expression pattern compilation, marking where precisely in the pattern it occurred.

- Non-octal character ’%c’ terminates \o early. Resolved as ‘‘%s’’

  ...now includes the phrase ‘‘terminates \o early’’, and has extra text added at the end, when raised during regular expression pattern compilation, marking where precisely in the pattern it occurred. In some instances the text of the resolution has been clarified.

- ’%s’ resolved to ’\o{%s}%d’

  As of Perl 5.32, this message is no longer generated. Instead, ‘‘Non-octal character ’%c’ terminates \o early. Resolved as ’’%s"" in perldiag is used instead.

- Use of code point 0x%s is not allowed; the permissible max is 0x%X

  Some instances of this message previously output the hex digits `A`, `B`, `C`, `D`, `E`, and `F` in lower case. Now they are all consistently upper case.

- The following three diagnostics have been removed, and replaced by `Expecting interpolated extended charclass in regex; marked by <-- HERE in m/%s/` : `Expecting close paren for nested extended charclass in regex; marked by <-- HERE in m/%s/`, `Expecting close paren for wrapper for nested extended charclass in regex; marked by <-- HERE in m/%s/`, and `Expecting '(?flags:(?[...' in regex; marked by <-- HERE in m/%s/`.

- The `Code point 0x%X is not Unicode, and not portable` warning removed the line `Code points above 0xFFFF_FFFF require larger than a 32 bit word.` as code points that large are no longer legal on 32−bit platforms.

- Can’t use global `%s` in `%s`

  This error message has been slightly reformatted from the original `Can't use global %s in "%s"`, and in particular misleading error messages like `Can't use global $_ in "my"` are now rendered as `Can't use global $_ in subroutine signature`.

- Constants from lexical variables potentially modified elsewhere are no longer permitted

  This error message replaces the former `Constants from lexical variables potentially modified elsewhere are deprecated. This will not be allowed in Perl 5.32` to reflect the fact that this previously deprecated usage has now been transformed into an exception. The message’s classification has also been updated from D (deprecated) to F (fatal).

  See also ‘‘Incompatible Changes’’.

- `\N{} here is restricted to one character` is now emitted in the same circumstances where previously `\N{} in inverted character class or as a range end-point is restricted to one character` was.

  This is due to new circumstances having been added in Perl 5.30 that weren't covered by the earlier wording.

## Utility Changes

### perlbug

- The bug tracker homepage URL now points to GitHub.

### streamzip

- This is a new utility, included as part of an IO::Compress::Base upgrade.

  streamzip creates a zip file from stdin. The program will read data from stdin, compress it into a zip container and, by default, write a streamed zip file to stdout.

## Configuration and Compilation

*Configure*

- For clang++, add `#include <stdlib.h>` to Configure's probes for `futimes`, `strtoll`, `strtoul`, `strtoull`, `strtouq`, otherwise the probes would fail to compile.
- Use a compile and run test for `lchown` to satisfy clang++ which should more reliably detect it.
- For C++ compilers, add `#include <stdio.h>` to Configure's probes for `getpgrp` and `setpgrp` as they use printf and C++ compilers may fail compilation instead of just warning.
- Check if the compiler can handle inline attribute.
- Check for character data alignment.
- *Configure* now correctly handles gcc-10. Previously it was interpreting it as gcc-1 and turned on `-fpcc-struct-return`.
- Perl now no longer probes for `d_u32align`, defaulting to `define` on all platforms. This check was error-prone when it was done, which was on 32-bit platforms only. [GH #16680] <https://github.com/Perl/perl5/issues/16680>
- Documentation and hints for building perl on Z/OS (native EBCDIC) have been updated. This is still a work in progress.
- A new probe for `malloc_usable_size` has been added.
- Improvements in *Configure* to detection in C++ and clang++. Work ongoing by Andy Dougherty. [GH #17033] <https://github.com/Perl/perl5/issues/17033>
- *autodoc.pl*

  This tool that regenerates perlintern and perlapi has been overhauled significantly, restoring consistency in flags used in *embed.fnc* and Devel::PPPort and allowing removal of many redundant `=for apidoc` entries in code.
- The `ECHO` macro is now defined. This is used in a `dtrace` rule that was originally changed for FreeBSD, and the FreeBSD make apparently predefines it. The Solaris make does not predefine `ECHO` which broke this rule on Solaris. [GH #17057] <https://github.com/Perl/perl5/issues/17057>
- Bison versions 3.1 through 3.4 are now supported.

## Testing

Tests were added and changed to reflect the other additions and changes in this release. Furthermore, these significant changes were made:

- *t/run/switches.t* no longer uses (and re-uses) the *tmpinplace/* directory under *t/*. This may prevent spurious failures. [GH #17424 <https://github.com/Perl/perl5/issues/17424>]
- Various bugs in `POSIX::mbtowc` were fixed. Potential races with other threads are now avoided, and previously the returned wide character could well be garbage.
- Various bugs in `POSIX::wctomb` were fixed. Potential races with other threads are now avoided, and previously it would segfault if the string parameter was shared or hadn't been pre-allocated with a string of sufficient length to hold the result.

- Certain test output of scalars containing control characters and Unicode has been fixed on EBCDIC.
- *t/charset_tools.pl*: Avoid some work on ASCII platforms.
- *t/re/regexp.t*: Speed up many regex tests on ASCII platform
- *t/re/pat.t*: Skip tests that don't work on EBCDIC.

## Platform Support

### Discontinued Platforms

Windows CE

Support for building perl on Windows CE has now been removed.

### Platform-Specific Notes

Linux

`cc` will be used to populate `plibpth` if `cc` is `clang`. [GH #17043] <https://github.com/Perl/perl5/issues/17043>

NetBSD 8.0

Fix compilation of Perl on NetBSD 8.0 with g++. [GH #17381 <https://github.com/Perl/perl5/issues/17381>]

Windows

- The configuration for `ccflags` and `optimize` are now separate, as with POSIX platforms. [GH #17156 <https://github.com/Perl/perl5/issues/17156>]
- Support for building perl with Visual C++ 6.0 has now been removed.
- The locale tests could crash on Win32 due to a Windows bug, and separately due to the CRT throwing an exception if the locale name wasn't validly encoded in the current code page.

  For the second we now decode the locale name ourselves, and always decode it as UTF-8. [GH #16922] <https://github.com/Perl/perl5/issues/16922>
- *t/op/magic.t* could fail if environment variables starting with `FOO` already existed.
- MYMALLOC (PERL_MALLOC) build has been fixed.

Solaris

- `Configure` will now find recent versions of the Oracle Developer Studio compiler, which are found under `/opt/developerstudio*`.
- `Configure` now uses the detected types for `gethostby*` functions, allowing Perl to once again compile on certain configurations of Solaris.

VMS

- With the release of the patch kit C99 V2.0, VSI has provided support for a number of previously-missing C99 features. On systems with that patch kit installed, Perl's configuration process will now detect the presence of the header `stdint.h` and the following functions: `fpclassify`, `isblank`, `isless`, `llrint`, `llrintl`, `llround`, `llroundl`, `nearbyint`, `round`, `scalbn`, and `scalbnl`.
- `-Duse64bitint` is now the default on VMS.

z/OS

Perl 5.32 has been tested on z/OS 2.4, with the following caveats:

- Only static builds (the default) build reliably
- When using locales, z/OS does not handle the `LC_MESSAGES` category properly, so when compiling perl, you should add the following to your *Configure* options

  ```
  ./Configure <other options> -Accflags=-DNO_LOCALE_MESSAGES
  ```
- z/OS does not support locales with threads, so when compiling a threaded perl, you should add the following to your *Configure* options

  ```
  ./Configure <other Configure options> -Accflags=-DNO_LOCALE
  ```

- Some CPAN modules that are shipped with perl fail at least one of their self-tests. These are: Archive::Tar, Config::Perl::V, CPAN::Meta, CPAN::Meta::YAML, Digest::MD5, Digest::SHA, Encode, ExtUtils::MakeMaker, ExtUtils::Manifest, HTTP::Tiny, IO::Compress, IPC::Cmd, JSON::PP, libnet, MIME::Base64, Module::Metadata, PerlIO::via−QuotedPrint, Pod::Checker, podlators, Pod::Simple, Socket, and Test::Harness.

  The causes of the failures range from the self-test itself is flawed, and the module actually works fine, up to the module doesn't work at all on EBCDIC platforms.

## Internal Changes

- `savepvn`'s len parameter is now a `Size_t` instead of an `I32` since we can handle longer strings than 31 bits.
- The lexer (`Perl_yylex()` in *toke.c*) was previously a single 4100−line function, relying heavily on `goto` and a lot of widely-scoped local variables to do its work. It has now been pulled apart into a few dozen smaller static functions; the largest remaining chunk (`yyl_word_or_keyword()`) is a little over 900 lines, and consists of a single `switch` statement, all of whose `case` groups are independent. This should be much easier to understand and maintain.
- The OS-level signal handlers and type (Sighandler_t) used by the perl core were declared as having three parameters, but the OS was always told to call them with one argument. This has been fixed by declaring them to have one parameter. See the merge commit `v5.31.5−346−g116e19abbf` for full details.
- The code that handles `tr///` has been extensively revised, fixing various bugs, especially when the source and/or replacement strings contain characters whose code points are above 255. Some of the bugs were undocumented, one being that under some circumstances (but not all) with `/s`, the squeezing was done based on the source, rather than the replacement. A documented bug that got fixed was [GH #14777] <https://github.com/Perl/perl5/issues/14777>.
- A new macro for XS writers dealing with UTF−8−encoded Unicode strings has been created "`UTF8_CHK_SKIP`" in perlapi that is safer in the face of malformed UTF−8 input than "`UTF8_SKIP`" in perlapi (but not as safe as "`UTF8_SAFE_SKIP`" in perlapi). It won't read past a NUL character. It has been backported in Devel::PPPort 3.55 and later.
- Added the `PL_curstackinfo->si_cxsubix` field. This records the stack index of the most recently pushed sub/format/eval context. It is set and restored automatically by `cx_pushsub()`, `cx_popsub()` etc., but would need to be manually managed if you do any unusual manipulation of the context stack.
- Various macros dealing with character type classification and changing case where the input is encoded in UTF−8 now require an extra parameter to prevent potential reads beyond the end of the buffer. Use of these has generated a deprecation warning since Perl 5.26. Details are in ''In XS code, use of various macros dealing with UTF−8.'' in perldeprecation
- A new parser function **parse_subsignature()** allows a keyword plugin to parse a subroutine signature while `use feature 'signatures'` is in effect. This allows custom keywords to implement semantics similar to regular `sub` declarations that include signatures. [GH #16261] <https://github.com/Perl/perl5/issues/16261>
- Since on some platforms we need to hold a mutex when temporarily switching locales, new macros (`STORE_LC_NUMERIC_SET_TO_NEEDED_IN`, `WITH_LC_NUMERIC_SET_TO_NEEDED` and `WITH_LC_NUMERIC_SET_TO_NEEDED_IN`) have been added to make it easier to do this safely and efficiently as part of [GH #17034] <https://github.com/Perl/perl5/issues/17034>.
- The memory bookkeeping overhead for allocating an OP structure has been reduced by 8 bytes per OP on 64−bit systems.
- **eval_pv()** no longer stringifies the exception when `[GH #17035]|https://github.com/Perl/perl5/issues/17035]`
- The PERL_DESTRUCT_LEVEL environment variable was formerly only honoured on perl binaries built with DEBUGGING support. It is now checked on all perl builds. Its normal use is to force perl to individually free every block of memory which it has allocated before exiting, which is

useful when using automated leak detection tools such as valgrind.

- The API **eval_sv()** now accepts a `G_RETHROW` flag. If this flag is set and an exception is thrown while compiling or executing the supplied code, it will be rethrown, and **eval_sv()** will not return. [GH #17036] <https://github.com/Perl/perl5/issues/17036>
- As part of the fix for [GH #1537] <https://github.com/Perl/perl5/issues/1537> **perl_parse()** now returns non-zero if **exit** (0) is called in a `BEGIN`, `UNITCHECK` or `CHECK` block.
- Most functions which recursively walked an op tree during compilation have been made non-recursive. This avoids SEGVs from stack overflow when the op tree is deeply nested, such as `$n == 1 ? "one" : $n == 2 ? "two" : ....` (especially in code which is auto-generated).

  This is particularly noticeable where the code is compiled within a separate thread, as threads tend to have small stacks by default.

## Selected Bug Fixes

- Previously ''require'' in perlfunc would only treat the special built-in SV `&PL_sv_undef` as a value in `%INC` as if a previous `require` has failed, treating other undefined SVs as if the previous `require` has succeeded. This could cause unexpected success from `require` e.g., on `local %INC = %INC;`. This has been fixed. [GH #17428 <https://github.com/Perl/perl5/issues/17428>]
- `(?{...})` eval groups in regular expressions no longer unintentionally trigger ''EVAL without pos change exceeded limit in regex'' [GH #17490 <https://github.com/Perl/perl5/issues/17490>].
- `(?[...])` extended bracketed character classes do not wrongly raise an error on some cases where a previously-compiled such class is interpolated into another. The heuristics previously used have been replaced by a reliable method, and hence the diagnostics generated have changed. See ''Diagnostics''.
- The debug display (say by specifying `-Dr` or `use re` (with appropriate options) of compiled Unicode property wildcard subpatterns no longer has extraneous output.
- Fix an assertion failure in the regular expression engine. [GH #17372 <https://github.com/Perl/perl5/issues/17372>]
- Fix coredump in pp_hot.c after `B::UNOP_AUX::aux_list()`. [GH #17301 <https://github.com/Perl/perl5/issues/17301>]
- Loading IO is now threadsafe. [GH #14816 <https://github.com/Perl/perl5/issues/14816>]
- `\p{user-defined}` overrides official Unicode [GH #17025 <https://github.com/Perl/perl5/issues/17025>]

  Prior to this patch, the override was only sometimes in effect.
- Properly handle filled `/il` regnodes and multi-char folds
- Compilation error during make minitest [GH #17293 <https://github.com/Perl/perl5/issues/17293>]
- Move the implementation of `%-`, `%+` into core.
- Read beyond buffer in `grok_inf_nan` [GH #17370 <https://github.com/Perl/perl5/issues/17370>]
- Workaround glibc bug with `LC_MESSAGES` [GH #17081 <https://github.com/Perl/perl5/issues/17081>]
- `printf()` or `sprintf()` with the `%n` format could cause a panic on debugging builds, or report an incorrectly cached length value when producing `SVfUTF8` flagged strings. [GH #17221 <https://github.com/Perl/perl5/issues/17221>]
- The tokenizer has been extensively refactored. [GH #17241 <https://github.com/Perl/perl5/issues/17241>] [GH #17189 <https://github.com/Perl/perl5/issues/17189>]

- `use strict "subs"` is now enforced for bareword constants optimized into a `multiconcat` operator. [GH #17254 <https://github.com/Perl/perl5/issues/17254>]
- A memory leak in regular expression patterns has been fixed. [GH #17218 <https://github.com/Perl/perl5/issues/17218>]
- Perl no longer treats strings starting with ‘‘0x’’ or ‘‘0b’’ as hex or binary numbers respectively when converting a string to a number. This reverts a change in behaviour inadvertently introduced in perl 5.30.0 intended to improve precision when converting a string to a floating point number. [GH #17062] <https://github.com/Perl/perl5/issues/17062>
- Matching a non−`SVf_UTF8` string against a regular expression containing unicode literals could leak a SV on each match attempt. [GH #17140] <https://github.com/Perl/perl5/issues/17140>
- Overloads for octal and binary floating point literals were always passed a string with a `0x` prefix instead of the appropriate `0` or [GH #14791]|https://github.com/Perl/perl5/issues/14791]
- `$@ = 100; die;` now correctly propagates the 100 as an exception instead of ignoring it. [GH #17098] <https://github.com/Perl/perl5/issues/17098>
- [GH #17108]|https://github.com/Perl/perl5/issues/17108]
- Exceptions thrown while `$@` is read-only could result in infinite recursion as perl tried to update `$@`, which throws another exception, resulting in a stack overflow. Perl now replaces `$@` with a copy if it’s not a simple writable SV. [GH #17083] <https://github.com/Perl/perl5/issues/17083>
- Setting `$)` now properly sets supplementary group ids if you have the necessary privileges. [GH #17031] <https://github.com/Perl/perl5/issues/17031>
- **close()** on a pipe now preemptively clears the PerlIO object from the IO SV. This prevents a second attempt to close the already closed PerlIO object if a signal handler calls **die()** or **exit()** while **close()** is waiting for the child process to complete. [GH #13929] <https://github.com/Perl/perl5/issues/13929>
- `sprintf("%.*a", −10000, $x)` would cause a buffer overflow due to mishandling of the negative precision value. [GH #16942] <https://github.com/Perl/perl5/issues/16942>
- **scalar()** on a reference could cause an erroneous assertion failure during compilation. [GH #16969] <https://github.com/Perl/perl5/issues/16969>
- `%{^CAPTURE_ALL}` is now an alias to `%−` as documented, rather than incorrectly an alias for [GH #16105]|https://github.com/Perl/perl5/issues/16105]
- `%{^CAPTURE}` didn’t work if `@{^CAPTURE}` was mentioned first. Similarly for `%{^CAPTURE_ALL}` and `@{^CAPTURE_ALL}`, though [GH #17045]|https://github.com/Perl/perl5/issues/17045]
- Extraordinarily large (over 2GB) floating point format widths could cause an integer overflow in the underlying call to **snprintf()**, resulting in an assertion. Formatted floating point widths are now limited to the range of int, the return value of **snprintf()**. [#16881 <https://github.com/Perl/perl5/issues/16881>]
- Parsing the following constructs within a sub-parse (such as with `"${code here}"` or `s/.../code here/e`) has changed to match how they’re parsed normally:
  - `print $fh ...` no longer produces a syntax error.
  - Code like `s/.../ ${time} /e` now properly produces an ‘‘Ambiguous use of ${time} resolved to `$time` at ...’’ warning when warnings are enabled.
  - `@x {"a"}` (with the space) in a sub-parse now properly produces a ‘‘better written as’’ warning when warnings are enabled.
  - Attributes can now be used in a sub-parse. [GH #16847] <https://github.com/Perl/perl5/issues/16847>
- Incomplete hex and binary literals like `0x` and `0b` are now treated as if the `x` or `b` is part of the next token. [#17010 <https://github.com/Perl/perl5/issues/17010>]

- A spurious `)` in a subparse, such as in `s/.../code here/e` or `"...${code here}"`, no longer confuses the parser.

  Previously a subparse was bracketed with generated `(` and `)` tokens, so a spurious `)` would close the construct without doing the normal subparse clean up, confusing the parser and possible causing an assertion failure.

  Such constructs are now surrounded by artificial tokens that can't be included in the source. [GH #15814] <https://github.com/Perl/perl5/issues/15814>

- Reference assignment of a sub, such as `\&foo = \&bar;`, silently did nothing in the `[GH #16987]|https://github.com/Perl/perl5/issues/16987]`

- **sv_gets()** now recovers better if the target SV is modified by a signal handler. [GH #16960] <https://github.com/Perl/perl5/issues/16960>

- `readline @foo` now evaluates `@foo` in scalar context. Previously it would be evaluated in list context, and since **readline()** pops only one argument from the stack, the stack could underflow, or be left with unexpected values on the stack. [GH #16929] <https://github.com/Perl/perl5/issues/16929>

- Parsing incomplete hex or binary literals was changed in 5.31.1 to treat such a literal as just the 0, leaving the following `x` or `b` to be parsed as part of the next token. This could lead to some silent changes in behaviour, so now incomplete hex or binary literals produce a fatal error. [GH #17010] <https://github.com/Perl/perl5/issues/17010>

- **eval_pv()**'s *croak_on_error* flag will now throw even if the exception is a false overloaded value. [GH #17036] <https://github.com/Perl/perl5/issues/17036>

- `INIT` blocks and the program itself are no longer run if **exit**(0) is called within a `BEGIN`, `UNITCHECK` or `CHECK` block. [GH #1537] <https://github.com/Perl/perl5/issues/1537>

- `open my $fh, ">>+", undef` now opens the temporary file in append mode: writes will seek to the end of file before writing. [GH #17058] <https://github.com/Perl/perl5/issues/17058>

- Fixed a SEGV when searching for the source of an uninitialized value warning on an op whose subtree includes an OP_MULTIDEREF. [GH #17088] <https://github.com/Perl/perl5/issues/17088>

## Obituary

Jeff Goff (JGOFF or DrForr), an integral part of the Perl and Raku communities and a dear friend to all of us, has passed away on March 13th, 2020. DrForr was a prominent member of the communities, attending and speaking at countless events, contributing to numerous projects, and assisting and helping in any way he could.

His passing leaves a hole in our hearts and in our communities and he will be sorely missed.

## Acknowledgements

Perl 5.32.0 represents approximately 13 months of development since Perl 5.30.0 and contains approximately 220,000 lines of changes across 1,800 files from 89 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 140,000 lines of changes to 880 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.32.0:

Aaron Crane, Alberto Simões, Alexandr Savca, Andreas König, Andrew Fresh, Andy Dougherty, Ask Bjørn Hansen, Atsushi Sugawara, Bernhard M. Wiedemann, brian d foy, Bryan Stenson, Chad Granum, Chase Whitener, Chris 'BinGOs' Williams, Craig A. Berry, Dagfinn Ilmari Mannsåker, Dan Book, Daniel Dragan, Dan Kogai, Dave Cross, Dave Rolsky, David Cantrell, David Mitchell, Dominic Hargreaves, E. Choroba, Felipe Gasper, Florian Weimer, Graham Knop, Håkon Hægland, Hauke D, H.Merijn Brand, Hugo van der Sanden, Ichinose Shogo, James E Keenan, Jason McIntosh, Jerome Duval, Johan Vromans, John Lightsey, John Paul Adrian Glaubitz, Kang-min Liu, Karen Etheridge, Karl Williamson, Leon Timmermans, Manuel Mausz, Marc Green, Matthew Horsfall, Matt Turner, Max Maischein, Michael Haardt, Nicholas Clark, Nicolas R., Niko Tyni, Pali, Paul Evans, Paul Johnson, Paul Marquess, Peter Eisentraut, Peter John Acklam, Peter Oliver, Petr PísaX, Renee Baecker, Ricardo Signes, Richard Leach, Russ Allbery, Samuel Smith, Santtu Ojanperä, Sawyer X, Sergey Aleynikov, Sergiy Borodych, Shirakata Kentaro, Shlomi Fish, Sisyphus, Slaven Rezic, Smylers, Stefan

Seifert, Steve Hay, Steve Peters, Svyatoslav, Thibault Duponchelle, Todd Rinaldo, Tomasz Konojacki, Tom Hukins, Tony Cook, Unicode Consortium, VanL, Vickenty Fesunov, Vitali Peil, Yves Orton, Zefram.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5321delta − what is new for perl v5.32.1

## DESCRIPTION

This document describes differences between the 5.32.0 release and the 5.32.1 release.

If you are upgrading from an earlier release such as 5.30.0, first read perl5320delta, which describes differences between 5.30.0 and 5.32.0.

## Incompatible Changes

There are no changes intentionally incompatible with Perl 5.32.0. If any exist, they are bugs, and we request that you submit a report. See ''Reporting Bugs'' below.

## Modules and Pragmata

### Updated Modules and Pragmata

- Data::Dumper has been upgraded from version 2.174 to 2.174_01.

  A number of memory leaks have been fixed.

- DynaLoader has been upgraded from version 1.47 to 1.47_01.

- Module::CoreList has been upgraded from version 5.20200620 to 5.20210123.

- Opcode has been upgraded from version 1.47 to 1.48.

  A warning has been added about evaluating untrusted code with the perl interpreter.

- Safe has been upgraded from version 2.41 to 2.41_01.

  A warning has been added about evaluating untrusted code with the perl interpreter.

## Documentation

### New Documentation

*perlgov*

Documentation of the newly formed rules of governance for Perl.

*perlsecpolicy*

Documentation of how the Perl security team operates and how the team evaluates new security reports.

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, open an issue at <https://github.com/Perl/perl5/issues>.

Additionally, the following selected changes have been made:

*perlop*

- Document range op behaviour change.

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### Changes to Existing Diagnostics

- \K not permitted in lookahead/lookbehind in regex; marked by <−− HERE in m/%s/

  This error was incorrectly produced in some cases involving nested lookarounds. This has been fixed.

  [GH #18123 <https://github.com/Perl/perl5/issues/18123>]

## Configuration and Compilation

- Newer 64−bit versions of the Intel C/C++ compiler are now recognized and have the correct flags set.

- We now trap SIGBUS when *Configure* checks for `va_copy`.

  On several systems the attempt to determine if we need `va_copy` or similar results in a SIGBUS instead of the expected SIGSEGV, which previously caused a core dump.

  [GH #18148 <https://github.com/Perl/perl5/issues/18148>]

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

## Platform Support

### Platform-Specific Notes

MacOS (Darwin)

The hints file for darwin has been updated to handle future macOS versions beyond 10. Perl can now be built on macOS Big Sur.

[GH #17946 <https://github.com/Perl/perl5/issues/17946>, GH #18406 <https://github.com/Perl/perl5/issues/18406>]

Minix

Build errors on Minix have been fixed.

[GH #17908 <https://github.com/Perl/perl5/issues/17908>]

## Selected Bug Fixes

- Some list assignments involving `undef` on the left-hand side were over-optimized and produced incorrect results.

  [GH #16685 <https://github.com/Perl/perl5/issues/16685>, GH #17816 <https://github.com/Perl/perl5/issues/17816>]

- Fixed a bug in which some regexps with recursive subpatterns matched incorrectly.

  [GH #18096 <https://github.com/Perl/perl5/issues/18096>]

- Fixed a deadlock that hung the build when Perl is compiled for debugging memory problems and has PERL_MEM_LOG enabled.

  [GH #18341 <https://github.com/Perl/perl5/issues/18341>]

- Fixed a crash in the use of chained comparison operators when run under ‘‘no warnings ’uninitialized’’’.

  [GH #17917 <https://github.com/Perl/perl5/issues/17917>, GH #18380 <https://github.com/Perl/perl5/issues/18380>]

- Exceptions thrown from destructors during global destruction are no longer swallowed.

  [GH #18063 <https://github.com/Perl/perl5/issues/18063>]

## Acknowledgements

Perl 5.32.1 represents approximately 7 months of development since Perl 5.32.0 and contains approximately 7,000 lines of changes across 80 files from 23 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,300 lines of changes to 23 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.32.1:

Adam Hartley, Andy Dougherty, Dagfinn Ilmari Mannsåker, Dan Book, David Mitchell, Graham Knop, Graham Ollis, Hauke D, H.Merijn Brand, Hugo van der Sanden, John Lightsey, Karen Etheridge, Karl Williamson, Leon Timmermans, Max Maischein, Nicolas R., Ricardo Signes, Richard Leach, Sawyer X, Sevan Janiyan, Steve Hay, Tom Hukins, Tony Cook.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <http://www.perl.org/>, the

Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see "SECURITY VULNERABILITY CONTACT INFORMATION" in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5340delta − what is new for perl v5.34.0

## DESCRIPTION

This document describes differences between the 5.32.0 release and the 5.34.0 release.

If you are upgrading from an earlier release such as 5.30.0, first read perl5320delta, which describes differences between 5.30.0 and 5.32.0.

## Core Enhancements

### Experimental Try/Catch Syntax

An initial experimental attempt at providing `try`/`catch` notation has been added.

```
use feature 'try';

try {
    a_function();
}
catch ($e) {
    warn "An error occurred: $e";
}
```

For more information, see ‘‘Try Catch Exception Handling’’ in perlsyn.

### `qr/{,n}/` is now accepted

An empty lower bound is now accepted for regular expression quantifiers, like `m/x{,3}/` meaning `m/x{0,3}/`

### Blanks freely allowed within but adjacent to curly braces

(in double-quotish contexts and regular expression patterns)

This means you can write things like `\x{ FFFC }` if you like. This applies to all such constructs, namely `\b{}`, `\g{}`, `\k{}`, `\N{}`, `\o{}`, and `\x{}`; as well as the regular expression quantifier *`{m,n}`*. `\p{}` and `\P{}` retain their already-existing, even looser, rules mandated by the Unicode standard (see ‘‘Properties accessible through \p{} and \P{}’’ in perluniprops).

This ability is in effect regardless of the presence of the `/x` regular expression pattern modifier.

Additionally, the comma in a regular expression braced quantifier may have blanks (tabs or spaces) before and/or after the comma, like `qr/a{ 5, 7 }/`.

### New octal syntax `0o`*`ddddd`*

It is now possible to specify octal literals with `0o` prefixes, as in `0o123_456`, parallel to the existing construct to specify hexadecimal literal `0x`*`ddddd`* and binary literal `0b`*`ddddd`*. Also, the builtin `oct()` function now accepts this new syntax.

See ‘‘Scalar value constructors’’ in perldata and ‘‘oct EXPR’’ in perlfunc.

## Performance Enhancements

- Fix a memory leak in RegEx [GH #18604 <https://github.com/Perl/perl5/issues/18604>]

## Modules and Pragmata

### New Modules and Pragmata

- ExtUtils::PL2Bat 0.004 has been added to the Perl core.

  This module is a generalization of the `pl2bat` script. It being a script has led to at least two forks of this code; this module will unify them under one implementation with tests.

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.36 to 2.38.
- autodie has been upgraded from version 2.32 to 2.34.
- B has been upgraded from version 1.80 to 1.82.
- B::Deparse has been upgraded from version 1.54 to 1.56.
- bytes has been upgraded from version 1.07 to 1.08.

- Carp has been upgraded from version 1.50 to 1.52.
- Compress::Raw::Bzip2 has been upgraded from version 2.093 to 2.101.
- Compress::Raw::Zlib has been upgraded from version 2.093 to 2.101.
- Config::Perl::V has been upgraded from version 0.32 to 0.33.
- CPAN has been upgraded from version 2.27 to 2.28.
- Data::Dumper has been upgraded from version 2.174 to 2.179.
- DB has been upgraded from version 1.58 to 1.59.
- DB_File has been upgraded from version 1.853 to 1.855.
- Devel::Peek has been upgraded from version 1.28 to 1.30.
- Devel::PPPort has been upgraded from version 3.57 to 3.62.

  New `PERL_VERSION_*` comparison macros are now available.

  `ppport.h --api-info` no longer includes non-API info unless that is the only match
- Digest has been upgraded from version 1.17_01 to 1.19.
- Digest::MD5 has been upgraded from version 2.55_01 to 2.58.
- DynaLoader has been upgraded from version 1.47 to 1.50.
- Encode has been upgraded from version 3.06 to 3.08.
- Env has been upgraded from version 1.04 to 1.05.
- Errno has been upgraded from version 1.30 to 1.33.
- experimental has been upgraded from version 0.020 to 0.024.
- Exporter has been upgraded from version 5.74 to 5.76.
- ExtUtils::CBuilder has been upgraded from version 0.280234 to 0.280236.
- ExtUtils::Install has been upgraded from version 2.14 to 2.20.
- ExtUtils::MakeMaker has been upgraded from version 7.44 to 7.62.
- ExtUtils::Manifest has been upgraded from version 1.72 to 1.73.
- ExtUtils::Miniperl has been upgraded from version 1.09 to 1.10.
- ExtUtils::ParseXS has been upgraded from version 3.40 to 3.43.
- ExtUtils::Typemaps has been upgraded from version 3.38 to 3.43.
- Fcntl has been upgraded from version 1.13 to 1.14.
- feature has been upgraded from version 1.58 to 1.64.

  Added the default enabled `bareword_filehandles` feature.

  A new multidimensional feature has been added, which is enabled by default but allows turning off multi-dimensional array emulation.
- File::Copy has been upgraded from version 2.34 to 2.35.
- File::Fetch has been upgraded from version 0.56 to 1.00.
- File::Find has been upgraded from version 1.37 to 1.39.
- File::Path has been upgraded from version 2.16 to 2.18.
- File::Spec has been upgraded from version 3.78 to 3.80.
- File::Temp has been upgraded from version 0.2309 to 0.2311.
- Filter::Util::Call has been upgraded from version 1.59 to 1.60.
- FindBin has been upgraded from version 1.51 to 1.52.
- GDBM_File has been upgraded from version 1.18 to 1.19.

  New functions and compatibility for newer versions of GDBM. [GH #18435

<https://github.com/Perl/perl5/pull/18435>]

- Getopt::Long has been upgraded from version 2.51 to 2.52.
- Getopt::Std has been upgraded from version 1.12 to 1.13.
- Hash::Util has been upgraded from version 0.23 to 0.25.
- Hash::Util::FieldHash has been upgraded from version 1.20 to 1.21.
- I18N::LangTags has been upgraded from version 0.44 to 0.45.
- if has been upgraded from version 0.0608 to 0.0609.
- IO has been upgraded from version 1.43 to 1.46.

  IO::Socket now stores error messages in `$IO::Socket::errstr`, in addition to in `$@`.

  The `error` method now reports the error state for both the input and output streams for sockets and character devices. Similarly `clearerr` now clears the error state for both streams.

  A spurious error reported for regular file handles has been fixed in IO::Handle. [GH #18019 <https://github.com/Perl/perl5/issues/18019>]
- IO-Compress has been upgraded from version 2.093 to 2.102.

  bin/zipdetails version 2.02
- IO::Socket::IP has been upgraded from version 0.39 to 0.41.
- IO::Zlib has been upgraded from version 1.10 to 1.11.
- IPC::SysV has been upgraded from version 2.07 to 2.09.
- JSON::PP has been upgraded from version 4.04 to 4.06.
- The libnet distribution has been upgraded from version 3.11 to 3.13.
- locale has been upgraded from version 1.09 to 1.10.
- Math::Complex has been upgraded from version 1.5901 to 1.5902.
- MIME::Base64 has been upgraded from version 3.15 to 3.16.
- Module::CoreList has been upgraded from version 5.20200620 to 5.20210520.
- Module::Load has been upgraded from version 0.34 to 0.36.
- Module::Load::Conditional has been upgraded from version 0.70 to 0.74.
- mro has been upgraded from version 1.23 to 1.25_001.
- Net::Ping has been upgraded from version 2.72 to 2.74.
- NEXT has been upgraded from version 0.67_01 to 0.68.
- ODBM_File has been upgraded from version 1.16 to 1.17.
- Opcode has been upgraded from version 1.47 to 1.50.
- overload has been upgraded from version 1.31 to 1.33.
- perlfaq has been upgraded from version 5.20200523 to 5.20210411.
- PerlIO::encoding has been upgraded from version 0.28 to 0.30.
- PerlIO::mmap has been upgraded from version 0.016 to 0.017.
- PerlIO::scalar has been upgraded from version 0.30 to 0.31.
- PerlIO::via::QuotedPrint has been upgraded from version 0.08 to 0.09.
- Pod::Checker has been upgraded from version 1.73 to 1.74.
- Pod::Html has been upgraded from version 1.25 to 1.27.
- Pod::Simple has been upgraded from version 3.40 to 3.42.
- Pod::Usage has been upgraded from version 1.69 to 2.01.

- POSIX has been upgraded from version 1.94 to 1.97.

  **POSIX::signbit()** behaviour has been improved. [GH #18441 <https://github.com/Perl/perl5/pull/18441>]

  Documentation for `asctime` clarifies that the result is always in English. (Use `strftime` for a localized result.)
- re has been upgraded from version 0.40 to 0.41.

  (See under ‘‘Internal Changes’’ for more information.)
- Safe has been upgraded from version 2.41 to 2.43.
- Socket has been upgraded from version 2.029 to 2.031.
- Storable has been upgraded from version 3.21 to 3.23.
- strict has been upgraded from version 1.11 to 1.12.
- subs has been upgraded from version 1.03 to 1.04.
- Symbol has been upgraded from version 1.08 to 1.09.
- Test::Harness has been upgraded from version 3.42 to 3.43.
- Test::Simple has been upgraded from version 1.302175 to 1.302183.
- Text::Balanced has been upgraded from version 2.03 to 2.04.
- threads has been upgraded from version 2.25 to 2.26.
- threads::shared has been upgraded from version 1.61 to 1.62.
- Tie::RefHash has been upgraded from version 1.39 to 1.40.
- Time::HiRes has been upgraded from version 1.9764 to 1.9767.
- Time::Local has been upgraded from version 1.28 to 1.30.
- Unicode::Collate has been upgraded from version 1.27 to 1.29.
- Unicode::Normalize has been upgraded from version 1.27 to 1.28.
- utf8 has been upgraded from version 1.22 to 1.24.
- version has been upgraded from version 0.9924 to 0.9928.
- warnings has been upgraded from version 1.47 to 1.51.
- Win32 has been upgraded from version 0.53 to 0.57.

  Fix calling convention for `PFNRegGetValueA`.

  Added `Win32::IsSymlinkCreationAllowed()`, `Win32::IsDeveloperModeEnabled()`, and `Win32::GetProcessPrivileges()`.

  Removed old code for versions before Windows 2000.
- XS::APItest has been upgraded from version 1.09 to 1.16.
- XS::Typemap has been upgraded from version 0.17 to 0.18.

## Documentation

### New Documentation

*perldocstyle*

This document is a guide for the authorship and maintenance of the documentation that ships with Perl.

*perlgov*

This document describes the goals, scope, system, and rules for Perl’s new governance model.

Other pod files, most notably perlpolicy, were amended to reflect its adoption.

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, open an issue at <https://github.com/Perl/perl5/issues>.

Additionally, the following selected changes have been made:

- perlapi, perlguts, perlxs, and perlxstut now prefer `SvPVbyte` over `SvPV`.
- References to **Pumpking** have been replaced with a more accurate term or **Steering Council** where appropriate.
- **The Perl Steering Council** is now the fallback contact for security issues.

*perlapi*

- Efforts continue in improving the presentation of this document, and to document more API elements.

*perlcommunity*

- The freenode IRC URL has been updated.

*perldebguts*

- Corrected the description of the scalar `${"_<$filename"}` variables.

*perldiag*

- Now documents additional examples of ‘‘not imported’’ warnings.

*perlfaq*

- The Perl FAQ was updated to CPAN version 5.20201107 with minor improvements.

*perlfunc*

- **my()** and **state()** now explicitly warn the reader that lexical variables should typically not be redeclared within the same scope or statement. [GH #18389 <https://github.com/Perl/perl5/issues/18389>]
- The localtime entry has been improved and now also states that the result of the function is always in English.
- **msgsnd()** documented a length field included in the packed `MSG` parameter to `msgsnd()`, but there was no such field. `MSG` contains only the type and the message content.
- Better explanation of what happens when `sleep` is called with a zero or negative value.
- Simplify the `split()` documentation by removing the `join()`s from the examples [GH #18676 <https://github.com/Perl/perl5/issues/18676>]

*perlgit*

- document how to create a remote-tracking branch for every PR
- document how to get a PR as a local branch

*perlguts*

- perlguts now explains in greater detail the need to consult `SvUTF8` when calling `SvPV` (or variants). A new ‘‘How do I pass a Perl string to a C library?’’ section in the same document discusses when to use which style of macro to read an SV’s string value.
- Corrected `my_rpeep` example in perlguts.
- A section has been added on the formatted printing of special sizes.

*perlop*

- The `<>` and `<<>>` operators are commonly referred to as the diamond and double diamond operators respectively, but that wasn’t mentioned previously in their documentation.
- Document range op behavior change.

*perlpacktut*

- Incorrect variables used in an example have been fixed.

*perlsyn*

- Document that **caller()** does not see try{} blocks

- A new example shows how a lexical `my` variable can be declared during the initialization of a `for` loop.

*perlunifaq*

- Fix description of what Perl does with unencoded strings

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Bareword filehandle ‘‘%s’’ not allowed under ’no feature ‘‘bareword_filehandles’’’

  This accompanies the new bareword_filehandles feature.

- Multidimensional hash lookup is disabled

  This accompanies the new multidimensional feature.

*New Warnings*

- Wide character in setenv key (encoding to utf8)

  Attempts to put wide characters into environment variable keys via `%ENV` now provoke this warning.

### Changes to Existing Diagnostics

- Error `%s` in expansion of `%s`

  An error was encountered in handling a user-defined property (‘‘User-Defined Character Properties’’ in perlunicode). These are programmer written subroutines, hence subject to errors that may prevent them from compiling or running.

- Infinite recursion in user-defined property

  A user-defined property (‘‘User-Defined Character Properties’’ in perlunicode) can depend on the definitions of other user-defined properties. If the chain of dependencies leads back to this property, infinite recursion would occur, were it not for the check that raised this error.

- Timeout waiting for another thread to define \p{%s}

  The first time a user-defined property (‘‘User-Defined Character Properties’’ in perlunicode) is used, its definition is looked up and converted into an internal form for more efficient handling in subsequent uses. There could be a race if two or more threads tried to do this processing nearly simultaneously.

- Unknown user-defined property name \p{%s}

  You specified to use a property within the `\p{...}` which was a syntactically valid user-defined property, but no definition was found for it

- Too few arguments for subroutine ’%s’ (got `%d`; expected `%d`)

  Subroutine argument-count mismatch errors now include the number of given and expected arguments.

- Too many arguments for subroutine ’%s’ (got `%d`; expected `%d`)

  Subroutine argument-count mismatch errors now include the number of given and expected arguments.

- Lost precision when `%s` `%f` by 1

  This warning was only issued for positive too-large values when incrementing, and only for negative ones when decrementing. It is now issued for both positive or negative too-large values. [GH #18333 <https://github.com/Perl/perl5/issues/18333>]

- \K not permitted in lookahead/lookbehind in regex; marked by <-- HERE in m/%s/

  This error was incorrectly produced in some cases involving nested lookarounds. This has been fixed. [GH #18123 <https://github.com/Perl/perl5/issues/18123>]

- Use of uninitialized value%s

  This warning may now include the array or hash index when the uninitialized value is the result of an element not found. This will only happen if the index is a simple non-magical variable.

## Utility Changes

### perl5db.pl (the debugger)

- New option: `HistItemMinLength`

  This option controls the minimum length a command must be to get stored in history. Traditionally, this has been fixed at 2. Changes to the debugger are often perilous, and new bugs should be reported so the debugger can be debugged.

- Fix to `i` and `l` commands

  The `i $var` and `l $var` commands work again with lexical variables.

## Configuration and Compilation

- Prevented incpath to spill into libpth
- Use realpath if available. (This might catch more duplicate paths.)
- Only include real existing paths.
- Filter inc paths out of libpth.
- stadtx hash support has been removed

  stadtx support has been entirely removed. Previously, it could be requested with `PERL_HASH_FUNC_STADTX`, and was default in 64−bit builds. It has been replaced with SipHash. SipHash has been more rigorously reviewed than stadtx.

- Configure

  A new probe checks for buggy libc implementations of the `gcvt`/`qgcvt` functions. [GH #18170 <https://github.com/Perl/perl5/issues/18170>]

- `−Dusedefaultstrict`

  Perl can now be built with strict on by default (using the configuration option `−Dusedefaultstrict`.

  These strict defaults do not apply when `perl` is run via `−e` or `−E`.

  This setting provides a diagnostic mechanism intended for development purposes only and is thus undefined by default.

- The minimum supported Bison version is now 2.4, and the maximum is 3.7.
- Newer 64−bit versions of the Intel C/C++ compiler are now recognised and have the correct flags set.
- We now trap SIGBUS when *Configure* checks for `va_copy`.

  On several systems the attempt to determine if we need `va_copy` or similar results in a SIGBUS instead of the expected SIGSEGV, which previously caused a core dump.

  [GH #18148 <https://github.com/Perl/perl5/issues/18148>]

## Testing

Tests were added and changed to reflect the other additions and changes in this release. Furthermore, these significant changes were made:

- Split Config-dependent tests in *t/opbasic/arith.t* to *t/op/arith2.t*
- *t/re/opt.t* was added, providing a test harness for regexp optimization. [GH #18213 <https://github.com/Perl/perl5/pull/18213>]
- A workaround for CPAN distributions needing dot in `@INC` has been removed [GH #18394 <https://github.com/Perl/perl5/pull/18394>]. All distributions that previously required the workaround have now been adapted.

- When testing in parallel on many-core platforms, you can now cause the test suite to finish somewhat earlier, but with less logical ordering of the tests, by setting

  ```
  PERL_TEST_HARNESS_ASAP=1
  ```

  while running the test suite.

## Platform Support

### New Platforms

9front

Allow building Perl on i386 9front systems (a fork of plan9).

### Updated Platforms

Plan9

Improve support for Plan9 on i386 platforms.

MacOS (Darwin)

The hints file for darwin has been updated to handle future MacOS versions beyond 10. [GH #17946 <https://github.com/Perl/perl5/issues/17946>]

### Discontinued Platforms

Symbian

Support code relating to Symbian has been removed. Symbian was an operating system for mobile devices. The port was last updated in July 2009, and the platform itself in October 2012.

### Platform-Specific Notes

DragonFlyBSD

Tests were updated to workaround DragonFlyBSD bugs in tc*() functions <https://bugs.dragonflybsd.org/issues/3252> and ctime updates <https://bugs.dragonflybsd.org/issues/3251>.

Mac OS X

A number of system libraries no longer exist as actual files on Big Sur, even though `dlopen` will pretend they do, so now we fall back to `dlopen` if a library file can not be found. [GH #18407 <https://github.com/Perl/perl5/issues/18407>]

Windows

Reading non-ASCII characters from the console when its codepage was set to 65001 (UTF-8) was broken due to a bug in Windows. A workaround for this problem has been implemented. [GH #18701 <https://github.com/Perl/perl5/issues/18701>]

Building with mingw.org compilers (version 3.4.5 or later) using mingw runtime versions < 3.22 now works again. This was broken in Perl 5.31.4.

Building with mingw.org compilers (version 3.4.5 or later) using mingw runtime versions >= 3.21 now works (for compilers up to version 5.3.0).

*Makefile.mk*, and thus support for dmake, has been removed. It is still possible to build Perl on Windows using nmake (Makefile) and GNU make (GNUmakefile). [GH #18511 <https://github.com/Perl/perl5/pull/18511>]

perl can now be built with `USE_QUADMATH` on MS Windows using (32-bit and 64-bit) mingw-w64 ports of gcc. [GH #18465 <https://github.com/Perl/perl5/pull/18465>]

The *pl2bat.pl* utility now needs to `use ExtUtils::PL2Bat`. This could cause failures in parallel builds.

Windows now supports **symlink()** and **readlink()**, and **lstat()** is no longer an alias for **stat()**. [GH #18005 <https://github.com/Perl/perl5/issues/18005>].

Unlike POSIX systems, creating a symbolic link on Windows requires either elevated privileges or Windows 10 1703 or later with Developer Mode enabled.

**stat()**, including `stat FILEHANDLE`, and **lstat()** now uses our own implementation that populates the device `dev` and inode numbers `ino` returned rather than always returning zero. The number of links `nlink` field is now always populated.

`${^WIN32_SLOPPY_STAT}` previously controlled whether the `nlink` field was populated

requiring a separate Windows API call to fetch, since `nlink` and the other information required for `stat()` is now retrieved in a single API call.

The `-r` and `-w` operators now return true for the `STDIN`, `STDOUT` and `STDERR` handles. Unfortunately it still won't return true for duplicates of those handles. [GH #8502 <https://github.com/Perl/perl5/issues/8502>].

The times returned by **stat()** and **lstat()** are no longer incorrect across Daylight Savings Time adjustments. [GH #6080 <https://github.com/Perl/perl5/issues/6080>].

`-x` on a filehandle should now match `-x` on the corresponding filename on Vista or later. [GH #4145 <https://github.com/Perl/perl5/issues/4145>].

`-e '"'` no longer incorrectly returns true. [GH #12431 <https://github.com/Perl/perl5/issues/12431>].

The same manifest is now used for Visual C++ and gcc builds.

Previously, MSVC builds were using the **/manifestdependency** flag instead of embedding *perlexe.manifest*, which caused issues such as `GetVersionEx()` returning the wrong version number on Windows 10.

z/OS

The locale categories `LC_SYNTAX` and `LC_TOD` are now recognized. Perl doesn't do anything with these, except it now allows you to specify them. They are included in `LC_ALL`.

## Internal Changes

- Corrected handling of double and long double parameters for perl's implementation of formatted output for `-Dusequadmath` builds.

  This applies to `PerlIO_printf()`, `croak()`, `warn()`, `sv_catpvf()` and their variants.

  Previously in `quadmath` builds, code like:

  ```
  PerlIO_printf(PerlIO_stderr(), "%g", somedouble);
  ```

  or

  ```
  PerlIO_printf(PerlIO_stderr(), "%Lg", somelongdouble);
  ```

  would erroneously throw an exception "panic: quadmath invalid format ..."', since the code added for quadmath builds assumed `NV`s were the only floating point format passed into these functions.

  This code would also process the standard C long double specifier `L` as if it expected an `NV` (`__float128` for quadmath builds), resulting in undefined behaviour.

  These functions now correctly accept doubles, long doubles and NVs.

- Previously the right operand of bitwise shift operators (shift amount) was implicitly cast from IV to int, but it might lead wrong results if IV does not fit in int.

  And also, shifting INT_MIN bits used to yield the shiftee unchanged (treated as 0-bit shift instead of negative shift).

- A set of `cop_hints_exists_{pv,pvn,pvs,sv}` functions was added, to support checking for the existence of keys in the hints hash of a specific cop without needing to create a mortal copy of said value.

- An aid has been added for using the `DEBUG` macros when debugging XS or C code. The comments in *perl.h* describe `DEBUG_PRE_STMTS` and `DEBUG_POST_STMTS`. which you can `#define` to do things like save and restore `errno`, in case the `DEBUG` calls are interfering with that, or to display timestamps, or which thread it's coming from, or the location of the call, or whatever. You can make a quick hack to help you track something down without having to edit individual `DEBUG` calls.

- Make `REFCOUNTED_HE_EXISTS` available outside of core

- All `SvTRUE`-ish functions now evaluate their arguments exactly once. In 5.32, plain "`SvTRUE`" in perlapi was changed to do that; now the rest do as well.

- Unicode is now a first class citizen when considering the pattern /A*B/ where A and B are arbitrary. The pattern matching code tries to make a tight loop to match the span of A's. The logic of this was now really updated with support for UTF−8.
- The re module has a new function `optimization`, which can return a hashref of optimization data discovered about a compiled regexp.
- The `PERL_GLOBAL_STRUCT` compilation option has been removed, and with it the need or the `dVAR` macro. `dVAR` remains defined as a no-op outside `PERL_CORE` for backwards compatiblity with XS modules.
- A new savestack type `SAVEt_HINTS_HH` has been added, which neatens the previous behaviour of `SAVEt_HINTS`. On previous versions the types and values pushed to the save stack would depend on whether the hints included the `HINT_LOCALIZE_HH` bit, which complicates external code that inspects the save stack. The new version uses a different savestack type to indicate the difference.
- A new API function ''av_count'' in perlapi has been added which gives a clearly named way to find how many elements are in an array.

## Selected Bug Fixes

- Setting `%ENV` now properly handles upgraded strings in the key. Previously Perl sent the SV's internal PV directly to the OS; now it will handle keys as it has handled values since 5.18: attempt to downgrade the string first; if that fails then warn and use the utf8 form.
- Fix a memory leak in regcomp.c [GH #18604 <https://github.com/Perl/perl5/issues/18604>]
- pack/unpack format 'D' now works on all systems that could support it

  Previously if `NV == long double`, now it is supported on all platforms that have long doubles. In particular that means it is now also supported on quadmath platforms.
- Skip trying to constant fold an incomplete op tree [GH #18380 <https://github.com/Perl/perl5/issues/18380>]

  Constant folding of chained comparison op trees could fail under certain conditions, causing perl to crash. As a quick fix, constant folding is now skipped for such op trees. This also addresses [GH #17917 <https://github.com/Perl/perl5/issues/17917>].
- `%g` formatting broken on Ubuntu−18.04, `NVSIZE == 8` [GH #18170 <https://github.com/Perl/perl5/issues/18170>]

  Buggy libc implementations of the `gcvt` and `qgcvt` functions caused `(s)printf` to incorrectly truncate `%g` formatted numbers. A new Configure probe now checks for this, with the result that the libc `sprintf` will be used in place of `gcvt` and `qgcvt`.

  Tests added as part of this fix also revealed related problems in some Windows builds. The makefiles for MINGW builds on Windows have thus been adjusted to use `USE_MINGW_ANSI_STDIO` by default, ensuring that they also provide correct `(s)printf` formatting of numbers.
- *op.c*: croak on `my $_` when `use utf8` is in effect [GH #18449 <https://github.com/Perl/perl5/issues/18449>]

  The lexical topic feature experiment was removed in Perl v5.24 and declaring `my $_` became a compile time error. However, it was previously still possible to make this declaration if `use utf8` was in effect.
- *regexec.c*: Fix assertion failure [GH #18451 <https://github.com/Perl/perl5/issues/18451>]

  Fuzzing triggered an assertion failure in the regexp engine when too many characters were copied into a buffer.
- **semctl()**, **msgctl()**, and **shmctl()** now properly reset the UTF−8 flag on the `ARG` parameter if it's modified for `IPC_STAT` or `GETALL` operations.
- `semctl()`, `msgctl()`, and `shmctl()` now attempt to downgrade the `ARG` parameter if its value is being used as input to `IPC_SET` or `SETALL` calls. A failed downgrade will thrown an exception.

- In cases where `semctl()`, `msgctl()` or `shmctl()` would treat the `ARG` parameter as a pointer, an undefined value no longer generates a warning. In most such calls the pointer isn't used anyway and this allows you to supply `undef` for a value not used by the underlying function.
- **semop()** now downgrades the `OPSTRING` parameter, **msgsnd()** now downgrades the `MSG` parameter and shmwrite now downgrades the `STRING` parameter to treat them as bytes. Previously they would be left upgraded, providing a corrupted structure to the underlying function call.
- **msgrcv()** now properly resets the UTF-8 flag the `VAR` parameter when it is modified. Previously the UTF-8 flag could be left on, resulting in a possibly corrupt result in `VAR`.
- Magic is now called correctly for stacked file test operators. [GH #18293 <https://github.com/Perl/perl5/issues/18293>]
- The `@ary = split(...)` optimization no longer switches in the target array as the value stack. [GH #18232 <https://github.com/Perl/perl5/issues/18232>] Also see discussion at <https://github.com/Perl/perl5/pull/18014#issuecomment-671299506>.
- Fixed a bug in which some regexps with recursive subpatterns matched incorrectly.

  [GH #18096 <https://github.com/Perl/perl5/issues/18096>]
- On Win32, `waitpid(-1, WNOHANG)` could sometimes have a very large timeout. [GH #16529 <https://github.com/Perl/perl5/issues/16529>]
- `MARK` and hence `items` are now correctly initialized in `BOOT` XSUBs.
- Some list assignments involving `undef` on the left-hand side were over-optimized and produced incorrect results. [GH #16685 <https://github.com/Perl/perl5/issues/16685>], [GH #17816 <https://github.com/Perl/perl5/issues/17816>]

## Known Problems

None

## Errata From Previous Releases

None

## Obituary

Kent Fredric (KENTNL) passed away in February 2021. A native of New Zealand and a self-described ''huge geek,'' Kent was the author or maintainer of 178 CPAN distributions, the Perl maintainer for the Gentoo Linux distribution and a contributor to the Perl core distribution. He is mourned by his family, friends and open source software communities worldwide.

## Acknowledgements

Perl 5.34.0 represents approximately 11 months of development since Perl 5.32.0 and contains approximately 280,000 lines of changes across 2,100 files from 78 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 150,000 lines of changes to 1,300 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.34.0:

Aaron Crane, Adam Hartley, Andy Dougherty, Ben Cornett, Branislav Zahradník, brian d foy, Chris 'BinGOs' Williams, Christian Walde (Mithaldu), Craig A. Berry, Dagfinn Ilmari Mannsåker, Dan Book, Daniel Böhmer, Daniel Laügt, Dan Kogai, David Cantrell, David Mitchell, Dominic Hamon, E. Choroba, Ed J, Eric Herman, Eric Lindblad, Eugene Alvin Villar, Felipe Gasper, Giovanni Tataranni, Graham Knop, Graham Ollis, Hauke D, H.Merijn Brand, Hugo van der Sanden, Ichinose Shogo, Ivan Baidakou, Jae Bradley, James E Keenan, Jason McIntosh, jkahrman, John Karr, John Lightsey, Kang-min Liu, Karen Etheridge, Karl Williamson, Keith Thompson, Leon Timmermans, Marc Reisner, Marcus Holland-Moritz, Max Maischein, Michael G Schwern, Nicholas Clark, Nicolas R., Paul Evans, Petr PísaX, raiph, Renee Baecker, Ricardo Signes, Richard Leach, Romano, Ryan Voots, Samanta Navarro, Samuel Thibault, Sawyer X, Scott Baker, Sergey Poznyakoff, Sevan Janiyan, Shirakata Kentaro, Shlomi Fish, Sisyphus, Sizhe Zhao, Steve Hay, TAKAI Kousuke, Thibault Duponchelle, Todd Rinaldo, Tomasz Konojacki, Tom Hukins, Tom Stellard, Tony Cook, vividsnow, Yves Orton, Zakariyya Mughal, XXXXXX XXXXXXXX.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl5341delta − what is new for perl v5.34.1

## DESCRIPTION

This document describes differences between the 5.34.0 release and the 5.34.1 release.

If you are upgrading from an earlier release such as 5.33.0, first read perl5340delta, which describes differences between 5.33.0 and 5.34.0.

## Incompatible Changes

There are no changes intentionally incompatible with 5.34.0. If any exist, they are bugs, and we request that you submit a report. See ‘‘Reporting Bugs’’ below.

## Modules and Pragmata

### Updated Modules and Pragmata

- B::Deparse has been upgraded from version 1.56 to 1.57.
- Encode has been upgraded from version 3.08 to 3.08_01.
- GDBM_File has been upgraded from version 1.19 to 1.19_01.
- Module::CoreList has been upgraded from version 5.20210520 to 5.20220313.
- perl5db.pl has been upgraded from version 1.60 to 1.60_01.

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

### Platform-Specific Notes

Windows

- Support for compiling perl on Windows using Microsoft Visual Studio 2022 (containing Visual C++ 14.3) has been added.

## Selected Bug Fixes

- B::Deparse now correctly handles try/catch blocks with more complex scopes. [GH #18874 <https://github.com/Perl/perl5/issues/18874>]
- try/catch now correctly returns the last evaluated expression when the catch block has more than one statement. [GH #18855 <https://github.com/Perl/perl5/issues/18855>]

## Acknowledgements

Perl 5.34.1 represents approximately 10 months of development since Perl 5.34.0 and contains approximately 4,600 lines of changes across 60 files from 23 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 1,100 lines of changes to 18 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.34.1:

Andrew Fresh, Atsushi Sugawara, Chris ’BinGOs’ Williams, Dan Book, Hugo van der Sanden, James E Keenan, Karen Etheridge, Leon Timmermans, Matthew Horsfall, Max Maischein, Michiel Beijen, Neil Bowers, Nicolas R., Paul Evans, Renee Baecker, Ricardo Signes, Richard Leach, Sawyer X, Sergey Poznyakoff, Steve Hay, Tomasz Konojacki, Tony Cook, Yves Orton.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl’s core. We’re grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl’s historical contributors, please see the *AUTHORS* file in the Perl source distribution.

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at

<https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perldelta − what is new for perl v5.36.0

## DESCRIPTION

This document describes differences between the 5.34.0 release and the 5.36.0 release.

## Core Enhancements

### use v5.36

As always, `use v5.36` turns on the feature bundle for that version of Perl.

The 5.36 bundle enables the `signatures` feature. Introduced in Perl version 5.20.0, and modified several times since, the subroutine signatures feature is now no longer considered experimental. It is now considered a stable language feature and no longer prints a warning.

```
use v5.36;

sub add ($x, $y) {
  return $x + $y;
}
```

Despite this, certain elements of signatured subroutines remain experimental; see below.

The 5.36 bundle enables the `isa` feature. Introduced in Perl version 5.32.0, this operator has remained unchanged since then. The operator is now considered a stable language feature. For more detail see ‘‘Class Instance Operator’’ in perlop.

The 5.36 bundle also *disables* the features `indirect`, and `multidimensional`. These will forbid, respectively: the use of ‘‘indirect’’ method calls (like `$x = new Class;`); the use of a list expression as a hash key to simulate sparse multidimensional arrays. The specifics of these changes can be found in feature, but the short version is: this is a bit like having more `use strict` turned on, disabling features that cause more trouble than they’re worth.

Furthermore, `use v5.36` will also enable warnings as if you’d written `use warnings`.

Finally, with this release, the experimental `switch` feature, present in every feature bundle since they were introduced in v5.10, has been removed from the v5.36 bundle. If you want to use it (against our advice), you’ll have to enable it explicitly.

### −g command-line flag

A new command-line flag, −g, is available. It is a simpler alias for −0777.

For more information, see ‘‘−g’’ in perlrun.

### Unicode 14.0 is supported

See <https://www.unicode.org/versions/Unicode14.0.0/> for details.

### regex sets are no longer considered experimental

Prior to this release, the regex sets feature (officially named ‘‘Extended Bracketed Character Classes’’) was considered experimental. Introduced in Perl version 5.18.0, and modified several times since, this is now considered a stable language feature and its use no longer prints a warning. See ‘‘Extended Bracketed Character Classes’’ in perlrecharclass.

### Variable length lookbehind is mostly no longer considered experimental

Prior to this release, any form of variable length lookbehind was considered experimental. With this release the experimental status has been reduced to cover only lookbehind that contains capturing parenthesis. This is because it is not clear if

```
"aaz"=~/(?=z)(?<=(a|aa))/
```

should match and leave `$1` equaling ‘‘a’’ or ‘‘aa’’. Currently it will match the longest possible alternative, ‘‘aa’’. While we are confident that the overall construct will now match only when it should, we are not confident that we will keep the current ‘‘longest match’’ behavior.

### SIGFPE no longer deferred

Floating-point exceptions are now delivered immediately, in the same way as other ‘‘fault’’−like signals such as SIGSEGV. This means one has at least a chance to catch such a signal with a `$SIG{FPE}` handler, e.g. so that `die` can report the line in perl that triggered it.

### Stable boolean tracking

The ‘‘true’’ and ‘‘false’’ boolean values, often accessed by constructions like `!!0` and `!!1`, as well as being returned from many core functions and operators, now remember their boolean nature even through assignment into variables. The new function `is_bool()` in builtin can check whether a value has boolean nature.

This is likely to be useful when interoperating with other languages or data-type serialisation, among other places.

### iterating over multiple values at a time (experimental)

You can now iterate over multiple values at a time by specifying a list of lexicals within parentheses. For example,

```
for my ($key, $value) (%hash) { ... }
for my ($left, $right, $gripping) (@moties) { ... }
```

Prior to perl v5.36, attempting to specify a list after `for my` was a syntax error.

This feature is currently experimental and will cause a warning of category `experimental::for_list`. For more detail see ‘‘Compound Statements’’ in perlsyn. See also ‘‘builtin::indexed’’ in this document, which is a handy companion to n−at-a-time foreach.

### builtin functions (experimental)

A new core module builtin has been added, which provides documentation for new always-present functions that are built into the interpreter.

```
say "Reference type of arrays is ", builtin::reftype([]);
```

It also provides a lexical import mechanism for providing short name versions of these functions.

```
use builtin 'reftype';
say "Reference type of arrays is ", reftype([]);
```

This builtin function mechanism and the functions it provides are all currently **experimental**. We expect that `builtin` itself will cease to be experimental in the near future, but that individual functions in it may become stable on an ongoing basis. Other functions will be added to `builtin` over time.

For details, see builtin, but here’s a summary of builtin functions in v5.36:

builtin::trim
: This function treats its argument as a string, returning the result of removing all white space at its beginning and ending.

builtin::indexed
: This function returns a list twice as big as its argument list, where each item is preceded by its index within that list. This is primarily useful for using the new `foreach` syntax with multiple iterator variables to iterate over an array or list, while also tracking the index of each item:

```
use builtin 'indexed';

foreach my ($index, $val) (indexed @array) {
    ...
}
```

builtin::true, builtin::false, builtin::is_bool
: `true` and `false` return boolean true and false values. Perl is still perl, and doesn’t have strict typing of booleans, but these values will be known to have been created as booleans. `is_bool` will tell you whether a value was known to have been created as a boolean.

builtin::weaken, builtin::unweaken, builtin::is_weak
: These functions will, respectively: weaken a reference; strengthen a reference; and return whether a reference is weak. (A weak reference is not counted for garbage collection purposes. See perlref.) These can take the place of some similar routines in Scalar::Util.

builtin::blessed, builtin::refaddr, builtin::reftype
: These functions provide more data about references (or non-references, actually!) and can take the place of similar routines found in Scalar::Util.

builtin::ceil, builtin::floor

`ceil` returns the smallest integer greater than or equal to its argument. `floor` returns the largest integer less than or equal to its argument. These can take the place of similar routines found in POSIX.

### `defer` blocks (experimental)

This release adds support for `defer` blocks, which are blocks of code prefixed by the `defer` modifier. They provide a section of code which runs at a later time, during scope exit.

In brief, when a `defer` block is reached at runtime, its body is set aside to be run when the enclosing scope is exited. It is unlike a UNITCHECK (among other reasons) in that if the block *containing* the `defer` block is exited before the block is reached, it will not be run.

`defer` blocks can be used to take the place of ''scope guard'' objects where an object is passed a code block to be run by its destructor.

For more information, see ''defer blocks'' in perlsyn.

### try/catch can now have a `finally` block (experimental)

The experimental `try`/`catch` syntax has been extended to support an optional third block introduced by the `finally` keyword.

```
try {
    attempt();
    print "Success\n";
}
catch ($e) {
    print "Failure\n";
}
finally {
    print "This happens regardless\n";
}
```

This provides code which runs at the end of the `try`/`catch` construct, even if aborted by an exception or control-flow keyword. They are similar to `defer` blocks.

For more information, see ''Try Catch Exception Handling'' in perlsyn.

### non-ASCII delimiters for quote-like operators (experimental)

Perl traditionally has allowed just four pairs of string/pattern delimiters: `( )` `{ }` `[ ]` and `< >`, all in the ASCII range. Unicode has hundreds more possibilities, and using this feature enables many of them. When enabled, you can say `qrX X` for example, or `use utf8; qXstringX`. See ''The 'extra_paired_delimiters' feature'' in feature for details.

### `@_` is now experimental within signatured subs

Even though subroutine signatures are now stable, use of the legacy arguments array (`@_`) with a subroutine that has a signature *remains* experimental, with its own warning category. Silencing the `experimental::signatures` warning category is not sufficient to dismiss this. The new warning is emitted with the category name `experimental::args_array_with_signatures`.

Any subroutine that has a signature and tries to make use of the defaults argument array or an element thereof (`@_` or `$_[INDEX]`), either explicitly or implicitly (such as `shift` or `pop` with no argument) will provoke a warning at compile-time:

```
use v5.36;

sub f ($x, $y = 123) {
  say "The first argument is $_[0]";
}
```

```
Use of @_ in array element with signatured subroutine is experimental
at file.pl line 4.
```

The behaviour of code which attempts to do this is no longer specified, and may be subject to change in a future version.

## Incompatible Changes

### A physically empty sort is now a compile-time error

```
@a = sort @empty; # unaffected
@a = sort;        # now a compile-time error
@a = sort ();     # also a compile-time error
```

A bare sort used to be a weird way to create an empty list; now it croaks at compile time. This change is intended to free up some of the syntax space for possible future enhancements to `sort`.

## Deprecations

### `use VERSION` (where VERSION is below v5.11) after `use v5.11` is deprecated

When in the scope of `use v5.11` or later, a `use vX` line where *X* is lower than v5.11 will now issue a warning:

```
Downgrading a use VERSION declaration to below v5.11 is deprecated
```

For example:

```
use v5.14;
say "The say statement is permitted";
use v5.8;                               # This will print a warning
print "We must use print\n";
```

This is because the Perl team plans to change the behavior in this case. Since Perl v5.12 (and parts of v5.11), strict is enabled *unless it had previously been disabled*. In other words:

```
no strict;
use v5.12;  # will not enable strict, because "no strict" preceded it
$x = 1;     # permitted, despite no "my" declaration
```

In the future, this behavior will be eliminated and `use VERSION` will *always* enable strict for versions v5.12 and later.

Code which wishes to mix versions in this manner should use lexical scoping with block syntax to ensure that the differently versioned regions remain lexically isolated.

```
{
    use v5.14;
    say "The say statement is permitted";
}

{
    use v5.8;                           # No warning is emitted
    print "We must use print\n";
}
```

Of course, this is probably not something you ever need to do! If the first block compiles, it means you're using perl v5.14.0 or later.

## Performance Enhancements

- We now probe for compiler support for C11 thread local storage, and where available use this for ''implicit context'' for XS extensions making API calls for a threaded Perl build. This requires fewer function calls at the C level than POSIX thread specific storage. We continue to use the the pthreads approach if the C11 approach is not available.

  *Configure* run with the defaults will build an unthreaded Perl (which is slightly faster), but most operating systems ship a threaded Perl.

- Perl can now be configured to no longer allocate keys for large hashes from the shared string table.

  The same internal datatype (PVHV) is used for all of

  - Symbol tables
  - Objects (by default)
  - Associative arrays

  The shared string table was originally added to improve performance for blessed hashes used as objects, because every object instance has the same keys, so it is an optimisation to share memory

between them. It also makes sense for symbol tables, where derived classes will have the same keys (typically method names), and the OP trees built for method calls can also share memory. The shared string table behaves roughly like a cache for hash keys.

But for hashes actually used as associative arrays − mapping keys to values − typically the keys are not re-used in other hashes. For example, ‘‘seen’’ hashes are keyed by object IDs (or addresses), and logically these keys won’t repeat in other hashes.

Storing these ‘‘used just once’’ keys in the shared string table increases CPU and RAM use for no gain. For such keys the shared string table behaves as a cache with a 0% hit rate. Storing all the keys there increases the total size of the shared string table, as well as increasing the number of times it is resized as it grows. **Worse** − in any environment that has ‘‘copy on write’’ memory for child process (such as a pre-forking server), the memory pages used for the shared string table rapidly need to be copied as the child process manipulates hashes. Hence if most of the shared string table is such that keys are used only in one place, there is no benefit from re-use within the perl interpreter, but a high cost due to more pages for the OS to copy.

The perl interpreter can now be Configured to disable shared hash keys for ‘‘large’’ hashes (that are neither objects nor symbol tables). To do so, add `-Accflags='-DPERL_USE_UNSHARED_KEYS_IN_LARGE_HASHES'` to your *Configure* options. ‘‘Large’’ is a heuristic — currently the heuristic is that sharing is disabled when adding a key to a hash triggers allocation of more storage, and the hash has more than 42 keys.

This **might** cause slightly increased memory usage for programs that create (unblessed) data structures that contain multiple large hashes that share the same keys. But generally our testing suggests that for the specific cases described it is a win, and other code is unaffected.

- In certain scenarios, creation of new scalars is now noticeably faster.

  For example, the following code is now executing ~30% faster:

  ```
  $str = "A" x 64;
  for (0..1_000_000) {
      @svs = split //, $str
  }
  ```

  (You can read more about this one in [perl #19414] <https://github.com/Perl/perl5/pull/19414>.)

## Modules and Pragmata

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.38 to 2.40.
- Attribute::Handlers has been upgraded from version 1.01 to 1.02.
- attributes has been upgraded from version 0.33 to 0.34.
- B has been upgraded from version 1.82 to 1.83.
- B::Concise has been upgraded from version 1.004 to 1.006.
- B::Deparse has been upgraded from version 1.56 to 1.64.
- bignum has been upgraded from version 0.51 to 0.65.
- charnames has been upgraded from version 1.48 to 1.50.
- Compress::Raw::Bzip2 has been upgraded from version 2.101 to 2.103.
- Compress::Raw::Zlib has been upgraded from version 2.101 to 2.105.
- CPAN has been upgraded from version 2.28 to 2.33.
- Data::Dumper has been upgraded from version 2.179 to 2.184.
- DB_File has been upgraded from version 1.855 to 1.857.
- Devel::Peek has been upgraded from version 1.30 to 1.32.
- Devel::PPPort has been upgraded from version 3.62 to 3.68.

- diagnostics has been upgraded from version 1.37 to 1.39.
- Digest has been upgraded from version 1.19 to 1.20.
- DynaLoader has been upgraded from version 1.50 to 1.52.
- Encode has been upgraded from version 3.08 to 3.17.
- Errno has been upgraded from version 1.33 to 1.36.
- experimental has been upgraded from version 0.024 to 0.028.
- Exporter has been upgraded from version 5.76 to 5.77.
- ExtUtils::MakeMaker has been upgraded from version 7.62 to 7.64.
- ExtUtils::Miniperl has been upgraded from version 1.10 to 1.11.
- ExtUtils::ParseXS has been upgraded from version 3.43 to 3.45.
- ExtUtils::Typemaps has been upgraded from version 3.43 to 3.45.
- Fcntl has been upgraded from version 1.14 to 1.15.
- feature has been upgraded from version 1.64 to 1.72.
- File::Compare has been upgraded from version 1.1006 to 1.1007.
- File::Copy has been upgraded from version 2.35 to 2.39.
- File::Fetch has been upgraded from version 1.00 to 1.04.
- File::Find has been upgraded from version 1.39 to 1.40.
- File::Glob has been upgraded from version 1.33 to 1.37.
- File::Spec has been upgraded from version 3.80 to 3.84.
- File::stat has been upgraded from version 1.09 to 1.12.
- FindBin has been upgraded from version 1.52 to 1.53.
- GDBM_File has been upgraded from version 1.19 to 1.23.
- Hash::Util has been upgraded from version 0.25 to 0.28.
- Hash::Util::FieldHash has been upgraded from version 1.21 to 1.26.
- HTTP::Tiny has been upgraded from version 0.076 to 0.080.
- I18N::Langinfo has been upgraded from version 0.19 to 0.21.
- if has been upgraded from version 0.0609 to 0.0610.
- IO has been upgraded from version 1.46 to 1.50.
- IO-Compress has been upgraded from version 2.102 to 2.106.
- IPC::Open3 has been upgraded from version 1.21 to 1.22.
- JSON::PP has been upgraded from version 4.06 to 4.07.
- libnet has been upgraded from version 3.13 to 3.14.
- Locale::Maketext has been upgraded from version 1.29 to 1.31.
- Math::BigInt has been upgraded from version 1.999818 to 1.999830.
- Math::BigInt::FastCalc has been upgraded from version 0.5009 to 0.5012.
- Math::BigRat has been upgraded from version 0.2614 to 0.2621.
- Module::CoreList has been upgraded from version 5.20210520 to 5.20220520.
- mro has been upgraded from version 1.25_001 to 1.26.
- NEXT has been upgraded from version 0.68 to 0.69.
- Opcode has been upgraded from version 1.50 to 1.57.
- open has been upgraded from version 1.12 to 1.13.

- overload has been upgraded from version 1.33 to 1.35.
- perlfaq has been upgraded from version 5.20210411 to 5.20210520.
- PerlIO has been upgraded from version 1.11 to 1.12.
- Pod::Functions has been upgraded from version 1.13 to 1.14.
- Pod::Html has been upgraded from version 1.27 to 1.33.
- Pod::Simple has been upgraded from version 3.42 to 3.43.
- POSIX has been upgraded from version 1.97 to 2.03.
- re has been upgraded from version 0.41 to 0.43.
- Scalar::Util has been upgraded from version 1.55 to 1.62.
- sigtrap has been upgraded from version 1.09 to 1.10.
- Socket has been upgraded from version 2.031 to 2.033.
- sort has been upgraded from version 2.04 to 2.05.
- Storable has been upgraded from version 3.23 to 3.26.
- Sys::Hostname has been upgraded from version 1.23 to 1.24.
- Test::Harness has been upgraded from version 3.43 to 3.44.
- Test::Simple has been upgraded from version 1.302183 to 1.302190.
- Text::ParseWords has been upgraded from version 3.30 to 3.31.
- Text::Tabs has been upgraded from version 2013.0523 to 2021.0814.
- Text::Wrap has been upgraded from version 2013.0523 to 2021.0814.
- threads has been upgraded from version 2.26 to 2.27.
- threads::shared has been upgraded from version 1.62 to 1.64.
- Tie::Handle has been upgraded from version 4.2 to 4.3.
- Tie::Hash has been upgraded from version 1.05 to 1.06.
- Tie::Scalar has been upgraded from version 1.05 to 1.06.
- Tie::SubstrHash has been upgraded from version 1.00 to 1.01.
- Time::HiRes has been upgraded from version 1.9767 to 1.9770.
- Unicode::Collate has been upgraded from version 1.29 to 1.31.
- Unicode::Normalize has been upgraded from version 1.28 to 1.31.
- Unicode::UCD has been upgraded from version 0.75 to 0.78.
- UNIVERSAL has been upgraded from version 1.13 to 1.14.
- version has been upgraded from version 0.9928 to 0.9929.
- VMS::Filespec has been upgraded from version 1.12 to 1.13.
- VMS::Stdio has been upgraded from version 2.45 to 2.46.
- warnings has been upgraded from version 1.51 to 1.58.
- Win32 has been upgraded from version 0.57 to 0.59.
- XS::APItest has been upgraded from version 1.16 to 1.22.
- XS::Typemap has been upgraded from version 0.18 to 0.19.
- XSLoader has been upgraded from version 0.30 to 0.31.

## Documentation

### New Documentation

*Porting/vote_admin_guide.pod*

This document provides the process for administering an election or vote within the Perl Core Team.

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, open an issue at <https://github.com/Perl/perl5/issues>.

Additionally, the following selected changes have been made:

*perlapi*

- This has been cleaned up some, and more than 80% of the (previously many) undocumented functions have now either been documented or deemed to have been inappropriately marked as API.

  As always, Patches Welcome!

*perldeprecation*

- notes the new location for functions moved from Pod::Html to Pod::Html::Util that are no longer intended to be used outside of core.

*perlexperiment*

- notes the `:win32` IO pseudolayer is removed (this happened in 5.35.2).

*perlgov*

- The election process has been finetuned to allow the vote to be skipped if there are no more candidates than open seats.
- A special election is now allowed to be postponed for up to twelve weeks, for example until a normal election.

*perlop*

- now notes that an invocant only needs to be an object or class name for method calls, not for subroutine references.

*perlre*

- Updated to discourage the use of the /d regexp modifier.

*perlrun*

- **−?** is now a synonym for **−h**
- **−g** is now a synonym for **−0777**

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Can’t ‘‘%s’’ out of a ‘‘defer’’ block

  (F) An attempt was made to jump out of the scope of a defer block by using a control-flow statement such as `return`, `goto` or a loop control. This is not permitted.

- Can’t modify `%s` in `%s` (for scalar assignment to `undef`)

  Attempting to perform a scalar assignment to `undef`, for example via `undef = $foo;`, previously triggered a fatal runtime error with the message "Modification of a read-only value attempted.‘‘ It is more helpful to detect such attempted assignments prior to runtime, so they are now compile time errors, resulting in the message ’’Can’t modify undef operator in scalar assignment".

- panic: newFORLOOP, `%s`

  The parser failed an internal consistency check while trying to parse a `foreach` loop.

*New Warnings*

- Built-in function ’%s’ is experimental

  A call is being made to a function in the `builtin::` namespace, which is currently experimental.

- defer is experimental

  The `defer` block modifier is experimental. If you want to use the feature, disable the warning with `no warnings 'experimental::defer'`, but know that in doing so you are taking the risk that your code may break in a future Perl version.

- Downgrading a use VERSION declaration to below v5.11 is deprecated

  This warning is emitted on a `use VERSION` statement that requests a version below v5.11 (when the effects of `use strict` would be disabled), after a previous declaration of one having a larger number (which would have enabled these effects)

- for my (...) is experimental

  This warning is emitted if you use `for` to iterate multiple values at a time. This syntax is currently experimental and its behaviour may change in future releases of Perl.

- Implicit use of `@_` in `%s` with signatured subroutine is experimental

  An expression that implicitly involves the `@_` arguments array was found in a subroutine that uses a signature.

- Use of `@_` in `%s` with signatured subroutine is experimental

  An expression involving the `@_` arguments array was found in a subroutine that uses a signature.

- Wide character in `$0`

  Attempts to put wide characters into the program name (`$0`) now provoke this warning.

### Changes to Existing Diagnostics

- '/' does not take a repeat count in `%s`

  This warning used to not include the `in %s`.

- Subroutine `%s` redefined

  Localized subroutine redefinitions no longer trigger this warning.

- unexpected constant lvalue entersub entry via type/targ `%d:`%d" now has a panic prefix

  This makes it consistent with other checks of internal consistency when compiling a subroutine.

- Useless use of sort in scalar context is now in the new `scalar` category.

  When `sort` is used in scalar context, it provokes a warning that doing this is not useful. This warning used to be in the `void` category. A new category for warnings about scalar context has now been added, called `scalar`.

- Removed a number of diagnostics

  Many diagnostics that have been removed from the perl core across many years have now *also* been removed from the documentation.

## Configuration and Compilation

- The Perl C source code now uses some C99 features, which we have verified are supported by all compilers we target. This means that Perl's headers now contain some code that is legal in C99 but not C89.

  This may cause problems for some XS modules that unconditionally add `-Werror=declaration-after-statement` to their C compiler flags if compiling with gcc or clang. Earlier versions of Perl support long obsolete compilers that are strict in rejecting certain C99 features, particularly mixed declarations and code, and hence it makes sense for XS module authors to audit that their code does not violate this. However, doing this is now only possible on these earlier versions of Perl, hence these modules need to be changed to only add this flag for `<$] < 5.035005>`.

- The makedepend step is now run in parallel by using make

  When using MAKEFLAGS=-j8, this significantly reduces the time required for:

  ```
  sh ./makedepend MAKE=make cflags
  ```

- *Configure* now tests whether `#include <xlocale.h>` is required to use the POSIX 1003 thread-safe locale functions or some related extensions. This prevents problems where a non-public *xlocale.h* is removed in a library update, or *xlocale.h* isn't intended for public use. (github #18936 <https://github.com/Perl/perl5/pull/18936>)

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

## Platform Support

### Windows

- Support for old MSVC++ (pre−VC12) has been removed

  These did not support C99 and hence can no longer be used to compile perl.

- Support for compiling perl on Windows using Microsoft Visual Studio 2022 (containing Visual C++ 14.3) has been added.

- The :win32 IO layer has been removed. This experimental replacement for the :unix layer never reached maturity in its nearly two decades of existence.

### VMS

`keys %ENV` on VMS returns consistent results

On VMS entries in the `%ENV` hash are loaded from the OS environment on first access, hence the first iteration of `%ENV` requires the entire environment to be scanned to find all possible keys. This initialisation had always been done correctly for full iteration, but previously was not happening for `%ENV` in scalar context, meaning that `scalar %ENV` would return 0 if called before any other `%ENV` access, or would only return the count of keys accessed if there had been no iteration.

These bugs are now fixed − `%ENV` and `keys %ENV` in scalar context now return the correct result − the count of all keys in the environment.

### Discontinued Platforms

AT&T UWIN

UWIN is a UNIX compatibility layer for Windows. It was last released in 2012 and has been superseded by Cygwin these days.

DOS/DJGPP

DJGPP is a port of the GNU toolchain to 32−bit x86 systems running DOS. The last known attempt to build Perl on it was on 5.20, which only got as far as building miniperl.

NetWare

Support code for Novell NetWare has been removed. NetWare was a server operating system by Novell. The port was last updated in July 2002, and the platform itself in May 2009.

Unrelated changes accidentally broke the build for the NetWare port in September 2009, and in 12 years no-one has reported this.

### Platform-Specific Notes

z/OS

This update enables us to build EBCDIC static/dynamic and 31−bit/64−bit addressing mode Perl. The number of tests that pass is consistent with the baseline before these updates.

These changes also provide the base support to be able to provide ASCII static/dynamic and 31−bit/64−bit addressing mode Perl.

The z/OS (previously called OS/390) README was updated to describe ASCII and EBCDIC builds.

## Internal Changes

- Since the removal of PERL_OBJECT in Perl 5.8, PERL_IMPLICIT_CONTEXT and MULTIPLICITY have been synonymous and they were being used interchangeably. To simplify the code, all instances of PERL_IMPLICIT_CONTEXT have been replaced with MULTIPLICITY.

  PERL_IMPLICIT_CONTEXT will remain defined for compatibility with XS modules.

- The API constant formerly named `G_ARRAY`, indicating list context, has now been renamed to a more accurate `G_LIST`. A compatibilty macro `G_ARRAY` has been added to allow existing code to work unaffected. New code should be written using the new constant instead. This is supported by `Devel::PPPort` version 3.63.

- Macros have been added to *perl.h* to facilitate version comparisons: `PERL_GCC_VERSION_GE`, `PERL_GCC_VERSION_GT`, `PERL_GCC_VERSION_LE` and `PERL_GCC_VERSION_LT`.

  Inline functions have been added to *embed.h* to determine the position of the least significant 1 bit in a word: `lsbit_pos32` and `lsbit_pos64`.

- `Perl_ptr_table_clear` has been deleted. This has been marked as deprecated since v5.14.0 (released in 2011), and is not used by any code on CPAN.

- Added new boolean macros and functions. See ‘‘Stable boolean tracking’’ for related information and perlapi for documentation.

  - sv_setbool
  - sv_setbool_mg
  - SvIsBOOL

- Added 4 missing functions for dealing with RVs:

  - sv_setrv_noinc
  - sv_setrv_noinc_mg
  - sv_setrv_inc
  - sv_setrv_inc_mg

- `xs_handshake()`'s two failure modes now provide distinct messages.

- Memory for hash iterator state (`struct xpvhv_aux`) is now allocated as part of the hash body, instead of as part of the block of memory allocated for the main hash array.

- A new **phase_name()** interface provides access to the name for each interpreter phase (i.e., PL_phase value).

- The `pack` behavior of `U` has changed for EBCDIC.

- New equality-test functions `sv_numeq` and `sv_streq` have been added, along with `..._flags`-suffixed variants. These expose a simple and consistent API to perform numerical or string comparison which is aware of operator overloading.

- Reading the string form of an integer value no longer sets the flag `SVf_POK`. The string form is still cached internally, and still re-read directly by the macros `SvPV(sv)` *etc* (inline, without calling a C function). XS code that already calls the APIs to get values will not be affected by this change. XS code that accesses flags directly instead of using API calls to express its intent *might* break, but such code likely is already buggy if passed some other values, such as floating point values or objects with string overloading.

  This small change permits code (such as JSON serializers) to reliably determine between

  - a value that was initially **written** as an integer, but then **read** as a string

    ```
    my $answer = 42;
    print "The answer is $answer\n";
    ```

  - that same value that was initially **written** as a string, but then **read** as an integer

    ```
    my $answer = "42";
    print "That doesn't look right\n"
        unless $answer == 6 * 9;
    ```

  For the first case (originally written as an integer), we now have:

  ```
  use Devel::Peek;
  my $answer = 42;
  Dump ($answer);
  my $void = "$answer";
  print STDERR "\n";
  Dump($answer)
  ```

```
SV = IV(0x562538925778) at 0x562538925788
  REFCNT = 1
  FLAGS = (IOK,pIOK)
  IV = 42

SV = PVIV(0x5625389263c0) at 0x562538925788
  REFCNT = 1
  FLAGS = (IOK,pIOK,pPOK)
  IV = 42
  PV = 0x562538919b50 "42"\0
  CUR = 2
  LEN = 10
```

For the second (originally written as a string), we now have:

```
use Devel::Peek;
my $answer = "42";
Dump ($answer);
my $void = $answer == 6 * 9;
print STDERR "\n";
Dump($answer)'


SV = PV(0x5586ffe9bfb0) at 0x5586ffec0788
  REFCNT = 1
  FLAGS = (POK,IsCOW,pPOK)
  PV = 0x5586ffee7fd0 "42"\0
  CUR = 2
  LEN = 10
  COW_REFCNT = 1

SV = PVIV(0x5586ffec13c0) at 0x5586ffec0788
  REFCNT = 1
  FLAGS = (IOK,POK,IsCOW,pIOK,pPOK)
  IV = 42
  PV = 0x5586ffee7fd0 "42"\0
  CUR = 2
  LEN = 10
  COW_REFCNT = 1
```

(One can't rely on the presence or absence of the flag `SVf_IsCOW` to determine the history of operations on a scalar.)

Previously both cases would be indistinguishable, with all 4 flags set:

```
SV = PVIV(0x55d4d62edaf0) at 0x55d4d62f0930
  REFCNT = 1
  FLAGS = (IOK,POK,pIOK,pPOK)
  IV = 42
  PV = 0x55d4d62e1740 "42"\0
  CUR = 2
  LEN = 10
```

(and possibly `SVf_IsCOW`, but not always)

This now means that if XS code *really* needs to determine which form a value was first written as, it should implement logic roughly

```
if (flags & SVf_IOK|SVf_NOK) && !(flags & SVf_POK)
    serialize as number
else if (flags & SVf_POK)
    serialize as string
else
    the existing guesswork ...
```

Note that this doesn't cover ''dualvars'' − scalars that report different values when asked for their string form or number form (such as `$!`). Most serialization formats cannot represent such duplicity.

*The existing guesswork* remains because as well as dualvars, values might be `undef`, references, overloaded references, typeglobs and other things that Perl itself can represent but do not map one-to-one into external formats, so need some amount of approximation or encapsulation.

- `sv_dump` (and Devel::PeekXs `Dump` function) now escapes high-bit octets in the PV as hex rather than octal. Since most folks understand hex more readily than octal, this should make these dumps a bit more legible. This does **not** affect any other diagnostic interfaces like `pv_display`.

## Selected Bug Fixes

- **utime()** now correctly sets errno/`$!` when called on a closed handle.
- The flags on the OPTVAL parameter to **setsockopt()** were previously checked before magic was called, possibly treating a numeric value as a packed buffer or vice versa. It also ignored the UTF−8 flag, potentially treating the internal representation of an upgraded SV as the bytes to supply to the **setsockopt()** system call. (github #18660 <https://github.com/Perl/perl5/issues/18660>)
- Only set IOKp, not IOK on $) and $(. This was issue #18955 <https://github.com/Perl/perl5/issues/18955>: This will prevent serializers from serializing these variables as numbers (which loses the additional groups). This restores behaviour from 5.16
- Use of the `mktables` debugging facility would cause perl to croak since v5.31.10; this problem has now been fixed.
- `makedepend` logic is now compatible with BSD make (fixes GH #19046 <https://github.com/Perl/perl5/issues/19046>).
- Calling `untie` on a tied hash that is partway through iteration now frees the iteration state immediately.

  Iterating a tied hash causes perl to store a copy of the current hash key to track the iteration state, with this stored copy passed as the second parameter to `NEXTKEY`. This internal state is freed immediately when tie hash iteration completes, or if the hash is destroyed, but due to an implementation oversight, it was not freed if the hash was untied. In that case, the internal copy of the key would persist until the earliest of

  1. `tie` was called again on the same hash
  2. The (now untied) hash was iterated (ie passed to any of `keys`, `values` or `each`)
  3. The hash was destroyed.

  This inconsistency is now fixed − the internal state is now freed immediately by `untie`.

  As the precise timing of this behaviour can be observed with pure Perl code (the timing of `DESTROY` on objects returned from `FIRSTKEY` and `NEXTKEY`) it's just possible that some code is sensitive to it.
- The `Internals::getcwd()` function added for bootstrapping miniperl in perl 5.30.0 is now only available in miniperl. [github #19122]
- Setting a breakpoint on a BEGIN or equivalently a `use` statement could cause a memory write to a freed `dbstate` op. [GH #19198 <https://github.com/Perl/perl5/issues/19198>]
- When bareword filehandles are disabled, the parser was interpreting any bareword as a filehandle, even when immediatey followed by parens.

## Errata From Previous Releases

- perl5300delta mistakenly identified a CVE whose correct identification is CVE-2015-1592.

## Obituaries

Raun ''Spider'' Boardman (SPIDB on CPAN), author of at least 66 commits to the Perl 5 core distribution between 1996 and 2002, passed away May 24, 2021 from complications of COVID. He will be missed.

David H. Adler (DHA) passed away on November 16, 2021. In 1997, David co-founded NY.pm, the first Perl user group, and in 1998 co-founded Perl Mongers to help establish other user groups across the globe. He was a frequent attendee at Perl conferences in both North America and Europe and well known for his role in organizing *Bad Movie Night* celebrations at those conferences. He also contributed to the work of the Perl Foundation, including administering the White Camel awards for community service. He will be missed.

## Acknowledgements

Perl 5.36.0 represents approximately a year of development since Perl 5.34.0 and contains approximately 250,000 lines of changes across 2,000 files from 82 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 190,000 lines of changes to 1,300 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.36.0:

Alyssa Ross, Andrew Fresh, Aristotle Pagaltzis, Asher Mancinelli, Atsushi Sugawara, Ben Cornett, Bernd, Biswapriyo Nath, Brad Barden, Bram, Branislav Zahradník, brian d foy, Chad Granum, Chris 'BinGOs' Williams, Christian Walde (Mithaldu), Christopher Yeleighton, Craig A. Berry, cuishuang, Curtis Poe, Dagfinn Ilmari Mannsåker, Dan Book, Daniel Laügt, Dan Jacobson, Dan Kogai, Dave Cross, Dave Lambley, David Cantrell, David Golden, David Marshall, David Mitchell, E. Choroba, Eugen Konkov, Felipe Gasper, François Perrad, Graham Knop, H.Merijn Brand, Hugo van der Sanden, Ilya Sashcheka, Ivan Panchenko, Jakub Wilk, James E Keenan, James Raspass, Karen Etheridge, Karl Williamson, Leam Hall, Leon Timmermans, Magnus Woldrich, Matthew Horsfall, Max Maischein, Michael G Schwern, Michiel Beijen, Mike Fulton, Neil Bowers, Nicholas Clark, Nicolas R, Niyas Sait, Olaf Alders, Paul Evans, Paul Marquess, Petar-Kaleychev, Pete Houston, Renee Baecker, Ricardo Signes, Richard Leach, Robert Rothenberg, Sawyer X, Scott Baker, Sergey Poznyakoff, Sergey Zhmylove, Sisyphus, Slaven Rezic, Steve Hay, Sven Kirmess, TAKAI Kousuke, Thibault Duponchelle, Todd Rinaldo, Tomasz Konojacki, Tomoyuki Sadahiro, Tony Cook, Unicode Consortium, Yves Orton, XXXXXX XXXXXXXX.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl561delta − what's new for perl v5.6.1

## DESCRIPTION

This document describes differences between the 5.005 release and the 5.6.1 release.

## Summary of changes between 5.6.0 and 5.6.1

This section contains a summary of the changes between the 5.6.0 release and the 5.6.1 release. More details about the changes mentioned here may be found in the *Changes* files that accompany the Perl source distribution. See perlhack for pointers to online resources where you can inspect the individual patches described by these changes.

### Security Issues

suidperl will not run /bin/mail anymore, because some platforms have a /bin/mail that is vulnerable to buffer overflow attacks.

Note that suidperl is neither built nor installed by default in any recent version of perl. Use of suidperl is highly discouraged. If you think you need it, try alternatives such as sudo first. See http://www.courtesan.com/sudo/ .

### Core bug fixes

This is not an exhaustive list. It is intended to cover only the significant user-visible changes.

`UNIVERSAL::isa()`

A bug in the caching mechanism used by `UNIVERSAL::isa()` that affected base.pm has been fixed. The bug has existed since the 5.005 releases, but wasn't tickled by base.pm in those releases.

Memory leaks

Various cases of memory leaks and attempts to access uninitialized memory have been cured. See ''Known Problems'' below for further issues.

Numeric conversions

Numeric conversions did not recognize changes in the string value properly in certain circumstances.

In other situations, large unsigned numbers (those above 2**31) could sometimes lose their unsignedness, causing bogus results in arithmetic operations.

Integer modulus on large unsigned integers sometimes returned incorrect values.

Perl 5.6.0 generated ''not a number'' warnings on certain conversions where previous versions didn't.

These problems have all been rectified.

Infinity is now recognized as a number.

qw(a\\b)

In Perl 5.6.0, qw(a\\b) produced a string with two backslashes instead of one, in a departure from the behavior in previous versions. The older behavior has been reinstated.

**caller()**

**caller()** could cause core dumps in certain situations. Carp was sometimes affected by this problem.

Bugs in regular expressions

Pattern matches on overloaded values are now handled correctly.

Perl 5.6.0 parsed m/\x{ab}/ incorrectly, leading to spurious warnings. This has been corrected.

The RE engine found in Perl 5.6.0 accidentally pessimised certain kinds of simple pattern matches. These are now handled better.

Regular expression debug output (whether through `use re 'debug'` or via `-Dr`) now looks better.

Multi-line matches like `"a\nxb\n" =~ /(?!\A)x/m` were flawed. The bug has been fixed.

Use of $& could trigger a core dump under some situations. This is now avoided.

Match variables `$1` et al., weren't being unset when a pattern match was backtracking, and the anomaly showed up inside `/...(?{ ... }).../` etc. These variables are now tracked correctly.

**pos()** did not return the correct value within s///ge in earlier versions. This is now handled correctly.

''slurp'' mode
**readline()** on files opened in ''slurp'' mode could return an extra "" at the end in certain situations. This has been corrected.

Autovivification of symbolic references to special variables
Autovivification of symbolic references of special variables described in perlvar (as in `${$num}`) was accidentally disabled. This works again now.

Lexical warnings
Lexical warnings now propagate correctly into `eval "..."`.

`use warnings qw(FATAL all)` did not work as intended. This has been corrected.

Lexical warnings could leak into other scopes in some situations. This is now fixed.

**warnings::enabled()** now reports the state of $^W correctly if the caller isn't using lexical warnings.

Spurious warnings and errors
Perl 5.6.0 could emit spurious warnings about redefinition of **dl_error()** when statically building extensions into perl. This has been corrected.

''our'' variables could result in bogus ''Variable will not stay shared'' warnings. This is now fixed.

''our'' variables of the same name declared in two sibling blocks resulted in bogus warnings about ''redeclaration'' of the variables. The problem has been corrected.

**glob()**
Compatibility of the builtin **glob()** with old csh-based glob has been improved with the addition of GLOB_ALPHASORT option. See `File::Glob`.

**File::Glob::glob()** has been renamed to **File::Glob::bsd_glob()** because the name clashes with the builtin **glob()**. The older name is still available for compatibility, but is deprecated.

Spurious syntax errors generated in certain situations, when **glob()** caused File::Glob to be loaded for the first time, have been fixed.

Tainting
Some cases of inconsistent taint propagation (such as within hash values) have been fixed.

The tainting behavior of **sprintf()** has been rationalized. It does not taint the result of floating point formats anymore, making the behavior consistent with that of string interpolation.

**sort()**
Arguments to **sort()** weren't being provided the right **wantarray()** context. The comparison block is now run in scalar context, and the arguments to be sorted are always provided list context.

**sort()** is also fully reentrant, in the sense that the sort function can itself call **sort()**. This did not work reliably in previous releases.

#line directives
#line directives now work correctly when they appear at the very beginning of `eval "..."`.

Subroutine prototypes
The (\&) prototype now works properly.

**map()**
**map()** could get pathologically slow when the result list it generates is larger than the source list. The performance has been improved for common scenarios.

Debugger
: Debugger exit code now reflects the script exit code.

  Condition `"0"` in breakpoints is now treated correctly.

  The `d` command now checks the line number.

  `$.` is no longer corrupted by the debugger.

  All debugger output now correctly goes to the socket if RemotePort is set.

PERL5OPT
: PERL5OPT can be set to more than one switch group. Previously, it used to be limited to one group of options only.

**chop()**
: chop(@list) in list context returned the characters chopped in reverse order. This has been reversed to be in the right order.

Unicode support
: Unicode support has seen a large number of incremental improvements, but continues to be highly experimental. It is not expected to be fully supported in the 5.6.x maintenance releases.

  **substr()**, **join()**, **repeat()**, **reverse()**, **quotemeta()** and string concatenation were all handling Unicode strings incorrectly in Perl 5.6.0. This has been corrected.

  Support for `tr///CU` and `tr///UC` etc., have been removed since we realized the interface is broken. For similar functionality, see ‘‘pack’’ in perlfunc.

  The Unicode Character Database has been updated to version 3.0.1 with additions made available to the public as of August 30, 2000.

  The Unicode character classes \p{Blank} and \p{SpacePerl} have been added. ‘‘Blank’’ is like C **isblank()**, that is, it contains only ‘‘horizontal whitespace’’ (the space character is, the newline isn’t), and the ‘‘SpacePerl’’ is the Unicode equivalent of `\s` (\p{Space} isn’t, since that includes the vertical tabulator character, whereas `\s` doesn’t.)

  If you are experimenting with Unicode support in perl, the development versions of Perl may have more to offer. In particular, I/O layers are now available in the development track, but not in the maintenance track, primarily to do backward compatibility issues. Unicode support is also evolving rapidly on a daily basis in the development track — the maintenance track only reflects the most conservative of these changes.

64−bit support
: Support for 64−bit platforms has been improved, but continues to be experimental. The level of support varies greatly among platforms.

Compiler
: The B Compiler and its various backends have had many incremental improvements, but they continue to remain highly experimental. Use in production environments is discouraged.

  The perlcc tool has been rewritten so that the user interface is much more like that of a C compiler.

  The perlbc tools has been removed. Use `perlcc -B` instead.

Lvalue subroutines
: There have been various bugfixes to support lvalue subroutines better. However, the feature still remains experimental.

IO::Socket
: IO::Socket::INET failed to open the specified port if the service name was not known. It now correctly uses the supplied port number as is.

File::Find
: File::Find now **chdir()**s correctly when chasing symbolic links.

xsubpp
: xsubpp now tolerates embedded POD sections.

`no Module;`
: `no Module;` does not produce an error even if Module does not have an **unimport()** method. This parallels the behavior of `use` vis-a-vis `import`.

Tests
: A large number of tests have been added.

### Core features

**untie()** will now call an **UNTIE()** hook if it exists. See perltie for details.

The `-DT` command line switch outputs copious tokenizing information. See perlrun.

Arrays are now always interpolated in double-quotish strings. Previously, `"foo@bar.com"` used to be a fatal error at compile time, if an array `@bar` was not used or declared. This transitional behavior was intended to help migrate perl4 code, and is deemed to be no longer useful. See ''Arrays now always interpolate into double-quoted strings''.

**keys()**, **each()**, **pop()**, **push()**, **shift()**, **splice()** and **unshift()** can all be overridden now.

`my __PACKAGE__ $obj` now does the expected thing.

### Configuration issues

On some systems (IRIX and Solaris among them) the system malloc is demonstrably better. While the defaults haven't been changed in order to retain binary compatibility with earlier releases, you may be better off building perl with `Configure -Uusemymalloc ...` as discussed in the *INSTALL* file.

`Configure` has been enhanced in various ways:

- Minimizes use of temporary files.
- By default, does not link perl with libraries not used by it, such as the various dbm libraries. SunOS 4.x hints preserve behavior on that platform.
- Support for pdp11-style memory models has been removed due to obsolescence.
- Building outside the source tree is supported on systems that have symbolic links. This is done by running

  ```
  sh /path/to/source/Configure -Dmksymlinks ...
  make all test install
  ```

  in a directory other than the perl source directory. See *INSTALL*.
- `Configure -S` can be run non-interactively.

### Documentation

README.aix, README.solaris and README.macos have been added. README.posix-bc has been renamed to README.bs2000. These are installed as perlaix, perlsolaris, perlmacos, and perlbs2000 respectively.

The following pod documents are brand new:

```
perlclib    Internal replacements for standard C library functions
perldebtut  Perl debugging tutorial
perlebcdic  Considerations for running Perl on EBCDIC platforms
perlnewmod  Perl modules: preparing a new module for distribution
perlrequick Perl regular expressions quick start
perlretut   Perl regular expressions tutorial
perlutil    utilities packaged with the Perl distribution
```

The *INSTALL* file has been expanded to cover various issues, such as 64-bit support.

A longer list of contributors has been added to the source distribution. See the file `AUTHORS`.

Numerous other changes have been made to the included documentation and FAQs.

### Bundled modules

The following modules have been added.

B::Concise
: Walks Perl syntax tree, printing concise info about ops. See B::Concise.

File::Temp
Returns name and handle of a temporary file safely. See File::Temp.

Pod::LaTeX
Converts Pod data to formatted LaTeX. See Pod::LaTeX.

Pod::Text::Overstrike
Converts POD data to formatted overstrike text. See Pod::Text::Overstrike.

The following modules have been upgraded.

CGI
CGI v2.752 is now included.

CPAN
CPAN v1.59_54 is now included.

Class::Struct
Various bugfixes have been added.

DB_File
DB_File v1.75 supports newer Berkeley DB versions, among other improvements.

Devel::Peek
Devel::Peek has been enhanced to support dumping of memory statistics, when perl is built with the included **malloc()**.

File::Find
File::Find now supports pre and post-processing of the files in order to **sort()** them, etc.

Getopt::Long
Getopt::Long v2.25 is included.

IO::Poll
Various bug fixes have been included.

IPC::Open3
IPC::Open3 allows use of numeric file descriptors.

Math::BigFloat
The **fmod()** function supports modulus operations. Various bug fixes have also been included.

Math::Complex
Math::Complex handles inf, NaN etc., better.

Net::Ping
**ping()** could fail on odd number of data bytes, and when the echo service isn't running. This has been corrected.

Opcode
A memory leak has been fixed.

Pod::Parser
Version 1.13 of the Pod::Parser suite is included.

Pod::Text
Pod::Text and related modules have been upgraded to the versions in podlators suite v2.08.

SDBM_File
On dosish platforms, some keys went missing because of lack of support for files with ''holes''. A workaround for the problem has been added.

Sys::Syslog
Various bug fixes have been included.

Tie::RefHash
Now supports Tie::RefHash::Nestable to automagically tie hashref values.

Tie::SubstrHash
Various bug fixes have been included.

### Platform-specific improvements

The following new ports are now available.

NCR MP-RAS
NonStop-UX

Perl now builds under Amdahl UTS.

Perl has also been verified to build under Amiga OS.

Support for EPOC has been much improved. See README.epoc.

Building perl with −Duseithreads or −Duse5005threads now works under HP-UX 10.20 (previously it only worked under 10.30 or later). You will need a thread library package installed. See README.hpux.

Long doubles should now work under Linux.

Mac OS Classic is now supported in the mainstream source package. See README.macos.

Support for MPE/iX has been updated. See README.mpeix.

Support for OS/2 has been improved. See `os2/Changes` and README.os2.

Dynamic loading on z/OS (formerly OS/390) has been improved. See README.os390.

Support for VMS has seen many incremental improvements, including better support for operators like backticks and **system()**, and better `%ENV` handling. See `README.vms` and perlvms.

Support for Stratus VOS has been improved. See `vos/Changes` and README.vos.

Support for Windows has been improved.

- **fork()** emulation has been improved in various ways, but still continues to be experimental. See perlfork for known bugs and caveats.
- `%SIG` has been enabled under USE_ITHREADS, but its use is completely unsupported under all configurations.
- Borland C++ v5.5 is now a supported compiler that can build Perl. However, the generated binaries continue to be incompatible with those generated by the other supported compilers (GCC and Visual C++).
- Non-blocking waits for child processes (or pseudo-processes) are supported via `waitpid($pid, &POSIX::WNOHANG)`.
- A memory leak in **accept()** has been fixed.
- **wait()**, **waitpid()** and backticks now return the correct exit status under Windows 9x.
- Trailing new `%ENV` entries weren't propagated to child processes. This is now fixed.
- Current directory entries in `%ENV` are now correctly propagated to child processes.
- Duping socket handles with open(F, ''>&MYSOCK'') now works under Windows 9x.
- The makefiles now provide a single switch to bulk-enable all the features enabled in ActiveState ActivePerl (a popular binary distribution).
- **Win32::GetCwd()** correctly returns C:\ instead of C: when at the drive root. Other bugs in **chdir()** and **Cwd::cwd()** have also been fixed.
- **fork()** correctly returns undef and sets EAGAIN when it runs out of pseudo-process handles.
- ExtUtils::MakeMaker now uses `$ENV{LIB}` to search for libraries.
- UNC path handling is better when perl is built to support **fork()**.
- A handle leak in socket handling has been fixed.
- **send()** works from within a pseudo-process.

Unless specifically qualified otherwise, the remainder of this document covers changes between the 5.005 and 5.6.0 releases.

## Core Enhancements

### Interpreter cloning, threads, and concurrency

Perl 5.6.0 introduces the beginnings of support for running multiple interpreters concurrently in different threads. In conjunction with the **perl_clone()** API call, which can be used to selectively duplicate the state of any given interpreter, it is possible to compile a piece of code once in an interpreter, clone that interpreter one or more times, and run all the resulting interpreters in distinct threads.

On the Windows platform, this feature is used to emulate **fork()** at the interpreter level. See perlfork for details about that.

This feature is still in evolution. It is eventually meant to be used to selectively clone a subroutine and data reachable from that subroutine in a separate interpreter and run the cloned subroutine in a separate thread. Since there is no shared data between the interpreters, little or no locking will be needed (unless parts of the symbol table are explicitly shared). This is obviously intended to be an easy-to-use replacement for the existing threads support.

Support for cloning interpreters and interpreter concurrency can be enabled using the −Dusethreads Configure option (see win32/Makefile for how to enable it on Windows.) The resulting perl executable will be functionally identical to one that was built with −Dmultiplicity, but the **perl_clone()** API call will only be available in the former.

−Dusethreads enables the cpp macro USE_ITHREADS by default, which in turn enables Perl source code changes that provide a clear separation between the op tree and the data it operates with. The former is immutable, and can therefore be shared between an interpreter and all of its clones, while the latter is considered local to each interpreter, and is therefore copied for each clone.

Note that building Perl with the −Dusemultiplicity Configure option is adequate if you wish to run multiple **independent** interpreters concurrently in different threads. −Dusethreads only provides the additional functionality of the **perl_clone()** API call and other support for running **cloned** interpreters concurrently.

```
NOTE: This is an experimental feature.  Implementation details are
subject to change.
```

### Lexically scoped warning categories

You can now control the granularity of warnings emitted by perl at a finer level using the `use warnings` pragma. warnings and perllexwarn have copious documentation on this feature.

### Unicode and UTF−8 support

Perl now uses UTF−8 as its internal representation for character strings. The `utf8` and `bytes` pragmas are used to control this support in the current lexical scope. See perlunicode, utf8 and bytes for more information.

This feature is expected to evolve quickly to support some form of I/O disciplines that can be used to specify the kind of input and output data (bytes or characters). Until that happens, additional modules from CPAN will be needed to complete the toolkit for dealing with Unicode.

```
NOTE: This should be considered an experimental feature.  Implementation
details are subject to change.
```

### Support for interpolating named characters

The new `\N` escape interpolates named characters within strings. For example, `"Hi! \N{WHITE SMILING FACE}"` evaluates to a string with a Unicode smiley face at the end.

### “our” declarations

An ‘‘our’’ declaration introduces a value that can be best understood as a lexically scoped symbolic alias to a global variable in the package that was current where the variable was declared. This is mostly useful as an alternative to the `vars` pragma, but also provides the opportunity to introduce typing and other attributes for such variables. See ‘‘our’’ in perlfunc.

### Support for strings represented as a vector of ordinals

Literals of the form `v1.2.3.4` are now parsed as a string composed of characters with the specified ordinals. This is an alternative, more readable way to construct (possibly Unicode) strings instead of interpolating characters, as in `"\x{1}\x{2}\x{3}\x{4}"`. The leading `v` may be omitted if there are more than two ordinals, so `1.2.3` is parsed the same as `v1.2.3`.

Strings written in this form are also useful to represent version ''numbers''. It is easy to compare such version ''numbers'' (which are really just plain strings) using any of the usual string comparison operators `eq`, `ne`, `lt`, `gt`, etc., or perform bitwise string operations on them using `|`, `&`, etc.

In conjunction with the new `$^V` magic variable (which contains the perl version as a string), such literals can be used as a readable way to check if you're running a particular version of Perl:

```
# this will parse in older versions of Perl also
if ($^V and $^V gt v5.6.0) {
    # new features supported
}
```

`require` and `use` also have some special magic to support such literals. They will be interpreted as a version rather than as a module name:

```
require v5.6.0;             # croak if $^V lt v5.6.0
use v5.6.0;                 # same, but croaks at compile-time
```

Alternatively, the `v` may be omitted if there is more than one dot:

```
require 5.6.0;
use 5.6.0;
```

Also, `sprintf` and `printf` support the Perl-specific format flag `%v` to print ordinals of characters in arbitrary strings:

```
printf "v%vd", $^V;         # prints current version, such as "v5.5.650"
printf "%*vX", ":", $addr;  # formats IPv6 address
printf "%*vb", " ", $bits;  # displays bitstring
```

See ''Scalar value constructors'' in perldata for additional information.

### Improved Perl version numbering system

Beginning with Perl version 5.6.0, the version number convention has been changed to a ''dotted integer'' scheme that is more commonly found in open source projects.

Maintenance versions of v5.6.0 will be released as v5.6.1, v5.6.2 etc. The next development series following v5.6.0 will be numbered v5.7.x, beginning with v5.7.0, and the next major production release following v5.6.0 will be v5.8.0.

The English module now sets `$PERL_VERSION` to $^V (a string value) rather than `$]` (a numeric value). (This is a potential incompatibility. Send us a report via perlbug if you are affected by this.)

The v1.2.3 syntax is also now legal in Perl. See ''Support for strings represented as a vector of ordinals'' for more on that.

To cope with the new versioning system's use of at least three significant digits for each version component, the method used for incrementing the subversion number has also changed slightly. We assume that versions older than v5.6.0 have been incrementing the subversion component in multiples of 10. Versions after v5.6.0 will increment them by 1. Thus, using the new notation, 5.005_03 is the ''same'' as v5.5.30, and the first maintenance version following v5.6.0 will be v5.6.1 (which should be read as being equivalent to a floating point value of 5.006_001 in the older format, stored in `$]`).

### New syntax for declaring subroutine attributes

Formerly, if you wanted to mark a subroutine as being a method call or as requiring an automatic **lock()** when it is entered, you had to declare that with a `use attrs` pragma in the body of the subroutine. That can now be accomplished with declaration syntax, like this:

```
sub mymethod : locked method;
...
sub mymethod : locked method {
    ...
}

sub othermethod :locked :method;
...
sub othermethod :locked :method {
    ...
```

```
}
```

(Note how only the first `:` is mandatory, and whitespace surrounding the `:` is optional.)

*AutoSplit.pm* and *SelfLoader.pm* have been updated to keep the attributes with the stubs they provide. See attributes.

### File and directory handles can be autovivified

Similar to how constructs such as `$x->[0]` autovivify a reference, handle constructors (**open()**, **opendir()**, **pipe()**, **socketpair()**, **sysopen()**, **socket()**, and **accept()**) now autovivify a file or directory handle if the handle passed to them is an uninitialized scalar variable. This allows the constructs such as `open(my $fh, ...)` and `open(local $fh,...)` to be used to create filehandles that will conveniently be closed automatically when the scope ends, provided there are no other references to them. This largely eliminates the need for typeglobs when opening filehandles that must be passed around, as in the following example:

```
sub myopen {
    open my $fh, "@_"
         or die "Can't open '@_': $!";
    return $fh;
}

{
    my $f = myopen("</etc/motd");
    print <$f>;
    # $f implicitly closed here
}
```

### open() with more than two arguments

If **open()** is passed three arguments instead of two, the second argument is used as the mode and the third argument is taken to be the file name. This is primarily useful for protecting against unintended magic behavior of the traditional two-argument form. See ‘‘open’’ in perlfunc.

### 64−bit support

Any platform that has 64−bit integers either

```
(1) natively as longs or ints
(2) via special compiler flags
(3) using long long or int64_t
```

is able to use ‘‘quads’’ (64−bit integers) as follows:

- constants (decimal, hexadecimal, octal, binary) in the code
- arguments to **oct()** and **hex()**
- arguments to **print()**, **printf()** and **sprintf()** (flag prefixes ll, L, q)
- printed as such
- **pack()** and **unpack()** ‘‘q’’ and ‘‘Q’’ formats
- in basic arithmetics: + − * / % (NOTE: operating close to the limits of the integer values may produce surprising results)
- in bit arithmetics: & | ^ ~ << >> (NOTE: these used to be forced to be 32 bits wide but now operate on the full native width.)
- **vec()**

Note that unless you have the case (a) you will have to configure and compile Perl using the −Duse64bitint Configure flag.

```
NOTE: The Configure flags -Duselonglong and -Duse64bits have been
deprecated.  Use -Duse64bitint instead.
```

There are actually two modes of 64−bitness: the first one is achieved using Configure −Duse64bitint and the second one using Configure −Duse64bitall. The difference is that the first one is minimal and the second one maximal. The first works in more places than the second.

The `use64bitint` does only as much as is required to get 64−bit integers into Perl (this may mean,

for example, using ''long longs'') while your memory may still be limited to 2 gigabytes (because your pointers could still be 32−bit). Note that the name `64bitint` does not imply that your C compiler will be using 64−bit `int`s (it might, but it doesn't have to): the `use64bitint` means that you will be able to have 64 bits wide scalar values.

The `use64bitall` goes all the way by attempting to switch also integers (if it can), longs (and pointers) to being 64−bit. This may create an even more binary incompatible Perl than −Duse64bitint: the resulting executable may not run at all in a 32−bit box, or you may have to reboot/reconfigure/rebuild your operating system to be 64−bit aware.

Natively 64−bit systems like Alpha and Cray need neither −Duse64bitint nor −Duse64bitall.

Last but not least: note that due to Perl's habit of always using floating point numbers, the quads are still not true integers. When quads overflow their limits (0...18_446_744_073_709_551_615 unsigned, −9_223_372_036_854_775_808...9_223_372_036_854_775_807 signed), they are silently promoted to floating point numbers, after which they will start losing precision (in their lower digits).

```
NOTE: 64−bit support is still experimental on most platforms.
Existing support only covers the LP64 data model.  In particular, the
LLP64 data model is not yet supported.  64−bit libraries and system
APIs on many platforms have not stabilized−−your mileage may vary.
```

### Large file support

If you have filesystems that support ''large files'' (files larger than 2 gigabytes), you may now also be able to create and access them from Perl.

```
NOTE: The default action is to enable large file support, if
available on the platform.
```

If the large file support is on, and you have a Fcntl constant O_LARGEFILE, the O_LARGEFILE is automatically added to the flags of **sysopen()**.

Beware that unless your filesystem also supports ''sparse files'' seeking to umpteen petabytes may be inadvisable.

Note that in addition to requiring a proper file system to do large files you may also need to adjust your per-process (or your per-system, or per-process-group, or per-user-group) maximum filesize limits before running Perl scripts that try to handle large files, especially if you intend to write such files.

Finally, in addition to your process/process group maximum filesize limits, you may have quota limits on your filesystems that stop you (your user id or your user group id) from using large files.

Adjusting your process/user/group/file system/operating system limits is outside the scope of Perl core language. For process limits, you may try increasing the limits using your shell's limits/limit/ulimit command before running Perl. The BSD::Resource extension (not included with the standard Perl distribution) may also be of use, it offers the getrlimit/setrlimit interface that can be used to adjust process resource usage limits, including the maximum filesize limit.

### Long doubles

In some systems you may be able to use long doubles to enhance the range and precision of your double precision floating point numbers (that is, Perl's numbers). Use Configure −Duselongdouble to enable this support (if it is available).

### "more bits"

You can ''Configure −Dusemorebits'' to turn on both the 64−bit support and the long double support.

### Enhanced support for sort() subroutines

Perl subroutines with a prototype of `($$)`, and XSUBs in general, can now be used as sort subroutines. In either case, the two elements to be compared are passed as normal parameters in `@_`. See ''sort'' in perlfunc.

For unprototyped sort subroutines, the historical behavior of passing the elements to be compared as the global variables `$a` and `$b` remains unchanged.

### `sort $coderef @foo` allowed

**sort()** did not accept a subroutine reference as the comparison function in earlier versions. This is now permitted.

### File globbing implemented internally

Perl now uses the File::Glob implementation of the **glob()** operator automatically. This avoids using an external csh process and the problems associated with it.

```
NOTE: This is currently an experimental feature.  Interfaces and
implementation are subject to change.
```

### Support for CHECK blocks

In addition to `BEGIN`, `INIT`, `END`, `DESTROY` and `AUTOLOAD`, subroutines named `CHECK` are now special. These are queued up during compilation and behave similar to END blocks, except they are called at the end of compilation rather than at the end of execution. They cannot be called directly.

### POSIX character class syntax [: :] supported

For example to match alphabetic characters use /[[:alpha:]]/. See perlre for details.

### Better pseudo-random number generator

In 5.005_0x and earlier, perl's **rand()** function used the C library **rand** (3) function. As of 5.005_52, Configure tests for **drand48()**, **random()**, and **rand()** (in that order) and picks the first one it finds.

These changes should result in better random numbers from **rand()**.

### Improved `qw//` operator

The `qw//` operator is now evaluated at compile time into a true list instead of being replaced with a run time call to `split()`. This removes the confusing misbehaviour of `qw//` in scalar context, which had inherited that behaviour from **split()**.

Thus:

```
$foo = ($bar) = qw(a b c); print "$foo|$bar\n";
```

now correctly prints ‘‘3|a’’, instead of ‘‘2|a’’.

### Better worst-case behavior of hashes

Small changes in the hashing algorithm have been implemented in order to improve the distribution of lower order bits in the hashed value. This is expected to yield better performance on keys that are repeated sequences.

### pack() format 'Z' supported

The new format type 'Z' is useful for packing and unpacking null-terminated strings. See ‘‘pack’’ in perlfunc.

### pack() format modifier '!' supported

The new format type modifier '!' is useful for packing and unpacking native shorts, ints, and longs. See ‘‘pack’’ in perlfunc.

### pack() and unpack() support counted strings

The template character '/' can be used to specify a counted string type to be packed or unpacked. See ‘‘pack’’ in perlfunc.

### Comments in pack() templates

The '#' character in a template introduces a comment up to end of the line. This facilitates documentation of **pack()** templates.

### Weak references

In previous versions of Perl, you couldn't cache objects so as to allow them to be deleted if the last reference from outside the cache is deleted. The reference in the cache would hold a reference count on the object and the objects would never be destroyed.

Another familiar problem is with circular references. When an object references itself, its reference count would never go down to zero, and it would not get destroyed until the program is about to exit.

Weak references solve this by allowing you to ‘‘weaken’’ any reference, that is, make it not count towards the reference count. When the last non-weak reference to an object is deleted, the object is destroyed and all the weak references to the object are automatically undef-ed.

To use this feature, you need the Devel::WeakRef package from CPAN, which contains additional documentation.

```
NOTE: This is an experimental feature.  Details are subject to change.
```

### Binary numbers supported

Binary numbers are now supported as literals, in s?printf formats, and `oct()`:

```
$answer = 0b101010;
printf "The answer is: %b\n", oct("0b101010");
```

### Lvalue subroutines

Subroutines can now return modifiable lvalues. See ‘‘Lvalue subroutines’’ in perlsub.

```
NOTE: This is an experimental feature.  Details are subject to change.
```

### Some arrows may be omitted in calls through references

Perl now allows the arrow to be omitted in many constructs involving subroutine calls through references. For example, `$foo[10]->('foo')` may now be written `$foo[10]('foo')`. This is rather similar to how the arrow may be omitted from `$foo[10]->{'foo'}`. Note however, that the arrow is still required for `foo(10)->('bar')`.

### Boolean assignment operators are legal lvalues

Constructs such as `($a ||= 2) += 1` are now allowed.

### exists() is supported on subroutine names

The **exists()** builtin now works on subroutine names. A subroutine is considered to exist if it has been declared (even if implicitly). See ‘‘exists’’ in perlfunc for examples.

### exists() and delete() are supported on array elements

The **exists()** and **delete()** builtins now work on simple arrays as well. The behavior is similar to that on hash elements.

**exists()** can be used to check whether an array element has been initialized. This avoids autovivifying array elements that don’t exist. If the array is tied, the **EXISTS()** method in the corresponding tied package will be invoked.

**delete()** may be used to remove an element from the array and return it. The array element at that position returns to its uninitialized state, so that testing for the same element with **exists()** will return false. If the element happens to be the one at the end, the size of the array also shrinks up to the highest element that tests true for **exists()**, or 0 if none such is found. If the array is tied, the **DELETE()** method in the corresponding tied package will be invoked.

See ‘‘exists’’ in perlfunc and ‘‘delete’’ in perlfunc for examples.

### Pseudo-hashes work better

Dereferencing some types of reference values in a pseudo-hash, such as `$ph->{foo}[1]`, was accidentally disallowed. This has been corrected.

When applied to a pseudo-hash element, **exists()** now reports whether the specified value exists, not merely if the key is valid.

**delete()** now works on pseudo-hashes. When given a pseudo-hash element or slice it deletes the values corresponding to the keys (but not the keys themselves). See ‘‘Pseudo-hashes: Using an array as a hash’’ in perlref.

Pseudo-hash slices with constant keys are now optimized to array lookups at compile-time.

List assignments to pseudo-hash slices are now supported.

The `fields` pragma now provides ways to create pseudo-hashes, via **fields::new()** and **fields::phash()**. See fields.

```
NOTE: The pseudo-hash data type continues to be experimental.
Limiting oneself to the interface elements provided by the
fields pragma will provide protection from any future changes.
```

### Automatic flushing of output buffers

**fork()**, **exec()**, **system()**, qx//, and pipe **open()**s now flush buffers of all files opened for output when the operation was attempted. This mostly eliminates confusing buffering mishaps suffered by users unaware of how Perl internally handles I/O.

This is not supported on some platforms like Solaris where a suitably correct implementation of fflush(NULL) isn’t available.

### Better diagnostics on meaningless filehandle operations

Constructs such as `open(<FH>)` and `close(<FH>)` are compile time errors. Attempting to read from filehandles that were opened only for writing will now produce warnings (just as writing to read-only filehandles does).

### Where possible, buffered data discarded from duped input filehandle

`open(NEW, "<&OLD")` now attempts to discard any data that was previously read and buffered in `OLD` before duping the handle. On platforms where doing this is allowed, the next read operation on `NEW` will return the same data as the corresponding operation on `OLD`. Formerly, it would have returned the data from the start of the following disk block instead.

### eof() has the same old magic as <>

`eof()` would return true if no attempt to read from `<>` had yet been made. `eof()` has been changed to have a little magic of its own, it now opens the `<>` files.

### binmode() can be used to set :crlf and :raw modes

**binmode()** now accepts a second argument that specifies a discipline for the handle in question. The two pseudo-disciplines ''`:raw`'' and ''`:crlf`'' are currently supported on DOS-derivative platforms. See ''binmode'' in perlfunc and open.

### `-T` filetest recognizes UTF-8 encoded files as "text"

The algorithm used for the `-T` filetest has been enhanced to correctly identify UTF-8 content as ''text''.

### system(), backticks and pipe open now reflect exec() failure

On Unix and similar platforms, **system()**, **qx()** and open(FOO, ''cmd |'') etc., are implemented via **fork()** and **exec()**. When the underlying **exec()** fails, earlier versions did not report the error properly, since the **exec()** happened to be in a different process.

The child process now communicates with the parent about the error in launching the external command, which allows these constructs to return with their usual error value and set $!.

### Improved diagnostics

Line numbers are no longer suppressed (under most likely circumstances) during the global destruction phase.

Diagnostics emitted from code running in threads other than the main thread are now accompanied by the thread ID.

Embedded null characters in diagnostics now actually show up. They used to truncate the message in prior versions.

`$foo::a` and `$foo::b` are now exempt from ''possible typo'' warnings only if **sort()** is encountered in package `foo`.

Unrecognized alphabetic escapes encountered when parsing quote constructs now generate a warning, since they may take on new semantics in later versions of Perl.

Many diagnostics now report the internal operation in which the warning was provoked, like so:

```
Use of uninitialized value in concatenation (.) at (eval 1) line 1.
Use of uninitialized value in print at (eval 1) line 1.
```

Diagnostics that occur within eval may also report the file and line number where the eval is located, in addition to the eval sequence number and the line number within the evaluated text itself. For example:

```
Not enough arguments for scalar at (eval 4)[newlib/perl5db.pl:1411] line 2, a
```

### Diagnostics follow STDERR

Diagnostic output now goes to whichever file the `STDERR` handle is pointing at, instead of always going to the underlying C runtime library's `stderr`.

### More consistent close-on-exec behavior

On systems that support a close-on-exec flag on filehandles, the flag is now set for any handles created by **pipe()**, **socketpair()**, **socket()**, and **accept()**, if that is warranted by the value of $^F that may be in effect. Earlier versions neglected to set the flag for handles created with these operators. See ''pipe'' in perlfunc, ''socketpair'' in perlfunc, ''socket'' in perlfunc, ''accept'' in perlfunc, and ''$^F'' in perlvar.

### syswrite() ease-of-use

The length argument of `syswrite()` has become optional.

### Better syntax checks on parenthesized unary operators

Expressions such as:

```
print defined(&foo,&bar,&baz);
print uc("foo","bar","baz");
undef($foo,&bar);
```

used to be accidentally allowed in earlier versions, and produced unpredictable behaviour. Some produced ancillary warnings when used in this way; others silently did the wrong thing.

The parenthesized forms of most unary operators that expect a single argument now ensure that they are not called with more than one argument, making the cases shown above syntax errors. The usual behaviour of:

```
print defined &foo, &bar, &baz;
print uc "foo", "bar", "baz";
undef $foo, &bar;
```

remains unchanged. See perlop.

### Bit operators support full native integer width

The bit operators (& | ^ ~ << >>) now operate on the full native integral width (the exact size of which is available in `$Config{ivsize}`). For example, if your platform is either natively 64−bit or if Perl has been configured to use 64−bit integers, these operations apply to 8 bytes (as opposed to 4 bytes on 32−bit platforms). For portability, be sure to mask off the excess bits in the result of unary `~`, e.g., `~$x & 0xffffffff`.

### Improved security features

More potentially unsafe operations taint their results for improved security.

The `passwd` and `shell` fields returned by the **getpwent()**, **getpwnam()**, and **getpwuid()** are now tainted, because the user can affect their own encrypted password and login shell.

The variable modified by **shmread()**, and messages returned by **msgrcv()** (and its object-oriented interface IPC::SysV::Msg::rcv) are also tainted, because other untrusted processes can modify messages and shared memory segments for their own nefarious purposes.

### More functional bareword prototype (*)

Bareword prototypes have been rationalized to enable them to be used to override builtins that accept barewords and interpret them in a special way, such as `require` or `do`.

Arguments prototyped as `*` will now be visible within the subroutine as either a simple scalar or as a reference to a typeglob. See ‘‘Prototypes’’ in perlsub.

### `require` and `do` may be overridden

`require` and `do 'file'` operations may be overridden locally by importing subroutines of the same name into the current package (or globally by importing them into the CORE::GLOBAL:: namespace). Overriding `require` will also affect `use`, provided the override is visible at compile-time. See ‘‘Overriding Built-in Functions’’ in perlsub.

### $^X variables may now have names longer than one character

Formerly, $^X was synonymous with ${‘‘\cX’’}, but $^XY was a syntax error. Now variable names that begin with a control character may be arbitrarily long. However, for compatibility reasons, these variables *must* be written with explicit braces, as `${^XY}` for example. `${^XYZ}` is synonymous with ${‘‘\cXYZ’’}. Variable names with more than one control character, such as `${^XY^Z}`, are illegal.

The old syntax has not changed. As before, ‘^X’ may be either a literal control-X character or the two-character sequence ‘caret’ plus ‘X’. When braces are omitted, the variable name stops after the control character. Thus `"$^XYZ"` continues to be synonymous with `$^X . "YZ"` as before.

As before, lexical variables may not have names beginning with control characters. As before, variables whose names begin with a control character are always forced to be in package ‘main’. All such variables are reserved for future extensions, except those that begin with `^_`, which may be used by user programs and are guaranteed not to acquire special meaning in any future version of Perl.

### New variable $^C reflects -c switch

`$^C` has a boolean value that reflects whether perl is being run in compile-only mode (i.e. via the `-c` switch). Since BEGIN blocks are executed under such conditions, this variable enables perl code to determine whether actions that make sense only during normal running are warranted. See perlvar.

### New variable $^V contains Perl version as a string

`$^V` contains the Perl version number as a string composed of characters whose ordinals match the version numbers, i.e. v5.6.0. This may be used in string comparisons.

See `Support for strings represented as a vector of ordinals` for an example.

### Optional Y2K warnings

If Perl is built with the cpp macro `PERL_Y2KWARN` defined, it emits optional warnings when concatenating the number 19 with another number.

This behavior must be specifically enabled when running Configure. See *INSTALL* and *README.Y2K*.

### Arrays now always interpolate into double-quoted strings

In double-quoted strings, arrays now interpolate, no matter what. The behavior in earlier versions of perl 5 was that arrays would interpolate into strings if the array had been mentioned before the string was compiled, and otherwise Perl would raise a fatal compile-time error. In versions 5.000 through 5.003, the error was

```
Literal @example now requires backslash
```

In versions 5.004_01 through 5.6.0, the error was

```
In string, @example now must be written as \@example
```

The idea here was to get people into the habit of writing `"fred\@example.com"` when they wanted a literal `@` sign, just as they have always written `"Give me back my \$5"` when they wanted a literal `$` sign.

Starting with 5.6.1, when Perl now sees an `@` sign in a double-quoted string, it *always* attempts to interpolate an array, regardless of whether or not the array has been used or declared already. The fatal error has been downgraded to an optional warning:

```
Possible unintended interpolation of @example in string
```

This warns you that `"fred@example.com"` is going to turn into `fred.com` if you don't backslash the `@`. See http://perl.plover.com/at-error.html for more details about the history here.

### @- and @+ provide starting/ending offsets of regex submatches

The new magic variables @- and @+ provide the starting and ending offsets, respectively, of $&, `$1`, `$2`, etc. See perlvar for details.

## Modules and Pragmata

### Modules

attributes

While used internally by Perl as a pragma, this module also provides a way to fetch subroutine and variable attributes. See attributes.

B The Perl Compiler suite has been extensively reworked for this release. More of the standard Perl test suite passes when run under the Compiler, but there is still a significant way to go to achieve production quality compiled executables.

```
NOTE: The Compiler suite remains highly experimental.  The
generated code may not be correct, even when it manages to execute
without errors.
```

Benchmark

Overall, Benchmark results exhibit lower average error and better timing accuracy.

You can now run tests for *n* seconds instead of guessing the right number of tests to run: e.g., timethese(-5, ...) will run each code for at least 5 CPU seconds. Zero as the "number of repetitions" means "for at least 3 CPU seconds". The output format has also changed. For example:

```
use Benchmark;$x=3;timethese(-5,{a=>sub{$x*$x},b=>sub{$x**2}})
```

will now output something like this:

```
Benchmark: running a, b, each for at least 5 CPU seconds...
         a:  5 wallclock secs ( 5.77 usr +  0.00 sys =  5.77 CPU) @ 200551.
         b:  4 wallclock secs ( 5.00 usr +  0.02 sys =  5.02 CPU) @ 159605.
```

New features: ‘‘each for at least N CPU seconds...’’, ‘‘wallclock secs’’, and the ‘‘@ operations/CPU second (n=operations)’’.

**timethese()** now returns a reference to a hash of Benchmark objects containing the test results, keyed on the names of the tests.

**timethis()** now returns the iterations field in the Benchmark result object instead of 0.

**timethese()**, **timethis()**, and the new **cmpthese()** (see below) can also take a format specifier of ’none’ to suppress output.

A new function **countit()** is just like **timeit()** except that it takes a TIME instead of a COUNT.

A new function **cmpthese()** prints a chart comparing the results of each test returned from a **timethese()** call. For each possible pair of tests, the percentage speed difference (iters/sec or seconds/iter) is shown.

For other details, see Benchmark.

ByteLoader
The ByteLoader is a dedicated extension to generate and run Perl bytecode. See ByteLoader.

constant
References can now be used.

The new version also allows a leading underscore in constant names, but disallows a double leading underscore (as in ‘‘__LINE__’’). Some other names are disallowed or warned against, including BEGIN, END, etc. Some names which were forced into main:: used to fail silently in some cases; now they’re fatal (outside of main::) and an optional warning (inside of main::). The ability to detect whether a constant had been set with a given name has been added.

See constant.

charnames
This pragma implements the `\N` string escape. See charnames.

Data::Dumper
A `Maxdepth` setting can be specified to avoid venturing too deeply into deep data structures. See Data::Dumper.

The XSUB implementation of **Dump()** is now automatically called if the `Useqq` setting is not in use.

Dumping `qr//` objects works correctly.

DB `DB` is an experimental module that exposes a clean abstraction to Perl’s debugging API.

DB_File
DB_File can now be built with Berkeley DB versions 1, 2 or 3. See `ext/DB_File/Changes`.

Devel::DProf
Devel::DProf, a Perl source code profiler has been added. See Devel::DProf and dprofpp.

Devel::Peek
The Devel::Peek module provides access to the internal representation of Perl variables and data. It is a data debugging tool for the XS programmer.

Dumpvalue
The Dumpvalue module provides screen dumps of Perl data.

DynaLoader
DynaLoader now supports a **dl_unload_file()** function on platforms that support unloading shared objects using **dlclose()**.

Perl can also optionally arrange to unload all extension shared objects loaded by Perl. To enable

this, build Perl with the Configure option `-Accflags=-DDL_UNLOAD_ALL_AT_EXIT`. (This maybe useful if you are using Apache with mod_perl.)

English
: `$PERL_VERSION` now stands for `$^V` (a string value) rather than for `$]` (a numeric value).

Env
: Env now supports accessing environment variables like PATH as array variables.

Fcntl
: More Fcntl constants added: F_SETLK64, F_SETLKW64, O_LARGEFILE for large file (more than 4GB) access (NOTE: the O_LARGEFILE is automatically added to **sysopen()** flags if large file support has been configured, as is the default), Free/Net/OpenBSD locking behaviour flags F_FLOCK, F_POSIX, Linux F_SHLCK, and O_ACCMODE: the combined mask of O_RDONLY, O_WRONLY, and O_RDWR. The **seek()**/**sysseek()** constants SEEK_SET, SEEK_CUR, and SEEK_END are available via the `:seek` tag. The **chmod()**/**stat()** S_IF* constants and S_IS* functions are available via the `:mode` tag.

File::Compare
: A **compare_text()** function has been added, which allows custom comparison functions. See File::Compare.

File::Find
: File::Find now works correctly when the **wanted()** function is either autoloaded or is a symbolic reference.

  A bug that caused File::Find to lose track of the working directory when pruning top-level directories has been fixed.

  File::Find now also supports several other options to control its behavior. It can follow symbolic links if the `follow` option is specified. Enabling the `no_chdir` option will make File::Find skip changing the current directory when walking directories. The `untaint` flag can be useful when running with taint checks enabled.

  See File::Find.

File::Glob
: This extension implements BSD-style file globbing. By default, it will also be used for the internal implementation of the **glob()** operator. See File::Glob.

File::Spec
: New methods have been added to the File::Spec module: **devnull()** returns the name of the null device (/dev/null on Unix) and **tmpdir()** the name of the temp directory (normally /tmp on Unix). There are now also methods to convert between absolute and relative filenames: **abs2rel()** and **rel2abs()**. For compatibility with operating systems that specify volume names in file paths, the **splitpath()**, **splitdir()**, and **catdir()** methods have been added.

File::Spec::Functions
: The new File::Spec::Functions modules provides a function interface to the File::Spec module. Allows shorthand

```
$fullname = catfile($dir1, $dir2, $file);
```

  instead of

```
$fullname = File::Spec->catfile($dir1, $dir2, $file);
```

Getopt::Long
: Getopt::Long licensing has changed to allow the Perl Artistic License as well as the GPL. It used to be GPL only, which got in the way of non-GPL applications that wanted to use Getopt::Long.

  Getopt::Long encourages the use of Pod::Usage to produce help messages. For example:

```
    use Getopt::Long;
    use Pod::Usage;
    my $man = 0;
    my $help = 0;
    GetOptions('help|?' => \$help, man => \$man) or pod2usage(2);
    pod2usage(1) if $help;
    pod2usage(−exitstatus => 0, −verbose => 2) if $man;

    __END__

    =head1 NAME

    sample − Using Getopt::Long and Pod::Usage

    =head1 SYNOPSIS

    sample [options] [file ...]

     Options:
       −help            brief help message
       −man             full documentation

    =head1 OPTIONS

    =over 8

    =item B<−help>

    Print a brief help message and exits.

    =item B<−man>

    Prints the manual page and exits.

    =back

    =head1 DESCRIPTION

    B<This program> will read the given input file(s) and do something
    useful with the contents thereof.

    =cut
```

See Pod::Usage for details.

A bug that prevented the non-option call-back <> from being specified as the first argument has been fixed.

To specify the characters < and > as option starters, use ><. Note, however, that changing option starters is strongly deprecated.

IO **write()** and **syswrite()** will now accept a single-argument form of the call, for consistency with Perl's **syswrite()**.

You can now create a TCP-based IO::Socket::INET without forcing a connect attempt. This allows you to configure its options (like making it non-blocking) and then call **connect()** manually.

A bug that prevented the **IO::Socket::protocol()** accessor from ever returning the correct value has been corrected.

IO::Socket::connect now uses non-blocking IO instead of **alarm()** to do connect timeouts.

IO::Socket::accept now uses **select()** instead of **alarm()** for doing timeouts.

IO::Socket::INET->new now sets $! correctly on failure. $@ is still set for backwards compatibility.

JPL Java Perl Lingo is now distributed with Perl. See jpl/README for more information.

lib `use lib` now weeds out any trailing duplicate entries. `no lib` removes all named entries.

Math::BigInt
The bitwise operations <<, >>, &, |, and ~ are now supported on bigints.

Math::Complex
The accessor methods Re, Im, arg, abs, rho, and theta can now also act as mutators (accessor `$z->`**Re()**, mutator `$z->`**Re**(3)).

The class method `display_format` and the corresponding object method `display_format`, in addition to accepting just one argument, now can also accept a parameter hash. Recognized keys of a parameter hash are `"style"`, which corresponds to the old one parameter case, and two new parameters: `"format"`, which is a **printf()**-style format string (defaults usually to `"%.15g"`, you can revert to the default by setting the format string to `undef`) used for both parts of a complex number, and `"polar_pretty_print"` (defaults to true), which controls whether an attempt is made to try to recognize small multiples and rationals of pi (2pi, pi/2) at the argument (angle) of a polar complex number.

The potentially disruptive change is that in list context both methods now *return the parameter hash*, instead of only the value of the `"style"` parameter.

Math::Trig
A little bit of radial trigonometry (cylindrical and spherical), radial coordinate conversions, and the great circle distance were added.

Pod::Parser, Pod::InputObjects
Pod::Parser is a base class for parsing and selecting sections of pod documentation from an input stream. This module takes care of identifying pod paragraphs and commands in the input and hands off the parsed paragraphs and commands to user-defined methods which are free to interpret or translate them as they see fit.

Pod::InputObjects defines some input objects needed by Pod::Parser, and for advanced users of Pod::Parser that need more about a command besides its name and text.

As of release 5.6.0 of Perl, Pod::Parser is now the officially sanctioned ''base parser code'' recommended for use by all pod2xxx translators. Pod::Text (pod2text) and Pod::Man (pod2man) have already been converted to use Pod::Parser and efforts to convert Pod::HTML (pod2html) are already underway. For any questions or comments about pod parsing and translating issues and utilities, please use the pod-people@perl.org mailing list.

For further information, please see Pod::Parser and Pod::InputObjects.

Pod::Checker, podchecker
This utility checks pod files for correct syntax, according to perlpod. Obvious errors are flagged as such, while warnings are printed for mistakes that can be handled gracefully. The checklist is not complete yet. See Pod::Checker.

Pod::ParseUtils, Pod::Find
These modules provide a set of gizmos that are useful mainly for pod translators. Pod::Find traverses directory structures and returns found pod files, along with their canonical names (like `File::Spec::Unix`). Pod::ParseUtils contains **Pod::List** (useful for storing pod list information), **Pod::Hyperlink** (for parsing the contents of `L<>` sequences) and **Pod::Cache** (for caching information about pod files, e.g., link nodes).

Pod::Select, podselect
Pod::Select is a subclass of Pod::Parser which provides a function named ''**podselect()**'' to filter out user-specified sections of raw pod documentation from an input stream. podselect is a script that provides access to Pod::Select from other scripts to be used as a filter. See Pod::Select.

Pod::Usage, pod2usage

Pod::Usage provides the function ‘‘**pod2usage()**’’ to print usage messages for a Perl script based on its embedded pod documentation. The **pod2usage()** function is generally useful to all script authors since it lets them write and maintain a single source (the pods) for documentation, thus removing the need to create and maintain redundant usage message text consisting of information already in the pods.

There is also a pod2usage script which can be used from other kinds of scripts to print usage messages from pods (even for non-Perl scripts with pods embedded in comments).

For details and examples, please see Pod::Usage.

Pod::Text and Pod::Man

Pod::Text has been rewritten to use Pod::Parser. While **pod2text()** is still available for backwards compatibility, the module now has a new preferred interface. See Pod::Text for the details. The new Pod::Text module is easily subclassed for tweaks to the output, and two such subclasses (Pod::Text::Termcap for man-page-style bold and underlining using termcap information, and Pod::Text::Color for markup with ANSI color sequences) are now standard.

pod2man has been turned into a module, Pod::Man, which also uses Pod::Parser. In the process, several outstanding bugs related to quotes in section headers, quoting of code escapes, and nested lists have been fixed. pod2man is now a wrapper script around this module.

SDBM_File

An EXISTS method has been added to this module (and **sdbm_exists()** has been added to the underlying sdbm library), so one can now call exists on an SDBM_File tied hash and get the correct result, rather than a runtime error.

A bug that may have caused data loss when more than one disk block happens to be read from the database in a single **FETCH()** has been fixed.

Sys::Syslog

Sys::Syslog now uses XSUBs to access facilities from syslog.h so it no longer requires syslog.ph to exist.

Sys::Hostname

Sys::Hostname now uses XSUBs to call the C library’s **gethostname()** or **uname()** if they exist.

Term::ANSIColor

Term::ANSIColor is a very simple module to provide easy and readable access to the ANSI color and highlighting escape sequences, supported by most ANSI terminal emulators. It is now included standard.

Time::Local

The **timelocal()** and **timegm()** functions used to silently return bogus results when the date fell outside the machine’s integer range. They now consistently **croak()** if the date falls in an unsupported range.

Win32

The error return value in list context has been changed for all functions that return a list of values. Previously these functions returned a list with a single element `undef` if an error occurred. Now these functions return the empty list in these situations. This applies to the following functions:

```
Win32::FsType
Win32::GetOSVersion
```

The remaining functions are unchanged and continue to return `undef` on error even in list context.

The Win32::SetLastError(ERROR) function has been added as a complement to the **Win32::GetLastError()** function.

The new Win32::GetFullPathName(FILENAME) returns the full absolute pathname for FILENAME in scalar context. In list context it returns a two-element list containing the fully qualified directory name and the filename. See Win32.

XSLoader
: The XSLoader extension is a simpler alternative to DynaLoader. See XSLoader.

DBM Filters
: A new feature called "DBM Filters" has been added to all the DBM modules--DB_File, GDBM_File, NDBM_File, ODBM_File, and SDBM_File. DBM Filters add four new methods to each DBM module:

```
filter_store_key
filter_store_value
filter_fetch_key
filter_fetch_value
```

These can be used to filter key-value pairs before the pairs are written to the database or just after they are read from the database. See perldbmfilter for further information.

### Pragmata

`use attrs` is now obsolete, and is only provided for backward-compatibility. It's been replaced by the `sub : attributes` syntax. See "Subroutine Attributes" in perlsub and attributes.

Lexical warnings pragma, `use warnings;`, to control optional warnings. See perllexwarn.

`use filetest` to control the behaviour of filetests (`-r -w` ...). Currently only one subpragma implemented, "use filetest 'access';", that uses **access**(2) or equivalent to check permissions instead of using **stat**(2) as usual. This matters in filesystems where there are ACLs (access control lists): the **stat**(2) might lie, but **access**(2) knows better.

The `open` pragma can be used to specify default disciplines for handle constructors (e.g. **open()**) and for qx//. The two pseudo-disciplines `:raw` and `:crlf` are currently supported on DOS-derivative platforms (i.e. where binmode is not a no-op). See also "**binmode()** can be used to set :crlf and :raw modes".

## Utility Changes

### dprofpp

`dprofpp` is used to display profile data generated using `Devel::DProf`. See dprofpp.

### find2perl

The `find2perl` utility now uses the enhanced features of the File::Find module. The -depth and -follow options are supported. Pod documentation is also included in the script.

### h2xs

The `h2xs` tool can now work in conjunction with `C::Scan` (available from CPAN) to automatically parse real-life header files. The `-M`, `-a`, `-k`, and `-o` options are new.

### perlcc

`perlcc` now supports the C and Bytecode backends. By default, it generates output from the simple C backend rather than the optimized C backend.

Support for non-Unix platforms has been improved.

### perldoc

`perldoc` has been reworked to avoid possible security holes. It will not by default let itself be run as the superuser, but you may still use the **-U** switch to try to make it drop privileges first.

### The Perl Debugger

Many bug fixes and enhancements were added to *perl5db.pl*, the Perl debugger. The help documentation was rearranged. New commands include `< ?`, `> ?`, and `{ ?` to list out current actions, `man` *`docpage`* to run your doc viewer on some perl docset, and support for quoted options. The help information was rearranged, and should be viewable once again if you're using **less** as your pager. A serious security hole was plugged — you should immediately remove all older versions of the Perl debugger as installed in previous releases, all the way back to perl3, from your system to avoid being bitten by this.

## Improved Documentation

Many of the platform-specific README files are now part of the perl installation. See perl for the complete list.

perlapi.pod
: The official list of public Perl API functions.

perlboot.pod
: A tutorial for beginners on object-oriented Perl.

perlcompile.pod
: An introduction to using the Perl Compiler suite.

perldbmfilter.pod
: A howto document on using the DBM filter facility.

perldebug.pod
: All material unrelated to running the Perl debugger, plus all low-level guts-like details that risked crushing the casual user of the debugger, have been relocated from the old manpage to the next entry below.

perldebguts.pod
: This new manpage contains excessively low-level material not related to the Perl debugger, but slightly related to debugging Perl itself. It also contains some arcane internal details of how the debugging process works that may only be of interest to developers of Perl debuggers.

perlfork.pod
: Notes on the **fork()** emulation currently available for the Windows platform.

perlfilter.pod
: An introduction to writing Perl source filters.

perlhack.pod
: Some guidelines for hacking the Perl source code.

perlintern.pod
: A list of internal functions in the Perl source code. (List is currently empty.)

perllexwarn.pod
: Introduction and reference information about lexically scoped warning categories.

perlnumber.pod
: Detailed information about numbers as they are represented in Perl.

perlopentut.pod
: A tutorial on using **open()** effectively.

perlreftut.pod
: A tutorial that introduces the essentials of references.

perltootc.pod
: A tutorial on managing class data for object modules.

perltodo.pod
: Discussion of the most often wanted features that may someday be supported in Perl.

perlunicode.pod
: An introduction to Unicode support features in Perl.

## Performance enhancements

### Simple sort() using { $a <=> $b } and the like are optimized

Many common **sort()** operations using a simple inlined block are now optimized for faster performance.

### Optimized assignments to lexical variables

Certain operations in the RHS of assignment statements have been optimized to directly set the lexical variable on the LHS, eliminating redundant copying overheads.

### Faster subroutine calls

Minor changes in how subroutine calls are handled internally provide marginal improvements in performance.

### delete(), each(), values() and hash iteration are faster

The hash values returned by **delete()**, **each()**, **values()** and hashes in a list context are the actual values in the hash, instead of copies. This results in significantly better performance, because it eliminates

needless copying in most situations.

## Installation and Configuration Improvements

### −Dusethreads means something different

The −Dusethreads flag now enables the experimental interpreter-based thread support by default. To get the flavor of experimental threads that was in 5.005 instead, you need to run Configure with ‘‘−Dusethreads −Duse5005threads’’.

As of v5.6.0, interpreter-threads support is still lacking a way to create new threads from Perl (i.e., `use Thread;` will not work with interpreter threads). `use Thread;` continues to be available when you specify the −Duse5005threads option to Configure, bugs and all.

```
NOTE: Support for threads continues to be an experimental feature.
Interfaces and implementation are subject to sudden and drastic changes.
```

### New Configure flags

The following new flags may be enabled on the Configure command line by running Configure with `−Dflag`.

```
usemultiplicity
usethreads useithreads      (new interpreter threads: no Perl API yet)
usethreads use5005threads   (threads as they were in 5.005)

use64bitint                 (equal to now deprecated 'use64bits')
use64bitall

uselongdouble
usemorebits
uselargefiles
usesocks                    (only SOCKS v5 supported)
```

### Threadedness and 64−bitness now more daring

The Configure options enabling the use of threads and the use of 64−bitness are now more daring in the sense that they no more have an explicit list of operating systems of known threads/64−bit capabilities. In other words: if your operating system has the necessary APIs and datatypes, you should be able just to go ahead and use them, for threads by Configure −Dusethreads, and for 64 bits either explicitly by Configure −Duse64bitint or implicitly if your system has 64−bit wide datatypes. See also ‘‘64−bit support’’.

### Long Doubles

Some platforms have ‘‘long doubles’’, floating point numbers of even larger range than ordinary ‘‘doubles’’. To enable using long doubles for Perl’s scalars, use −Duselongdouble.

### −Dusemorebits

You can enable both −Duse64bitint and −Duselongdouble with −Dusemorebits. See also ‘‘64−bit support’’.

### −Duselargefiles

Some platforms support system APIs that are capable of handling large files (typically, files larger than two gigabytes). Perl will try to use these APIs if you ask for −Duselargefiles.

See ‘‘Large file support’’ for more information.

### installusrbinperl

You can use ‘‘Configure −Uinstallusrbinperl’’ which causes installperl to skip installing perl also as /usr/bin/perl. This is useful if you prefer not to modify /usr/bin for some reason or another but harmful because many scripts assume to find Perl in /usr/bin/perl.

### SOCKS support

You can use ‘‘Configure −Dusesocks’’ which causes Perl to probe for the SOCKS proxy protocol library (v5, not v4). For more information on SOCKS, see:

```
http://www.socks.nec.com/
```

### −A flag

You can ‘‘post-edit’’ the Configure variables using the Configure `−A` switch. The editing happens immediately after the platform specific hints files have been processed but before the actual

configuration process starts. Run `Configure -h` to find out the full `-A` syntax.

#### Enhanced Installation Directories

The installation structure has been enriched to improve the support for maintaining multiple versions of perl, to provide locations for vendor-supplied modules, scripts, and manpages, and to ease maintenance of locally-added modules, scripts, and manpages. See the section on Installation Directories in the INSTALL file for complete details. For most users building and installing from source, the defaults should be fine.

If you previously used `Configure -Dsitelib` or `-Dsitearch` to set special values for library directories, you might wish to consider using the new `-Dsiteprefix` setting instead. Also, if you wish to re-use a config.sh file from an earlier version of perl, you should be sure to check that Configure makes sensible choices for the new directories. See INSTALL for complete details.

#### gcc automatically tried if 'cc' does not seem to be working

In many platforms the vendor-supplied 'cc' is too stripped-down to build Perl (basically, the 'cc' doesn't do ANSI C). If this seems to be the case and the 'cc' does not seem to be the GNU C compiler 'gcc', an automatic attempt is made to find and use 'gcc' instead.

### Platform specific changes

#### Supported platforms

- The Mach CThreads (NEXTSTEP, OPENSTEP) are now supported by the Thread extension.
- GNU/Hurd is now supported.
- Rhapsody/Darwin is now supported.
- EPOC is now supported (on Psion 5).
- The cygwin port (formerly cygwin32) has been greatly improved.

#### DOS

- Perl now works with djgpp 2.02 (and 2.03 alpha).
- Environment variable names are not converted to uppercase any more.
- Incorrect exit codes from backticks have been fixed.
- This port continues to use its own builtin globbing (not File::Glob).

#### OS390 (OpenEdition MVS)

Support for this EBCDIC platform has not been renewed in this release. There are difficulties in reconciling Perl's standardization on UTF-8 as its internal representation for characters with the EBCDIC character set, because the two are incompatible.

It is unclear whether future versions will renew support for this platform, but the possibility exists.

#### VMS

Numerous revisions and extensions to configuration, build, testing, and installation process to accommodate core changes and VMS-specific options.

Expand `%ENV`-handling code to allow runtime mapping to logical names, CLI symbols, and CRTL environ array.

Extension of subprocess invocation code to accept filespecs as command "verbs".

Add to Perl command line processing the ability to use default file types and to recognize Unix-style `2>&1`.

Expansion of File::Spec::VMS routines, and integration into ExtUtils::MM_VMS.

Extension of ExtUtils::MM_VMS to handle complex extensions more flexibly.

Barewords at start of Unix-syntax paths may be treated as text rather than only as logical names.

Optional secure translation of several logical names used internally by Perl.

Miscellaneous bugfixing and porting of new core code to VMS.

Thanks are gladly extended to the many people who have contributed VMS patches, testing, and ideas.

#### Win32

Perl can now emulate **fork()** internally, using multiple interpreters running in different concurrent threads. This support must be enabled at build time. See perlfork for detailed information.

When given a pathname that consists only of a drivename, such as `A:`, **opendir()** and **stat()** now use the current working directory for the drive rather than the drive root.

The builtin XSUB functions in the Win32:: namespace are documented. See Win32.

$^X now contains the full path name of the running executable.

A **Win32::GetLongPathName()** function is provided to complement **Win32::GetFullPathName()** and **Win32::GetShortPathName()**. See Win32.

**POSIX::uname()** is supported.

system(1,...) now returns true process IDs rather than process handles. **kill()** accepts any real process id, rather than strictly return values from system(1,...).

For better compatibility with Unix, `kill(0, $pid)` can now be used to test whether a process exists.

The `Shell` module is supported.

Better support for building Perl under command.com in Windows 95 has been added.

Scripts are read in binary mode by default to allow ByteLoader (and the filter mechanism in general) to work properly. For compatibility, the DATA filehandle will be set to text mode if a carriage return is detected at the end of the line containing the `__END__` or `__DATA__` token; if not, the DATA filehandle will be left open in binary mode. Earlier versions always opened the DATA filehandle in text mode.

The **glob()** operator is implemented via the `File::Glob` extension, which supports glob syntax of the C shell. This increases the flexibility of the **glob()** operator, but there may be compatibility issues for programs that relied on the older globbing syntax. If you want to preserve compatibility with the older syntax, you might want to run perl with `-MFile::DosGlob`. For details and compatibility information, see File::Glob.

## Significant bug fixes

### <HANDLE> on empty files

With `$/` set to `undef`, ''slurping'' an empty file returns a string of zero length (instead of `undef`, as it used to) the first time the HANDLE is read after `$/` is set to `undef`. Further reads yield `undef`.

This means that the following will append ''foo'' to an empty file (it used to do nothing):

```
perl -0777 -pi -e 's/^/foo/' empty_file
```

The behaviour of:

```
perl -pi -e 's/^/foo/' empty_file
```

is unchanged (it continues to leave the file empty).

### `eval '...'` improvements

Line numbers (as reflected by **caller()** and most diagnostics) within `eval '...'` were often incorrect where here documents were involved. This has been corrected.

Lexical lookups for variables appearing in `eval '...'` within functions that were themselves called within an `eval '...'` were searching the wrong place for lexicals. The lexical search now correctly ends at the subroutine's block boundary.

The use of `return` within `eval {...}` caused $@ not to be reset correctly when no exception occurred within the eval. This has been fixed.

Parsing of here documents used to be flawed when they appeared as the replacement expression in `eval 's/.../.../e'`. This has been fixed.

### All compilation errors are true errors

Some ''errors'' encountered at compile time were by necessity generated as warnings followed by eventual termination of the program. This enabled more such errors to be reported in a single run, rather than causing a hard stop at the first error that was encountered.

The mechanism for reporting such errors has been reimplemented to queue compile-time errors and report them at the end of the compilation as true errors rather than as warnings. This fixes cases where error messages leaked through in the form of warnings when code was compiled at run time using `eval STRING`, and also allows such errors to be reliably trapped using `eval "..."`.

### Implicitly closed filehandles are safer

Sometimes implicitly closed filehandles (as when they are localized, and Perl automatically closes them on exiting the scope) could inadvertently set $? or $!. This has been corrected.

### Behavior of list slices is more consistent

When taking a slice of a literal list (as opposed to a slice of an array or hash), Perl used to return an empty list if the result happened to be composed of all undef values.

The new behavior is to produce an empty list if (and only if) the original list was empty. Consider the following example:

```
@a = (1,undef,undef,2)[2,1,2];
```

The old behavior would have resulted in `@a` having no elements. The new behavior ensures it has three undefined elements.

Note in particular that the behavior of slices of the following cases remains unchanged:

```
@a = ()[1,2];
@a = (getpwent)[7,0];
@a = (anything_returning_empty_list())[2,1,2];
@a = @b[2,1,2];
@a = @c{'a','b','c'};
```

See perldata.

### `(\$)` prototype and `$foo{a}`

A scalar reference prototype now correctly allows a hash or array element in that slot.

### `goto &sub` and AUTOLOAD

The `goto &sub` construct works correctly when `&sub` happens to be autoloaded.

### `-bareword` allowed under `use integer`

The autoquoting of barewords preceded by `-` did not work in prior versions when the `integer` pragma was enabled. This has been fixed.

### Failures in DESTROY()

When code in a destructor threw an exception, it went unnoticed in earlier versions of Perl, unless someone happened to be looking in $@ just after the point the destructor happened to run. Such failures are now visible as warnings when warnings are enabled.

### Locale bugs fixed

**printf()** and **sprintf()** previously reset the numeric locale back to the default ''C'' locale. This has been fixed.

Numbers formatted according to the local numeric locale (such as using a decimal comma instead of a decimal dot) caused ''isn't numeric'' warnings, even while the operations accessing those numbers produced correct results. These warnings have been discontinued.

### Memory leaks

The `eval 'return sub {...}'` construct could sometimes leak memory. This has been fixed.

Operations that aren't filehandle constructors used to leak memory when used on invalid filehandles. This has been fixed.

Constructs that modified `@_` could fail to deallocate values in `@_` and thus leak memory. This has been corrected.

### Spurious subroutine stubs after failed subroutine calls

Perl could sometimes create empty subroutine stubs when a subroutine was not found in the package. Such cases stopped later method lookups from progressing into base packages. This has been corrected.

### Taint failures under `-U`

When running in unsafe mode, taint violations could sometimes cause silent failures. This has been fixed.

### END blocks and the `-c` switch

Prior versions used to run BEGIN **and** END blocks when Perl was run in compile-only mode. Since this is typically not the expected behavior, END blocks are not executed anymore when the `-c` switch is

used, or if compilation fails.

See ‘‘Support for CHECK blocks’’ for how to run things when the compile phase ends.

### Potential to leak DATA filehandles

Using the `__DATA__` token creates an implicit filehandle to the file that contains the token. It is the program's responsibility to close it when it is done reading from it.

This caveat is now better explained in the documentation. See perldata.

## New or Changed Diagnostics

‘‘%s’’ variable `%s` masks earlier declaration in same `%s`

(W misc) A ‘‘my’’ or ‘‘our’’ variable has been redeclared in the current scope or statement, effectively eliminating all access to the previous instance. This is almost always a typographical error. Note that the earlier variable will still exist until the end of the scope or until all closure referents to it are destroyed.

‘‘my sub’’ not yet implemented

(F) Lexically scoped subroutines are not yet implemented. Don't try that yet.

‘‘our’’ variable `%s` redeclared

(W misc) You seem to have already declared the same global once before in the current lexical scope.

’!’ allowed only after types `%s`

(F) The ’!’ is allowed in **pack()** and **unpack()** only after certain types. See ‘‘pack’’ in perlfunc.

/ cannot take a count

(F) You had an unpack template indicating a counted-length string, but you have also specified an explicit size for the string. See ‘‘pack’’ in perlfunc.

/ must be followed by a, A or Z

(F) You had an unpack template indicating a counted-length string, which must be followed by one of the letters a, A or Z to indicate what sort of string is to be unpacked. See ‘‘pack’’ in perlfunc.

/ must be followed by a*, A* or Z*

(F) You had a pack template indicating a counted-length string, Currently the only things that can have their length counted are a*, A* or Z*. See ‘‘pack’’ in perlfunc.

/ must follow a numeric type

(F) You had an unpack template that contained a ’#’, but this did not follow some numeric unpack specification. See ‘‘pack’’ in perlfunc.

/%s/: Unrecognized escape \\%c passed through

(W regexp) You used a backslash-character combination which is not recognized by Perl. This combination appears in an interpolated variable or a `'`−delimited regular expression. The character was understood literally.

/%s/: Unrecognized escape \\%c in character class passed through

(W regexp) You used a backslash-character combination which is not recognized by Perl inside character classes. The character was understood literally.

/%s/ should probably be written as ‘‘%s’’

(W syntax) You have used a pattern where Perl expected to find a string, as in the first argument to `join`. Perl will treat the true or false result of matching the pattern against `$_` as the string, which is probably not what you had in mind.

%s() called too early to check prototype

(W prototype) You've called a function that has a prototype before the parser saw a definition or declaration for it, and Perl could not check that the call conforms to the prototype. You need to either add an early prototype declaration for the subroutine in question, or move the subroutine definition ahead of the call to get proper prototype checking. Alternatively, if you are certain that you're calling the function correctly, you may put an ampersand before the name to avoid the warning. See perlsub.

%s argument is not a HASH or ARRAY element
(F) The argument to **exists()** must be a hash or array element, such as:

```
$foo{$bar}
$ref->{"susie"}[12]
```

%s argument is not a HASH or ARRAY element or slice
(F) The argument to **delete()** must be either a hash or array element, such as:

```
$foo{$bar}
$ref->{"susie"}[12]
```

or a hash or array slice, such as:

```
@foo[$bar, $baz, $xyzzy]
@{$ref->[12]}{"susie", "queue"}
```

%s argument is not a subroutine name
(F) The argument to **exists()** for `exists &sub` must be a subroutine name, and not a subroutine call. `exists &sub()` will generate this error.

%s package attribute may clash with future reserved word: %s
(W reserved) A lowercase attribute name was used that had a package-specific handler. That name might have a meaning to Perl itself some day, even though it doesn't yet. Perhaps you should use a mixed-case attribute name, instead. See attributes.

(in cleanup) %s
(W misc) This prefix usually indicates that a **DESTROY()** method raised the indicated exception. Since destructors are usually called by the system at arbitrary points during execution, and often a vast number of times, the warning is issued only once for any number of failures that would otherwise result in the same message being repeated.

Failure of user callbacks dispatched using the `G_KEEPERR` flag could also result in this warning. See ''G_KEEPERR'' in perlcall.

<> should be quotes
(F) You wrote `require <file>` when you should have written `require 'file'`.

Attempt to join self
(F) You tried to join a thread from within itself, which is an impossible task. You may be joining the wrong thread, or you may need to move the **join()** to some other thread.

Bad evalled substitution pattern
(F) You've used the /e switch to evaluate the replacement for a substitution, but perl found a syntax error in the code to evaluate, most likely an unexpected right brace '}'.

Bad **realloc()** ignored
(S) An internal routine called **realloc()** on something that had never been **malloc()**ed in the first place. Mandatory, but can be disabled by setting environment variable `PERL_BADFREE` to 1.

Bareword found in conditional
(W bareword) The compiler found a bareword where it expected a conditional, which often indicates that an || or && was parsed as part of the last argument of the previous construct, for example:

```
open FOO || die;
```

It may also indicate a misspelled constant that has been interpreted as a bareword:

```
use constant TYPO => 1;
if (TYOP) { print "foo" }
```

The `strict` pragma is useful in avoiding such errors.

Binary number > 0b11111111111111111111111111111111 non-portable
(W portable) The binary number you specified is larger than 2**32−1 (4294967295) and therefore non-portable between systems. See perlport for more on portability concerns.

Bit vector size > 32 non-portable
: (W portable) Using bit vector sizes larger than 32 is non-portable.

Buffer overflow in prime_env_iter: %s
: (W internal) A warning peculiar to VMS. While Perl was preparing to iterate over %ENV, it encountered a logical name or symbol definition which was too long, so it was truncated to the string shown.

Can't check filesystem of script "%s"
: (P) For some reason you can't check the filesystem of the script for nosuid.

Can't declare class for non-scalar %s in "%s"
: (S) Currently, only scalar variables can declared with a specific class qualifier in a "my" or "our" declaration. The semantics may be extended for other types of variables in future.

Can't declare %s in "%s"
: (F) Only scalar, array, and hash variables may be declared as "my" or "our" variables. They must have ordinary identifiers as names.

Can't ignore signal CHLD, forcing to default
: (W signal) Perl has detected that it is being run with the SIGCHLD signal (sometimes known as SIGCLD) disabled. Since disabling this signal will interfere with proper determination of exit status of child processes, Perl has reset the signal to its default value. This situation typically indicates that the parent program under which Perl may be running (e.g., cron) is being very careless.

Can't modify non-lvalue subroutine call
: (F) Subroutines meant to be used in lvalue context should be declared as such, see "Lvalue subroutines" in perlsub.

Can't read CRTL environ
: (S) A warning peculiar to VMS. Perl tried to read an element of %ENV from the CRTL's internal environment array and discovered the array was missing. You need to figure out where your CRTL misplaced its environ or define *PERL_ENV_TABLES* (see perlvms) so that environ is not searched.

Can't remove %s: %s, skipping file
: (S) You requested an inplace edit without creating a backup file. Perl was unable to remove the original file to replace it with the modified file. The file was left unmodified.

Can't return %s from lvalue subroutine
: (F) Perl detected an attempt to return illegal lvalues (such as temporary or readonly values) from a subroutine used as an lvalue. This is not allowed.

Can't weaken a nonreference
: (F) You attempted to weaken something that was not a reference. Only references can be weakened.

Character class [:%s:] unknown
: (F) The class in the character class [: :] syntax is unknown. See perlre.

Character class syntax [%s] belongs inside character classes
: (W unsafe) The character class constructs [: :], [= =], and [. .] go *inside* character classes, the [] are part of the construct, for example: /[012[:alpha:]345]/. Note that [= =] and [. .] are not currently implemented; they are simply placeholders for future extensions.

Constant is not %s reference
: (F) A constant value (perhaps declared using the `use constant` pragma) is being dereferenced, but it amounts to the wrong type of reference. The message indicates the type of reference that was expected. This usually indicates a syntax error in dereferencing the constant value. See "Constant Functions" in perlsub and constant.

constant(%s): %s
: (F) The parser found inconsistencies either while attempting to define an overloaded constant, or when trying to find the character name specified in the `\N{...}` escape. Perhaps you forgot to load the corresponding `overload` or `charnames` pragma? See charnames and overload.

CORE::%s is not a keyword
(F) The CORE:: namespace is reserved for Perl keywords.

defined(@array) is deprecated
(D) **defined()** is not usually useful on arrays because it checks for an undefined *scalar* value. If you want to see if the array is empty, just use `if (@array) { # not empty }` for example.

defined(%hash) is deprecated
(D) **defined()** is not usually useful on hashes because it checks for an undefined *scalar* value. If you want to see if the hash is empty, just use `if (%hash) { # not empty }` for example.

Did not produce a valid header
See Server error.

(Did you mean ‘‘local’’ instead of ‘‘our’’?)
(W misc) Remember that ‘‘our’’ does not localize the declared global variable. You have declared it again in the same lexical scope, which seems superfluous.

Document contains no data
See Server error.

entering effective %s failed
(F) While under the `use filetest` pragma, switching the real and effective uids or gids failed.

false [] range ‘‘%s’’ in regexp
(W regexp) A character class range must start and end at a literal character, not another character class like `\d` or `[:alpha:]`. The ‘‘-’’ in your false range is interpreted as a literal ‘‘-’’. Consider quoting the ‘‘-’’, ‘‘\-’’. See perlre.

Filehandle %s opened only for output
(W io) You tried to read from a filehandle opened only for writing. If you intended it to be a read/write filehandle, you needed to open it with ‘‘+<’’ or ‘‘+>’’ or ‘‘+>>’’ instead of with ‘‘<’’ or nothing. If you intended only to read from the file, use ‘‘<’’. See ‘‘open’’ in perlfunc.

**flock()** on closed filehandle %s
(W closed) The filehandle you’re attempting to **flock()** got itself closed some time before now. Check your logic flow. **flock()** operates on filehandles. Are you attempting to call **flock()** on a dirhandle by the same name?

Global symbol ‘‘%s’’ requires explicit package name
(F) You’ve said ‘‘use strict vars’’, which indicates that all variables must either be lexically scoped (using ‘‘my’’), declared beforehand using ‘‘our’’, or explicitly qualified to say which package the global variable is in (using ‘‘::’’).

Hexadecimal number > 0xffffffff non-portable
(W portable) The hexadecimal number you specified is larger than 2**32-1 (4294967295) and therefore non-portable between systems. See perlport for more on portability concerns.

Ill-formed CRTL environ value ‘‘%s’’
(W internal) A warning peculiar to VMS. Perl tried to read the CRTL’s internal environ array, and encountered an element without the = delimiter used to separate keys from values. The element is ignored.

Ill-formed message in prime_env_iter: |%s|
(W internal) A warning peculiar to VMS. Perl tried to read a logical name or CLI symbol definition when preparing to iterate over %ENV, and didn’t see the expected delimiter between key and value, so the line was ignored.

Illegal binary digit %s
(F) You used a digit other than 0 or 1 in a binary number.

Illegal binary digit %s ignored
(W digit) You may have tried to use a digit other than 0 or 1 in a binary number. Interpretation of the binary number stopped before the offending digit.

Illegal number of bits in vec
(F) The number of bits in **vec()** (the third argument) must be a power of two from 1 to 32 (or 64, if your platform supports that).

Integer overflow in %s number
(W overflow) The hexadecimal, octal or binary number you have specified either as a literal or as an argument to **hex()** or **oct()** is too big for your architecture, and has been converted to a floating point number. On a 32−bit architecture the largest hexadecimal, octal or binary number representable without overflow is 0xFFFFFFFF, 037777777777, or 0b11111111111111111111111111111111 respectively. Note that Perl transparently promotes all numbers to a floating point representation internally — subject to loss of precision errors in subsequent operations.

Invalid %s attribute: %s
The indicated attribute for a subroutine or variable was not recognized by Perl or by a user-supplied handler. See attributes.

Invalid %s attributes: %s
The indicated attributes for a subroutine or variable were not recognized by Perl or by a user-supplied handler. See attributes.

invalid [] range ‘‘%s’’ in regexp
The offending range is now explicitly displayed.

Invalid separator character %s in attribute list
(F) Something other than a colon or whitespace was seen between the elements of an attribute list. If the previous attribute had a parenthesised parameter list, perhaps that list was terminated too soon. See attributes.

Invalid separator character %s in subroutine attribute list
(F) Something other than a colon or whitespace was seen between the elements of a subroutine attribute list. If the previous attribute had a parenthesised parameter list, perhaps that list was terminated too soon.

leaving effective %s failed
(F) While under the `use filetest` pragma, switching the real and effective uids or gids failed.

Lvalue subs returning %s not implemented yet
(F) Due to limitations in the current implementation, array and hash values cannot be returned in subroutines used in lvalue context. See ‘‘Lvalue subroutines’’ in perlsub.

Method %s not permitted
See Server error.

Missing %sbrace%s on \N{}
(F) Wrong syntax of character name literal `\N{charname}` within double-quotish context.

Missing command in piped open
(W pipe) You used the `open(FH, "| command")` or `open(FH, "command |")` construction, but the command was missing or blank.

Missing name in ‘‘my sub’’
(F) The reserved syntax for lexically scoped subroutines requires that they have a name with which they can be found.

No %s specified for −%c
(F) The indicated command line switch needs a mandatory argument, but you haven’t specified one.

No package name allowed for variable %s in ‘‘our’’
(F) Fully qualified variable names are not allowed in ‘‘our’’ declarations, because that doesn’t make much sense under existing semantics. Such syntax is reserved for future extensions.

No space allowed after −%c
(F) The argument to the indicated command line switch must follow immediately after the switch, without intervening spaces.

no UTC offset information; assuming local time is UTC
(S) A warning peculiar to VMS. Perl was unable to find the local timezone offset, so it's assuming that local system time is equivalent to UTC. If it's not, define the logical name *SYS$TIMEZONE_DIFFERENTIAL* to translate to the number of seconds which need to be added to UTC to get local time.

Octal number > 037777777777 non-portable
(W portable) The octal number you specified is larger than 2**32−1 (4294967295) and therefore non-portable between systems. See perlport for more on portability concerns.

See also perlport for writing portable code.

panic: del_backref
(P) Failed an internal consistency check while trying to reset a weak reference.

panic: kid popen errno read
(F) forked child returned an incomprehensible message about its errno.

panic: magic_killbackrefs
(P) Failed an internal consistency check while trying to reset all weak references to an object.

Parentheses missing around ‘‘%s’’ list
(W parenthesis) You said something like

```
my $foo, $bar = @_;
```

when you meant

```
my ($foo, $bar) = @_;
```

Remember that ‘‘my’’, ‘‘our’’, and ‘‘local’’ bind tighter than comma.

Possible unintended interpolation of `%s` in string
(W ambiguous) It used to be that Perl would try to guess whether you wanted an array interpolated or a literal @. It no longer does this; arrays are now *always* interpolated into strings. This means that if you try something like:

```
print "fred@example.com";
```

and the array `@example` doesn't exist, Perl is going to print `fred.com`, which is probably not what you wanted. To get a literal @ sign in a string, put a backslash before it, just as you would to get a literal $ sign.

Possible Y2K bug: `%s`
(W y2k) You are concatenating the number 19 with another number, which could be a potential Year 2000 problem.

pragma ‘‘attrs’’ is deprecated, use ‘‘sub NAME : ATTRS’’ instead
(W deprecated) You have written something like this:

```
sub doit
{
    use attrs qw(locked);
}
```

You should use the new declaration syntax instead.

```
sub doit : locked
{
    ...
```

The `use attrs` pragma is now obsolete, and is only provided for backward-compatibility. See ‘‘Subroutine Attributes’’ in perlsub.

Premature end of script headers
See Server error.

Repeat count in pack overflows
(F) You can't specify a repeat count so large that it overflows your signed integers. See ‘‘pack’’ in perlfunc.

Repeat count in unpack overflows
(F) You can't specify a repeat count so large that it overflows your signed integers. See "unpack" in perlfunc.

**realloc()** of freed memory ignored
(S) An internal routine called **realloc()** on something that had already been freed.

Reference is already weak
(W misc) You have attempted to weaken a reference that is already weak. Doing so has no effect.

setpgrp can't take arguments
(F) Your system has the **setpgrp()** from BSD 4.2, which takes no arguments, unlike POSIX **setpgid()**, which takes a process ID and process group ID.

Strange *+?{} on zero-length expression
(W regexp) You applied a regular expression quantifier in a place where it makes no sense, such as on a zero-width assertion. Try putting the quantifier inside the assertion instead. For example, the way to match "abc" provided that it is followed by three repetitions of "xyz" is `/abc(?=(?:xyz){3})/`, not `/abc(?=xyz){3}/`.

switching effective %s is not implemented
(F) While under the `use filetest` pragma, we cannot switch the real and effective uids or gids.

This Perl can't reset CRTL environ elements (%s)
This Perl can't set CRTL environ elements (%s=%s)
(W internal) Warnings peculiar to VMS. You tried to change or delete an element of the CRTL's internal environ array, but your copy of Perl wasn't built with a CRTL that contained the **setenv()** function. You'll need to rebuild Perl with a CRTL that does, or redefine *PERL_ENV_TABLES* (see perlvms) so that the environ array isn't the target of the change to `%ENV` which produced the warning.

Too late to run %s block
(W void) A CHECK or INIT block is being defined during run time proper, when the opportunity to run them has already passed. Perhaps you are loading a file with `require` or `do` when you should be using `use` instead. Or perhaps you should put the `require` or `do` inside a BEGIN block.

Unknown **open()** mode '%s'
(F) The second argument of 3−argument **open()** is not among the list of valid modes: `<`, `>`, `>>`, `+<`, `+>`, `+>>`, `-|`, `|-`.

Unknown process %x sent message to prime_env_iter: %s
(P) An error peculiar to VMS. Perl was reading values for `%ENV` before iterating over it, and someone else stuck a message in the stream of data Perl expected. Someone's very confused, or perhaps trying to subvert Perl's population of `%ENV` for nefarious purposes.

Unrecognized escape \\%c passed through
(W misc) You used a backslash-character combination which is not recognized by Perl. The character was understood literally.

Unterminated attribute parameter in attribute list
(F) The lexer saw an opening (left) parenthesis character while parsing an attribute list, but the matching closing (right) parenthesis character was not found. You may need to add (or remove) a backslash character to get your parentheses to balance. See attributes.

Unterminated attribute list
(F) The lexer found something other than a simple identifier at the start of an attribute, and it wasn't a semicolon or the start of a block. Perhaps you terminated the parameter list of the previous attribute too soon. See attributes.

Unterminated attribute parameter in subroutine attribute list
(F) The lexer saw an opening (left) parenthesis character while parsing a subroutine attribute list, but the matching closing (right) parenthesis character was not found. You may need to add (or remove) a backslash character to get your parentheses to balance.

Unterminated subroutine attribute list
: (F) The lexer found something other than a simple identifier at the start of a subroutine attribute, and it wasn't a semicolon or the start of a block. Perhaps you terminated the parameter list of the previous attribute too soon.

Value of CLI symbol ''%s'' too long
: (W misc) A warning peculiar to VMS. Perl tried to read the value of an `%ENV` element from a CLI symbol table, and found a resultant string longer than 1024 characters. The return value has been truncated to 1024 characters.

Version number must be a constant number
: (P) The attempt to translate a `use Module n.n LIST` statement into its equivalent `BEGIN` block found an internal inconsistency with the version number.

## New tests

lib/attrs
: Compatibility tests for `sub : attrs` vs the older `use attrs`.

lib/env
: Tests for new environment scalar capability (e.g., `use Env qw($BAR);`).

lib/env−array
: Tests for new environment array capability (e.g., `use Env qw(@PATH);`).

lib/io_const
: IO constants (SEEK_*, _IO*).

lib/io_dir
: Directory-related IO methods (new, read, close, rewind, tied delete).

lib/io_multihomed
: INET sockets with multi-homed hosts.

lib/io_poll
: IO **poll()**.

lib/io_unix
: UNIX sockets.

op/attrs
: Regression tests for `my ($x,@y,%z) : attrs` and <sub : attrs>.

op/filetest
: File test operators.

op/lex_assign
: Verify operations that access pad objects (lexicals and temporaries).

op/exists_sub
: Verify `exists &sub` operations.

## Incompatible Changes

### Perl Source Incompatibilities

Beware that any new warnings that have been added or old ones that have been enhanced are **not** considered incompatible changes.

Since all new warnings must be explicitly requested via the `−w` switch or the `warnings` pragma, it is ultimately the programmer's responsibility to ensure that warnings are enabled judiciously.

CHECK is a new keyword
: All subroutine definitions named CHECK are now special. See `/"Support for CHECK blocks"` for more information.

Treatment of list slices of undef has changed
: There is a potential incompatibility in the behavior of list slices that are comprised entirely of undefined values. See ''Behavior of list slices is more consistent''.

Format of `$English::PERL_VERSION` is different
: The English module now sets `$PERL_VERSION` to $^V (a string value) rather than `$]` (a numeric value). This is a potential incompatibility. Send us a report via perlbug if you are

affected by this.

See ‘‘Improved Perl version numbering system’’ for the reasons for this change.

Literals of the form `1.2.3` parse differently
: Previously, numeric literals with more than one dot in them were interpreted as a floating point number concatenated with one or more numbers. Such ‘‘numbers’’ are now parsed as strings composed of the specified ordinals.

  For example, `print 97.98.99` used to output `97.9899` in earlier versions, but now prints `abc`.

  See ‘‘Support for strings represented as a vector of ordinals’’.

Possibly changed pseudo-random number generator
: Perl programs that depend on reproducing a specific set of pseudo-random numbers may now produce different output due to improvements made to the **rand()** builtin. You can use `sh Configure -Drandfunc=rand` to obtain the old behavior.

  See ‘‘Better pseudo-random number generator’’.

Hashing function for hash keys has changed
: Even though Perl hashes are not order preserving, the apparently random order encountered when iterating on the contents of a hash is actually determined by the hashing algorithm used. Improvements in the algorithm may yield a random order that is **different** from that of previous versions, especially when iterating on hashes.

  See ‘‘Better worst-case behavior of hashes’’ for additional information.

`undef` fails on read only values
: Using the `undef` operator on a readonly value (such as `$1`) has the same effect as assigning `undef` to the readonly value — it throws an exception.

Close-on-exec bit may be set on pipe and socket handles
: Pipe and socket handles are also now subject to the close-on-exec behavior determined by the special variable $^F.

  See ‘‘More consistent close-on-exec behavior’’.

Writing `''$$1''` to mean `''${$}1''` is unsupported
: Perl 5.004 deprecated the interpretation of `$$1` and similar within interpolated strings to mean `$$ . "1"`, but still allowed it.

  In Perl 5.6.0 and later, `"$$1"` always means `"${$1}"`.

**delete()**, **each()**, **values()** and `\(%h)`
: operate on aliases to values, not copies

  **delete()**, **each()**, **values()** and hashes (e.g. `\(%h)`) in a list context return the actual values in the hash, instead of copies (as they used to in earlier versions). Typical idioms for using these constructs copy the returned values, but this can make a significant difference when creating references to the returned values. Keys in the hash are still returned as copies when iterating on a hash.

  See also ‘‘**delete()**, **each()**, **values()** and hash iteration are faster’’.

vec(EXPR,OFFSET,BITS) enforces powers-of-two BITS
: **vec()** generates a run-time error if the BITS argument is not a valid power-of-two integer.

Text of some diagnostic output has changed
: Most references to internal Perl operations in diagnostics have been changed to be more descriptive. This may be an issue for programs that may incorrectly rely on the exact text of diagnostics for proper functioning.

%@ has been removed
: The undocumented special variable `%@` that used to accumulate ‘‘background’’ errors (such as those that happen in **DESTROY()**) has been removed, because it could potentially result in memory leaks.

Parenthesized **not()** behaves like a list operator

The `not` operator now falls under the ''if it looks like a function, it behaves like a function'' rule.

As a result, the parenthesized form can be used with `grep` and `map`. The following construct used to be a syntax error before, but it works as expected now:

```
grep not($_), @things;
```

On the other hand, using `not` with a literal list slice may not work. The following previously allowed construct:

```
print not (1,2,3)[0];
```

needs to be written with additional parentheses now:

```
print not((1,2,3)[0]);
```

The behavior remains unaffected when `not` is not followed by parentheses.

Semantics of bareword prototype `(*)` have changed

The semantics of the bareword prototype `*` have changed. Perl 5.005 always coerced simple scalar arguments to a typeglob, which wasn't useful in situations where the subroutine must distinguish between a simple scalar and a typeglob. The new behavior is to not coerce bareword arguments to a typeglob. The value will always be visible as either a simple scalar or as a reference to a typeglob.

See ''More functional bareword prototype (*)''.

Semantics of bit operators may have changed on 64−bit platforms

If your platform is either natively 64−bit or if Perl has been configured to used 64−bit integers, i.e., `$Config{ivsize}` is 8, there may be a potential incompatibility in the behavior of bitwise numeric operators (& | ^ ~ << >>). These operators used to strictly operate on the lower 32 bits of integers in previous versions, but now operate over the entire native integral width. In particular, note that unary ~ will produce different results on platforms that have different `$Config{ivsize}`. For portability, be sure to mask off the excess bits in the result of unary ~, e.g., `~$x & 0xffffffff`.

See ''Bit operators support full native integer width''.

More builtins taint their results

As described in ''Improved security features'', there may be more sources of taint in a Perl program.

To avoid these new tainting behaviors, you can build Perl with the Configure option `-Accflags=-DINCOMPLETE_TAINTS`. Beware that the ensuing perl binary may be insecure.

## C Source Incompatibilities

`PERL_POLLUTE`

Release 5.005 grandfathered old global symbol names by providing preprocessor macros for extension source compatibility. As of release 5.6.0, these preprocessor definitions are not available by default. You need to explicitly compile perl with `-DPERL_POLLUTE` to get these definitions. For extensions still using the old symbols, this option can be specified via MakeMaker:

```
perl Makefile.PL POLLUTE=1
```

`PERL_IMPLICIT_CONTEXT`

This new build option provides a set of macros for all API functions such that an implicit interpreter/thread context argument is passed to every API function. As a result of this, something like `sv_setsv(foo,bar)` amounts to a macro invocation that actually translates to something like `Perl_sv_setsv(my_perl,foo,bar)`. While this is generally expected to not have any significant source compatibility issues, the difference between a macro and a real function call will need to be considered.

This means that there **is** a source compatibility issue as a result of this if your extensions attempt to use pointers to any of the Perl API functions.

Note that the above issue is not relevant to the default build of Perl, whose interfaces continue to

match those of prior versions (but subject to the other options described here).

See ''Background and PERL_IMPLICIT_CONTEXT'' in perlguts for detailed information on the ramifications of building Perl with this option.

```
NOTE: PERL_IMPLICIT_CONTEXT is automatically enabled whenever Perl is buil
with one of -Dusethreads, -Dusemultiplicity, or both.  It is not
intended to be enabled by users at this time.
```

PERL_POLLUTE_MALLOC

Enabling Perl's malloc in release 5.005 and earlier caused the namespace of the system's malloc family of functions to be usurped by the Perl versions, since by default they used the same names. Besides causing problems on platforms that do not allow these functions to be cleanly replaced, this also meant that the system versions could not be called in programs that used Perl's malloc. Previous versions of Perl have allowed this behaviour to be suppressed with the HIDEMYMALLOC and EMBEDMYMALLOC preprocessor definitions.

As of release 5.6.0, Perl's malloc family of functions have default names distinct from the system versions. You need to explicitly compile perl with `-DPERL_POLLUTE_MALLOC` to get the older behaviour. HIDEMYMALLOC and EMBEDMYMALLOC have no effect, since the behaviour they enabled is now the default.

Note that these functions do **not** constitute Perl's memory allocation API. See ''Memory Allocation'' in perlguts for further information about that.

### Compatible C Source API Changes

PATCHLEVEL is now PERL_VERSION

The cpp macros `PERL_REVISION`, `PERL_VERSION`, and `PERL_SUBVERSION` are now available by default from perl.h, and reflect the base revision, patchlevel, and subversion respectively. `PERL_REVISION` had no prior equivalent, while `PERL_VERSION` and `PERL_SUBVERSION` were previously available as `PATCHLEVEL` and `SUBVERSION`.

The new names cause less pollution of the **cpp** namespace and reflect what the numbers have come to stand for in common practice. For compatibility, the old names are still supported when *patchlevel.h* is explicitly included (as required before), so there is no source incompatibility from the change.

### Binary Incompatibilities

In general, the default build of this release is expected to be binary compatible for extensions built with the 5.005 release or its maintenance versions. However, specific platforms may have broken binary compatibility due to changes in the defaults used in hints files. Therefore, please be sure to always check the platform-specific README files for any notes to the contrary.

The usethreads or usemultiplicity builds are **not** binary compatible with the corresponding builds in 5.005.

On platforms that require an explicit list of exports (AIX, OS/2 and Windows, among others), purely internal symbols such as parser functions and the run time opcodes are not exported by default. Perl 5.005 used to export all functions irrespective of whether they were considered part of the public API or not.

For the full list of public API functions, see perlapi.

## Known Problems

### Localizing a tied hash element may leak memory

As of the 5.6.1 release, there is a known leak when code such as this is executed:

```
use Tie::Hash;
tie my %tie_hash => 'Tie::StdHash';

...

local($tie_hash{Foo}) = 1; # leaks
```

### Known test failures

- 64−bit builds

  Subtest #15 of lib/b.t may fail under 64−bit builds on platforms such as HP-UX PA64 and Linux IA64. The issue is still being investigated.

  The lib/io_multihomed test may hang in HP-UX if Perl has been configured to be 64−bit. Because other 64−bit platforms do not hang in this test, HP-UX is suspect. All other tests pass in 64−bit HP-UX. The test attempts to create and connect to ''multihomed'' sockets (sockets which have multiple IP addresses).

  Note that 64−bit support is still experimental.

- Failure of Thread tests

  The subtests 19 and 20 of lib/thr5005.t test are known to fail due to fundamental problems in the 5.005 threading implementation. These are not new failures — Perl 5.005_0x has the same bugs, but didn't have these tests. (Note that support for 5.005−style threading remains experimental.)

- NEXTSTEP 3.3 POSIX test failure

  In NEXTSTEP 3.3p2 the implementation of the **strftime**(3) in the operating system libraries is buggy: the `%j` format numbers the days of a month starting from zero, which, while being logical to programmers, will cause the subtests 19 to 27 of the lib/posix test may fail.

- Tru64 (aka Digital UNIX, aka DEC OSF/1) lib/sdbm test failure with gcc

  If compiled with gcc 2.95 the lib/sdbm test will fail (dump core). The cure is to use the vendor cc, it comes with the operating system and produces good code.

### EBCDIC platforms not fully supported

In earlier releases of Perl, EBCDIC environments like OS390 (also known as Open Edition MVS) and VM-ESA were supported. Due to changes required by the UTF−8 (Unicode) support, the EBCDIC platforms are not supported in Perl 5.6.0.

The 5.6.1 release improves support for EBCDIC platforms, but they are not fully supported yet.

### UNICOS/mk CC failures during Configure run

In UNICOS/mk the following errors may appear during the Configure run:

```
Guessing which symbols your C compiler and preprocessor define...
CC−20 cc: ERROR File = try.c, Line = 3
...
  bad switch yylook 79bad switch yylook 79bad switch yylook 79bad switch y
...
4 errors detected in the compilation of "try.c".
```

The culprit is the broken awk of UNICOS/mk. The effect is fortunately rather mild: Perl itself is not adversely affected by the error, only the h2ph utility coming with Perl, and that is rather rarely needed these days.

### Arrow operator and arrays

When the left argument to the arrow operator `->` is an array, or the `scalar` operator operating on an array, the result of the operation must be considered erroneous. For example:

```
@x->[2]
scalar(@x)->[2]
```

These expressions will get run-time errors in some future release of Perl.

### Experimental features

As discussed above, many features are still experimental. Interfaces and implementation of these features are subject to change, and in extreme cases, even subject to removal in some future release of Perl. These features include the following:

Threads
Unicode
64−bit support

Lvalue subroutines
Weak references
The pseudo-hash data type
The Compiler suite
Internal implementation of file globbing
The DB module
The regular expression code constructs:
`(?{ code })` and `(??{ code })`

## Obsolete Diagnostics

Character class syntax [: :] is reserved for future extensions

(W) Within regular expression character classes ([]) the syntax beginning with "[:" and ending with ":]" is reserved for future extensions. If you need to represent those character sequences inside a regular expression character class, just quote the square brackets with the backslash: "\[:" and ":\]".

Ill-formed logical name |%s| in prime_env_iter

(W) A warning peculiar to VMS. A logical name was encountered when preparing to iterate over `%ENV` which violates the syntactic rules governing logical names. Because it cannot be translated normally, it is skipped, and will not appear in `%ENV`. This may be a benign occurrence, as some software packages might directly modify logical name tables and introduce nonstandard names, or it may indicate that a logical name table has been corrupted.

In string, @%s now must be written as \@%s

The description of this error used to say:

```
(Someday it will simply assume that an unbackslashed @
 interpolates an array.)
```

That day has come, and this fatal error has been removed. It has been replaced by a non-fatal warning instead. See "Arrays now always interpolate into double-quoted strings" for details.

Probable precedence problem on `%s`

(W) The compiler found a bareword where it expected a conditional, which often indicates that an || or && was parsed as part of the last argument of the previous construct, for example:

```
open FOO || die;
```

regexp too big

(F) The current implementation of regular expressions uses shorts as address offsets within a string. Unfortunately this means that if the regular expression compiles to longer than 32767, it'll blow up. Usually when you want a regular expression this big, there is a better way to do it with multiple statements. See perlre.

Use of "$$<digit>" to mean "${$}<digit>" is deprecated

(D) Perl versions before 5.004 misinterpreted any type marker followed by "$" and a digit. For example, "$$0" was incorrectly taken to mean "${$}0" instead of "${$0}". This bug is (mostly) fixed in Perl 5.004.

However, the developers of Perl 5.004 could not fix this bug completely, because at least two widely-used modules depend on the old meaning of "$$0" in a string. So Perl 5.004 still interprets "$$<digit>" in the old (broken) way inside strings; but it generates this message as a warning. And in Perl 5.005, this special treatment will cease.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup. There may also be information at http://www.perl.com/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## HISTORY

Written by Gurusamy Sarathy <*gsar@ActiveState.com*>, with many contributions from The Perl Porters.

Send omissions or corrections to <*perlbug@perl.org*>.

## NAME

perl56delta − what's new for perl v5.6.0

## DESCRIPTION

This document describes differences between the 5.005 release and the 5.6.0 release.

## Core Enhancements

### Interpreter cloning, threads, and concurrency

Perl 5.6.0 introduces the beginnings of support for running multiple interpreters concurrently in different threads. In conjunction with the **perl_clone()** API call, which can be used to selectively duplicate the state of any given interpreter, it is possible to compile a piece of code once in an interpreter, clone that interpreter one or more times, and run all the resulting interpreters in distinct threads.

On the Windows platform, this feature is used to emulate **fork()** at the interpreter level. See perlfork for details about that.

This feature is still in evolution. It is eventually meant to be used to selectively clone a subroutine and data reachable from that subroutine in a separate interpreter and run the cloned subroutine in a separate thread. Since there is no shared data between the interpreters, little or no locking will be needed (unless parts of the symbol table are explicitly shared). This is obviously intended to be an easy-to-use replacement for the existing threads support.

Support for cloning interpreters and interpreter concurrency can be enabled using the −Dusethreads Configure option (see win32/Makefile for how to enable it on Windows.) The resulting perl executable will be functionally identical to one that was built with −Dmultiplicity, but the **perl_clone()** API call will only be available in the former.

−Dusethreads enables the cpp macro USE_ITHREADS by default, which in turn enables Perl source code changes that provide a clear separation between the op tree and the data it operates with. The former is immutable, and can therefore be shared between an interpreter and all of its clones, while the latter is considered local to each interpreter, and is therefore copied for each clone.

Note that building Perl with the −Dusemultiplicity Configure option is adequate if you wish to run multiple **independent** interpreters concurrently in different threads. −Dusethreads only provides the additional functionality of the **perl_clone()** API call and other support for running **cloned** interpreters concurrently.

```
NOTE: This is an experimental feature.  Implementation details are
subject to change.
```

### Lexically scoped warning categories

You can now control the granularity of warnings emitted by perl at a finer level using the `use warnings` pragma. warnings and perllexwarn have copious documentation on this feature.

### Unicode and UTF−8 support

Perl now uses UTF−8 as its internal representation for character strings. The `utf8` and `bytes` pragmas are used to control this support in the current lexical scope. See perlunicode, utf8 and bytes for more information.

This feature is expected to evolve quickly to support some form of I/O disciplines that can be used to specify the kind of input and output data (bytes or characters). Until that happens, additional modules from CPAN will be needed to complete the toolkit for dealing with Unicode.

```
NOTE: This should be considered an experimental feature.  Implementation
details are subject to change.
```

### Support for interpolating named characters

The new `\N` escape interpolates named characters within strings. For example, `"Hi! \N{WHITE SMILING FACE}"` evaluates to a string with a unicode smiley face at the end.

### "our" declarations

An ''our'' declaration introduces a value that can be best understood as a lexically scoped symbolic alias to a global variable in the package that was current where the variable was declared. This is mostly useful as an alternative to the `vars` pragma, but also provides the opportunity to introduce typing and other attributes for such variables. See ''our'' in perlfunc.

### Support for strings represented as a vector of ordinals

Literals of the form `v1.2.3.4` are now parsed as a string composed of characters with the specified ordinals. This is an alternative, more readable way to construct (possibly unicode) strings instead of interpolating characters, as in `"\x{1}\x{2}\x{3}\x{4}"`. The leading `v` may be omitted if there are more than two ordinals, so `1.2.3` is parsed the same as `v1.2.3`.

Strings written in this form are also useful to represent version ''numbers''. It is easy to compare such version ''numbers'' (which are really just plain strings) using any of the usual string comparison operators `eq`, `ne`, `lt`, `gt`, etc., or perform bitwise string operations on them using `|`, `&`, etc.

In conjunction with the new `$^V` magic variable (which contains the perl version as a string), such literals can be used as a readable way to check if you're running a particular version of Perl:

```
# this will parse in older versions of Perl also
if ($^V and $^V gt v5.6.0) {
    # new features supported
}
```

`require` and `use` also have some special magic to support such literals, but this particular usage should be avoided because it leads to misleading error messages under versions of Perl which don't support vector strings. Using a true version number will ensure correct behavior in all versions of Perl:

```
require 5.006;    # run time check for v5.6
use 5.006_001;    # compile time check for v5.6.1
```

Also, `sprintf` and `printf` support the Perl-specific format flag `%v` to print ordinals of characters in arbitrary strings:

```
printf "v%vd", $^V;         # prints current version, such as "v5.5.650"
printf "%*vX", ":", $addr;  # formats IPv6 address
printf "%*vb", " ", $bits;  # displays bitstring
```

See ''Scalar value constructors'' in perldata for additional information.

### Improved Perl version numbering system

Beginning with Perl version 5.6.0, the version number convention has been changed to a ''dotted integer'' scheme that is more commonly found in open source projects.

Maintenance versions of v5.6.0 will be released as v5.6.1, v5.6.2 etc. The next development series following v5.6.0 will be numbered v5.7.x, beginning with v5.7.0, and the next major production release following v5.6.0 will be v5.8.0.

The English module now sets `$PERL_VERSION` to $^V (a string value) rather than `$]` (a numeric value). (This is a potential incompatibility. Send us a report via perlbug if you are affected by this.)

The v1.2.3 syntax is also now legal in Perl. See ''Support for strings represented as a vector of ordinals'' for more on that.

To cope with the new versioning system's use of at least three significant digits for each version component, the method used for incrementing the subversion number has also changed slightly. We assume that versions older than v5.6.0 have been incrementing the subversion component in multiples of 10. Versions after v5.6.0 will increment them by 1. Thus, using the new notation, 5.005_03 is the ''same'' as v5.5.30, and the first maintenance version following v5.6.0 will be v5.6.1 (which should be read as being equivalent to a floating point value of 5.006_001 in the older format, stored in `$]`).

### New syntax for declaring subroutine attributes

Formerly, if you wanted to mark a subroutine as being a method call or as requiring an automatic **lock()** when it is entered, you had to declare that with a `use attrs` pragma in the body of the subroutine. That can now be accomplished with declaration syntax, like this:

```
sub mymethod : locked method;
...
sub mymethod : locked method {
    ...
}

sub othermethod :locked :method;
...
```

```
sub othermethod :locked :method {
    ...
}
```

(Note how only the first `:` is mandatory, and whitespace surrounding the `:` is optional.)

*AutoSplit.pm* and *SelfLoader.pm* have been updated to keep the attributes with the stubs they provide. See attributes.

### File and directory handles can be autovivified

Similar to how constructs such as `$x->[0]` autovivify a reference, handle constructors (**open()**, **opendir()**, **pipe()**, **socketpair()**, **sysopen()**, **socket()**, and **accept()**) now autovivify a file or directory handle if the handle passed to them is an uninitialized scalar variable. This allows the constructs such as `open(my $fh, ...)` and `open(local $fh,...)` to be used to create filehandles that will conveniently be closed automatically when the scope ends, provided there are no other references to them. This largely eliminates the need for typeglobs when opening filehandles that must be passed around, as in the following example:

```
sub myopen {
    open my $fh, "@_"
         or die "Can't open '@_': $!";
    return $fh;
}

{
    my $f = myopen("</etc/motd");
    print <$f>;
    # $f implicitly closed here
}
```

### open() with more than two arguments

If **open()** is passed three arguments instead of two, the second argument is used as the mode and the third argument is taken to be the file name. This is primarily useful for protecting against unintended magic behavior of the traditional two-argument form. See ‘‘open’’ in perlfunc.

### 64−bit support

Any platform that has 64−bit integers either

```
(1) natively as longs or ints
(2) via special compiler flags
(3) using long long or int64_t
```

is able to use ‘‘quads’’ (64−bit integers) as follows:

- constants (decimal, hexadecimal, octal, binary) in the code
- arguments to **oct()** and **hex()**
- arguments to **print()**, **printf()** and **sprintf()** (flag prefixes ll, L, q)
- printed as such
- **pack()** and **unpack()** ‘‘q’’ and ‘‘Q’’ formats
- in basic arithmetics: + − * / % (NOTE: operating close to the limits of the integer values may produce surprising results)
- in bit arithmetics: & | ^ ~ << >> (NOTE: these used to be forced to be 32 bits wide but now operate on the full native width.)
- **vec()**

Note that unless you have the case (a) you will have to configure and compile Perl using the −Duse64bitint Configure flag.

```
NOTE: The Configure flags -Duselonglong and -Duse64bits have been
deprecated.  Use -Duse64bitint instead.
```

There are actually two modes of 64−bitness: the first one is achieved using Configure −Duse64bitint and the second one using Configure −Duse64bitall. The difference is that the first one is minimal and

the second one maximal. The first works in more places than the second.

The `use64bitint` does only as much as is required to get 64−bit integers into Perl (this may mean, for example, using ‘‘long longs’’) while your memory may still be limited to 2 gigabytes (because your pointers could still be 32−bit). Note that the name `64bitint` does not imply that your C compiler will be using 64−bit `int`s (it might, but it doesn't have to): the `use64bitint` means that you will be able to have 64 bits wide scalar values.

The `use64bitall` goes all the way by attempting to switch also integers (if it can), longs (and pointers) to being 64−bit. This may create an even more binary incompatible Perl than −Duse64bitint: the resulting executable may not run at all in a 32−bit box, or you may have to reboot/reconfigure/rebuild your operating system to be 64−bit aware.

Natively 64−bit systems like Alpha and Cray need neither −Duse64bitint nor −Duse64bitall.

Last but not least: note that due to Perl's habit of always using floating point numbers, the quads are still not true integers. When quads overflow their limits (0...18_446_744_073_709_551_615 unsigned, −9_223_372_036_854_775_808...9_223_372_036_854_775_807 signed), they are silently promoted to floating point numbers, after which they will start losing precision (in their lower digits).

```
NOTE: 64−bit support is still experimental on most platforms.
Existing support only covers the LP64 data model.  In particular, the
LLP64 data model is not yet supported.  64−bit libraries and system
APIs on many platforms have not stabilized−−your mileage may vary.
```

### Large file support

If you have filesystems that support ‘‘large files’’ (files larger than 2 gigabytes), you may now also be able to create and access them from Perl.

```
NOTE: The default action is to enable large file support, if
available on the platform.
```

If the large file support is on, and you have a Fcntl constant O_LARGEFILE, the O_LARGEFILE is automatically added to the flags of **sysopen()**.

Beware that unless your filesystem also supports ‘‘sparse files’’ seeking to umpteen petabytes may be inadvisable.

Note that in addition to requiring a proper file system to do large files you may also need to adjust your per-process (or your per-system, or per-process-group, or per-user-group) maximum filesize limits before running Perl scripts that try to handle large files, especially if you intend to write such files.

Finally, in addition to your process/process group maximum filesize limits, you may have quota limits on your filesystems that stop you (your user id or your user group id) from using large files.

Adjusting your process/user/group/file system/operating system limits is outside the scope of Perl core language. For process limits, you may try increasing the limits using your shell's limits/limit/ulimit command before running Perl. The BSD::Resource extension (not included with the standard Perl distribution) may also be of use, it offers the getrlimit/setrlimit interface that can be used to adjust process resource usage limits, including the maximum filesize limit.

### Long doubles

In some systems you may be able to use long doubles to enhance the range and precision of your double precision floating point numbers (that is, Perl's numbers). Use Configure −Duselongdouble to enable this support (if it is available).

### “more bits”

You can ‘‘Configure −Dusemorebits’’ to turn on both the 64−bit support and the long double support.

### Enhanced support for sort() subroutines

Perl subroutines with a prototype of `($$)`, and XSUBs in general, can now be used as sort subroutines. In either case, the two elements to be compared are passed as normal parameters in `@_`. See ‘‘sort’’ in perlfunc.

For unprototyped sort subroutines, the historical behavior of passing the elements to be compared as the global variables `$a` and `$b` remains unchanged.

`sort $coderef @foo` **allowed**

**sort()** did not accept a subroutine reference as the comparison function in earlier versions. This is now permitted.

**File globbing implemented internally**

Perl now uses the File::Glob implementation of the **glob()** operator automatically. This avoids using an external csh process and the problems associated with it.

```
NOTE: This is currently an experimental feature.  Interfaces and
implementation are subject to change.
```

**Support for CHECK blocks**

In addition to `BEGIN`, `INIT`, `END`, `DESTROY` and `AUTOLOAD`, subroutines named `CHECK` are now special. These are queued up during compilation and behave similar to END blocks, except they are called at the end of compilation rather than at the end of execution. They cannot be called directly.

**POSIX character class syntax [: :] supported**

For example to match alphabetic characters use /[[:alpha:]]/. See perlre for details.

**Better pseudo-random number generator**

In 5.005_0x and earlier, perl's **rand()** function used the C library **rand**(3) function. As of 5.005_52, Configure tests for **drand48()**, **random()**, and **rand()** (in that order) and picks the first one it finds.

These changes should result in better random numbers from **rand()**.

**Improved `qw//` operator**

The `qw//` operator is now evaluated at compile time into a true list instead of being replaced with a run time call to `split()`. This removes the confusing misbehaviour of `qw//` in scalar context, which had inherited that behaviour from **split()**.

Thus:

```
$foo = ($bar) = qw(a b c); print "$foo|$bar\n";
```

now correctly prints ‘‘3|a’’, instead of ‘‘2|a’’.

**Better worst-case behavior of hashes**

Small changes in the hashing algorithm have been implemented in order to improve the distribution of lower order bits in the hashed value. This is expected to yield better performance on keys that are repeated sequences.

**pack() format 'Z' supported**

The new format type 'Z' is useful for packing and unpacking null-terminated strings. See ‘‘pack’’ in perlfunc.

**pack() format modifier '!' supported**

The new format type modifier '!' is useful for packing and unpacking native shorts, ints, and longs. See ‘‘pack’’ in perlfunc.

**pack() and unpack() support counted strings**

The template character '/' can be used to specify a counted string type to be packed or unpacked. See ‘‘pack’’ in perlfunc.

**Comments in pack() templates**

The '#' character in a template introduces a comment up to end of the line. This facilitates documentation of **pack()** templates.

**Weak references**

In previous versions of Perl, you couldn't cache objects so as to allow them to be deleted if the last reference from outside the cache is deleted. The reference in the cache would hold a reference count on the object and the objects would never be destroyed.

Another familiar problem is with circular references. When an object references itself, its reference count would never go down to zero, and it would not get destroyed until the program is about to exit.

Weak references solve this by allowing you to ‘‘weaken’’ any reference, that is, make it not count towards the reference count. When the last non-weak reference to an object is deleted, the object is destroyed and all the weak references to the object are automatically undef-ed.

To use this feature, you need the Devel::WeakRef package from CPAN, which contains additional

documentation.

```
NOTE: This is an experimental feature.  Details are subject to change.
```

### Binary numbers supported

Binary numbers are now supported as literals, in s?printf formats, and `oct()`:

```
$answer = 0b101010;
printf "The answer is: %b\n", oct("0b101010");
```

### Lvalue subroutines

Subroutines can now return modifiable lvalues. See ''Lvalue subroutines'' in perlsub.

```
NOTE: This is an experimental feature.  Details are subject to change.
```

### Some arrows may be omitted in calls through references

Perl now allows the arrow to be omitted in many constructs involving subroutine calls through references. For example, `$foo[10]->('foo')` may now be written `$foo[10]('foo')`. This is rather similar to how the arrow may be omitted from `$foo[10]->{'foo'}`. Note however, that the arrow is still required for `foo(10)->('bar')`.

### Boolean assignment operators are legal lvalues

Constructs such as `($a ||= 2) += 1` are now allowed.

### exists() is supported on subroutine names

The **exists()** builtin now works on subroutine names. A subroutine is considered to exist if it has been declared (even if implicitly). See ''exists'' in perlfunc for examples.

### exists() and delete() are supported on array elements

The **exists()** and **delete()** builtins now work on simple arrays as well. The behavior is similar to that on hash elements.

**exists()** can be used to check whether an array element has been initialized. This avoids autovivifying array elements that don't exist. If the array is tied, the **EXISTS()** method in the corresponding tied package will be invoked.

**delete()** may be used to remove an element from the array and return it. The array element at that position returns to its uninitialized state, so that testing for the same element with **exists()** will return false. If the element happens to be the one at the end, the size of the array also shrinks up to the highest element that tests true for **exists()**, or 0 if none such is found. If the array is tied, the **DELETE()** method in the corresponding tied package will be invoked.

See ''exists'' in perlfunc and ''delete'' in perlfunc for examples.

### Pseudo-hashes work better

Dereferencing some types of reference values in a pseudo-hash, such as `$ph->{foo}[1]`, was accidentally disallowed. This has been corrected.

When applied to a pseudo-hash element, **exists()** now reports whether the specified value exists, not merely if the key is valid.

**delete()** now works on pseudo-hashes. When given a pseudo-hash element or slice it deletes the values corresponding to the keys (but not the keys themselves). See ''Pseudo-hashes: Using an array as a hash'' in perlref.

Pseudo-hash slices with constant keys are now optimized to array lookups at compile-time.

List assignments to pseudo-hash slices are now supported.

The `fields` pragma now provides ways to create pseudo-hashes, via **fields::new()** and **fields::phash()**. See fields.

```
NOTE: The pseudo-hash data type continues to be experimental.
Limiting oneself to the interface elements provided by the
fields pragma will provide protection from any future changes.
```

### Automatic flushing of output buffers

**fork()**, **exec()**, **system()**, qx//, and pipe **open()**s now flush buffers of all files opened for output when the operation was attempted. This mostly eliminates confusing buffering mishaps suffered by users unaware of how Perl internally handles I/O.

This is not supported on some platforms like Solaris where a suitably correct implementation of fflush(NULL) isn't available.

### Better diagnostics on meaningless filehandle operations

Constructs such as `open(<FH>)` and `close(<FH>)` are compile time errors. Attempting to read from filehandles that were opened only for writing will now produce warnings (just as writing to read-only filehandles does).

### Where possible, buffered data discarded from duped input filehandle

`open(NEW, "<&OLD")` now attempts to discard any data that was previously read and buffered in `OLD` before duping the handle. On platforms where doing this is allowed, the next read operation on `NEW` will return the same data as the corresponding operation on `OLD`. Formerly, it would have returned the data from the start of the following disk block instead.

### eof() has the same old magic as <>

`eof()` would return true if no attempt to read from `<>` had yet been made. `eof()` has been changed to have a little magic of its own, it now opens the `<>` files.

### binmode() can be used to set :crlf and :raw modes

**binmode()** now accepts a second argument that specifies a discipline for the handle in question. The two pseudo-disciplines ''`:raw`'' and ''`:crlf`'' are currently supported on DOS-derivative platforms. See ''binmode'' in perlfunc and open.

### `-T` filetest recognizes UTF-8 encoded files as "text"

The algorithm used for the `-T` filetest has been enhanced to correctly identify UTF-8 content as ''text''.

### system(), backticks and pipe open now reflect exec() failure

On Unix and similar platforms, **system()**, **qx()** and open(FOO, ''cmd |'') etc., are implemented via **fork()** and **exec()**. When the underlying **exec()** fails, earlier versions did not report the error properly, since the **exec()** happened to be in a different process.

The child process now communicates with the parent about the error in launching the external command, which allows these constructs to return with their usual error value and set $!.

### Improved diagnostics

Line numbers are no longer suppressed (under most likely circumstances) during the global destruction phase.

Diagnostics emitted from code running in threads other than the main thread are now accompanied by the thread ID.

Embedded null characters in diagnostics now actually show up. They used to truncate the message in prior versions.

`$foo::a` and `$foo::b` are now exempt from ''possible typo'' warnings only if **sort()** is encountered in package `foo`.

Unrecognized alphabetic escapes encountered when parsing quote constructs now generate a warning, since they may take on new semantics in later versions of Perl.

Many diagnostics now report the internal operation in which the warning was provoked, like so:

```
Use of uninitialized value in concatenation (.) at (eval 1) line 1.
Use of uninitialized value in print at (eval 1) line 1.
```

Diagnostics that occur within eval may also report the file and line number where the eval is located, in addition to the eval sequence number and the line number within the evaluated text itself. For example:

```
Not enough arguments for scalar at (eval 4)[newlib/perl5db.pl:1411] line 2, a
```

### Diagnostics follow STDERR

Diagnostic output now goes to whichever file the `STDERR` handle is pointing at, instead of always going to the underlying C runtime library's `stderr`.

### More consistent close-on-exec behavior

On systems that support a close-on-exec flag on filehandles, the flag is now set for any handles created by **pipe()**, **socketpair()**, **socket()**, and **accept()**, if that is warranted by the value of $^F that may be in effect. Earlier versions neglected to set the flag for handles created with these operators. See ''pipe'' in perlfunc, ''socketpair'' in perlfunc, ''socket'' in perlfunc, ''accept'' in perlfunc, and ''$^F'' in perlvar.

#### syswrite() ease-of-use

The length argument of `syswrite()` has become optional.

#### Better syntax checks on parenthesized unary operators

Expressions such as:

```
print defined(&foo,&bar,&baz);
print uc("foo","bar","baz");
undef($foo,&bar);
```

used to be accidentally allowed in earlier versions, and produced unpredictable behaviour. Some produced ancillary warnings when used in this way; others silently did the wrong thing.

The parenthesized forms of most unary operators that expect a single argument now ensure that they are not called with more than one argument, making the cases shown above syntax errors. The usual behaviour of:

```
print defined &foo, &bar, &baz;
print uc "foo", "bar", "baz";
undef $foo, &bar;
```

remains unchanged. See perlop.

#### Bit operators support full native integer width

The bit operators (& | ^ ~ << >>) now operate on the full native integral width (the exact size of which is available in `$Config{ivsize}`). For example, if your platform is either natively 64−bit or if Perl has been configured to use 64−bit integers, these operations apply to 8 bytes (as opposed to 4 bytes on 32−bit platforms). For portability, be sure to mask off the excess bits in the result of unary `~`, e.g., `~$x & 0xffffffff`.

#### Improved security features

More potentially unsafe operations taint their results for improved security.

The `passwd` and `shell` fields returned by the **getpwent()**, **getpwnam()**, and **getpwuid()** are now tainted, because the user can affect their own encrypted password and login shell.

The variable modified by **shmread()**, and messages returned by **msgrcv()** (and its object-oriented interface IPC::SysV::Msg::rcv) are also tainted, because other untrusted processes can modify messages and shared memory segments for their own nefarious purposes.

#### More functional bareword prototype (*)

Bareword prototypes have been rationalized to enable them to be used to override builtins that accept barewords and interpret them in a special way, such as `require` or `do`.

Arguments prototyped as `*` will now be visible within the subroutine as either a simple scalar or as a reference to a typeglob. See ‘‘Prototypes’’ in perlsub.

#### `require` and `do` may be overridden

`require` and `do 'file'` operations may be overridden locally by importing subroutines of the same name into the current package (or globally by importing them into the CORE::GLOBAL:: namespace). Overriding `require` will also affect `use`, provided the override is visible at compile-time. See ‘‘Overriding Built-in Functions’’ in perlsub.

#### $^X variables may now have names longer than one character

Formerly, $^X was synonymous with ${‘‘\cX’’}, but $^XY was a syntax error. Now variable names that begin with a control character may be arbitrarily long. However, for compatibility reasons, these variables *must* be written with explicit braces, as `${^XY}` for example. `${^XYZ}` is synonymous with ${‘‘\cXYZ’’}. Variable names with more than one control character, such as `${^XY^Z}`, are illegal.

The old syntax has not changed. As before, ‘^X’ may be either a literal control-X character or the two-character sequence ‘caret’ plus ‘X’. When braces are omitted, the variable name stops after the control character. Thus `"$^XYZ"` continues to be synonymous with `$^X . "YZ"` as before.

As before, lexical variables may not have names beginning with control characters. As before, variables whose names begin with a control character are always forced to be in package ‘main’. All such variables are reserved for future extensions, except those that begin with `^_`, which may be used by user programs and are guaranteed not to acquire special meaning in any future version of Perl.

### New variable $^C reflects -c switch

`$^C` has a boolean value that reflects whether perl is being run in compile-only mode (i.e. via the `-c` switch). Since BEGIN blocks are executed under such conditions, this variable enables perl code to determine whether actions that make sense only during normal running are warranted. See perlvar.

### New variable $^V contains Perl version as a string

`$^V` contains the Perl version number as a string composed of characters whose ordinals match the version numbers, i.e. v5.6.0. This may be used in string comparisons.

See `Support for strings represented as a vector of ordinals` for an example.

### Optional Y2K warnings

If Perl is built with the cpp macro `PERL_Y2KWARN` defined, it emits optional warnings when concatenating the number 19 with another number.

This behavior must be specifically enabled when running Configure. See *INSTALL* and *README.Y2K*.

### Arrays now always interpolate into double-quoted strings

In double-quoted strings, arrays now interpolate, no matter what. The behavior in earlier versions of perl 5 was that arrays would interpolate into strings if the array had been mentioned before the string was compiled, and otherwise Perl would raise a fatal compile-time error. In versions 5.000 through 5.003, the error was

```
Literal @example now requires backslash
```

In versions 5.004_01 through 5.6.0, the error was

```
In string, @example now must be written as \@example
```

The idea here was to get people into the habit of writing `"fred\@example.com"` when they wanted a literal `@` sign, just as they have always written `"Give me back my \$5"` when they wanted a literal `$` sign.

Starting with 5.6.1, when Perl now sees an `@` sign in a double-quoted string, it *always* attempts to interpolate an array, regardless of whether or not the array has been used or declared already. The fatal error has been downgraded to an optional warning:

```
Possible unintended interpolation of @example in string
```

This warns you that `"fred@example.com"` is going to turn into `fred.com` if you don't backslash the `@`. See http://perl.plover.com/at-error.html for more details about the history here.

### @- and @+ provide starting/ending offsets of regex matches

The new magic variables @- and @+ provide the starting and ending offsets, respectively, of $&, `$1`, `$2`, etc. See perlvar for details.

## Modules and Pragmata

### Modules

attributes

While used internally by Perl as a pragma, this module also provides a way to fetch subroutine and variable attributes. See attributes.

B The Perl Compiler suite has been extensively reworked for this release. More of the standard Perl test suite passes when run under the Compiler, but there is still a significant way to go to achieve production quality compiled executables.

```
NOTE: The Compiler suite remains highly experimental.  The
generated code may not be correct, even when it manages to execute
without errors.
```

Benchmark

Overall, Benchmark results exhibit lower average error and better timing accuracy.

You can now run tests for *n* seconds instead of guessing the right number of tests to run: e.g., timethese(-5, ...) will run each code for at least 5 CPU seconds. Zero as the "number of repetitions" means "for at least 3 CPU seconds". The output format has also changed. For example:

```
use Benchmark;$x=3;timethese(-5,{a=>sub{$x*$x},b=>sub{$x**2}})
```

will now output something like this:

```
Benchmark: running a, b, each for at least 5 CPU seconds...
         a:  5 wallclock secs ( 5.77 usr +  0.00 sys =  5.77 CPU) @ 200551.
         b:  4 wallclock secs ( 5.00 usr +  0.02 sys =  5.02 CPU) @ 159605.
```

New features: ‘‘each for at least N CPU seconds...’’, ‘‘wallclock secs’’, and the ‘‘@ operations/CPU second (n=operations)’’.

**timethese()** now returns a reference to a hash of Benchmark objects containing the test results, keyed on the names of the tests.

**timethis()** now returns the iterations field in the Benchmark result object instead of 0.

**timethese()**, **timethis()**, and the new **cmpthese()** (see below) can also take a format specifier of ’none’ to suppress output.

A new function **countit()** is just like **timeit()** except that it takes a TIME instead of a COUNT.

A new function **cmpthese()** prints a chart comparing the results of each test returned from a **timethese()** call. For each possible pair of tests, the percentage speed difference (iters/sec or seconds/iter) is shown.

For other details, see Benchmark.

ByteLoader
The ByteLoader is a dedicated extension to generate and run Perl bytecode. See ByteLoader.

constant
References can now be used.

The new version also allows a leading underscore in constant names, but disallows a double leading underscore (as in ‘‘__LINE__’’). Some other names are disallowed or warned against, including BEGIN, END, etc. Some names which were forced into main:: used to fail silently in some cases; now they’re fatal (outside of main::) and an optional warning (inside of main::). The ability to detect whether a constant had been set with a given name has been added.

See constant.

charnames
This pragma implements the `\N` string escape. See charnames.

Data::Dumper
A `Maxdepth` setting can be specified to avoid venturing too deeply into deep data structures. See Data::Dumper.

The XSUB implementation of **Dump()** is now automatically called if the `Useqq` setting is not in use.

Dumping `qr//` objects works correctly.

DB `DB` is an experimental module that exposes a clean abstraction to Perl’s debugging API.

DB_File
DB_File can now be built with Berkeley DB versions 1, 2 or 3. See `ext/DB_File/Changes`.

Devel::DProf
Devel::DProf, a Perl source code profiler has been added. See Devel::DProf and dprofpp.

Devel::Peek
The Devel::Peek module provides access to the internal representation of Perl variables and data. It is a data debugging tool for the XS programmer.

Dumpvalue
The Dumpvalue module provides screen dumps of Perl data.

DynaLoader
DynaLoader now supports a **dl_unload_file()** function on platforms that support unloading shared objects using **dlclose()**.

Perl can also optionally arrange to unload all extension shared objects loaded by Perl. To enable

this, build Perl with the Configure option `-Accflags=-DDL_UNLOAD_ALL_AT_EXIT`. (This maybe useful if you are using Apache with mod_perl.)

English
: `$PERL_VERSION` now stands for `$^V` (a string value) rather than for `$]` (a numeric value).

Env
: Env now supports accessing environment variables like PATH as array variables.

Fcntl
: More Fcntl constants added: F_SETLK64, F_SETLKW64, O_LARGEFILE for large file (more than 4GB) access (NOTE: the O_LARGEFILE is automatically added to **sysopen()** flags if large file support has been configured, as is the default), Free/Net/OpenBSD locking behaviour flags F_FLOCK, F_POSIX, Linux F_SHLCK, and O_ACCMODE: the combined mask of O_RDONLY, O_WRONLY, and O_RDWR. The **seek()**/**sysseek()** constants SEEK_SET, SEEK_CUR, and SEEK_END are available via the `:seek` tag. The **chmod()**/**stat()** S_IF* constants and S_IS* functions are available via the `:mode` tag.

File::Compare
: A **compare_text()** function has been added, which allows custom comparison functions. See File::Compare.

File::Find
: File::Find now works correctly when the **wanted()** function is either autoloaded or is a symbolic reference.

  A bug that caused File::Find to lose track of the working directory when pruning top-level directories has been fixed.

  File::Find now also supports several other options to control its behavior. It can follow symbolic links if the `follow` option is specified. Enabling the `no_chdir` option will make File::Find skip changing the current directory when walking directories. The `untaint` flag can be useful when running with taint checks enabled.

  See File::Find.

File::Glob
: This extension implements BSD-style file globbing. By default, it will also be used for the internal implementation of the **glob()** operator. See File::Glob.

File::Spec
: New methods have been added to the File::Spec module: **devnull()** returns the name of the null device (/dev/null on Unix) and **tmpdir()** the name of the temp directory (normally /tmp on Unix). There are now also methods to convert between absolute and relative filenames: **abs2rel()** and **rel2abs()**. For compatibility with operating systems that specify volume names in file paths, the **splitpath()**, **splitdir()**, and **catdir()** methods have been added.

File::Spec::Functions
: The new File::Spec::Functions modules provides a function interface to the File::Spec module. Allows shorthand

```
$fullname = catfile($dir1, $dir2, $file);
```

  instead of

```
$fullname = File::Spec->catfile($dir1, $dir2, $file);
```

Getopt::Long
: Getopt::Long licensing has changed to allow the Perl Artistic License as well as the GPL. It used to be GPL only, which got in the way of non-GPL applications that wanted to use Getopt::Long.

  Getopt::Long encourages the use of Pod::Usage to produce help messages. For example:

```
    use Getopt::Long;
    use Pod::Usage;
    my $man = 0;
    my $help = 0;
    GetOptions('help|?' => \$help, man => \$man) or pod2usage(2);
    pod2usage(1) if $help;
    pod2usage(−exitstatus => 0, −verbose => 2) if $man;

    __END__

    =head1 NAME

    sample − Using Getopt::Long and Pod::Usage

    =head1 SYNOPSIS

    sample [options] [file ...]

     Options:
       −help            brief help message
       −man             full documentation

    =head1 OPTIONS

    =over 8

    =item B<−help>

    Print a brief help message and exits.

    =item B<−man>

    Prints the manual page and exits.

    =back

    =head1 DESCRIPTION

    B<This program> will read the given input file(s) and do something
    useful with the contents thereof.

    =cut
```

See Pod::Usage for details.

A bug that prevented the non-option call-back <> from being specified as the first argument has been fixed.

To specify the characters < and > as option starters, use ><. Note, however, that changing option starters is strongly deprecated.

IO **write()** and **syswrite()** will now accept a single-argument form of the call, for consistency with Perl's **syswrite()**.

You can now create a TCP-based IO::Socket::INET without forcing a connect attempt. This allows you to configure its options (like making it non-blocking) and then call **connect()** manually.

A bug that prevented the **IO::Socket::protocol()** accessor from ever returning the correct value has been corrected.

IO::Socket::connect now uses non-blocking IO instead of **alarm()** to do connect timeouts.

IO::Socket::accept now uses **select()** instead of **alarm()** for doing timeouts.

IO::Socket::INET->new now sets $! correctly on failure. $@ is still set for backwards compatibility.

JPL Java Perl Lingo is now distributed with Perl. See jpl/README for more information.

lib `use lib` now weeds out any trailing duplicate entries. `no lib` removes all named entries.

Math::BigInt
The bitwise operations <<, >>, &, |, and ~ are now supported on bigints.

Math::Complex
The accessor methods Re, Im, arg, abs, rho, and theta can now also act as mutators (accessor `$z->`**Re()**, mutator `$z->`**Re**(3)).

The class method `display_format` and the corresponding object method `display_format`, in addition to accepting just one argument, now can also accept a parameter hash. Recognized keys of a parameter hash are `"style"`, which corresponds to the old one parameter case, and two new parameters: `"format"`, which is a **printf()**-style format string (defaults usually to `"%.15g"`, you can revert to the default by setting the format string to `undef`) used for both parts of a complex number, and `"polar_pretty_print"` (defaults to true), which controls whether an attempt is made to try to recognize small multiples and rationals of pi (2pi, pi/2) at the argument (angle) of a polar complex number.

The potentially disruptive change is that in list context both methods now *return the parameter hash*, instead of only the value of the `"style"` parameter.

Math::Trig
A little bit of radial trigonometry (cylindrical and spherical), radial coordinate conversions, and the great circle distance were added.

Pod::Parser, Pod::InputObjects
Pod::Parser is a base class for parsing and selecting sections of pod documentation from an input stream. This module takes care of identifying pod paragraphs and commands in the input and hands off the parsed paragraphs and commands to user-defined methods which are free to interpret or translate them as they see fit.

Pod::InputObjects defines some input objects needed by Pod::Parser, and for advanced users of Pod::Parser that need more about a command besides its name and text.

As of release 5.6.0 of Perl, Pod::Parser is now the officially sanctioned ''base parser code'' recommended for use by all pod2xxx translators. Pod::Text (pod2text) and Pod::Man (pod2man) have already been converted to use Pod::Parser and efforts to convert Pod::HTML (pod2html) are already underway. For any questions or comments about pod parsing and translating issues and utilities, please use the pod-people@perl.org mailing list.

For further information, please see Pod::Parser and Pod::InputObjects.

Pod::Checker, podchecker
This utility checks pod files for correct syntax, according to perlpod. Obvious errors are flagged as such, while warnings are printed for mistakes that can be handled gracefully. The checklist is not complete yet. See Pod::Checker.

Pod::ParseUtils, Pod::Find
These modules provide a set of gizmos that are useful mainly for pod translators. Pod::Find traverses directory structures and returns found pod files, along with their canonical names (like `File::Spec::Unix`). Pod::ParseUtils contains **Pod::List** (useful for storing pod list information), **Pod::Hyperlink** (for parsing the contents of `L<>` sequences) and **Pod::Cache** (for caching information about pod files, e.g., link nodes).

Pod::Select, podselect
Pod::Select is a subclass of Pod::Parser which provides a function named ''**podselect()**'' to filter out user-specified sections of raw pod documentation from an input stream. podselect is a script that provides access to Pod::Select from other scripts to be used as a filter. See Pod::Select.

Pod::Usage, pod2usage

Pod::Usage provides the function ‘‘**pod2usage()**’’ to print usage messages for a Perl script based on its embedded pod documentation. The **pod2usage()** function is generally useful to all script authors since it lets them write and maintain a single source (the pods) for documentation, thus removing the need to create and maintain redundant usage message text consisting of information already in the pods.

There is also a pod2usage script which can be used from other kinds of scripts to print usage messages from pods (even for non-Perl scripts with pods embedded in comments).

For details and examples, please see Pod::Usage.

Pod::Text and Pod::Man

Pod::Text has been rewritten to use Pod::Parser. While **pod2text()** is still available for backwards compatibility, the module now has a new preferred interface. See Pod::Text for the details. The new Pod::Text module is easily subclassed for tweaks to the output, and two such subclasses (Pod::Text::Termcap for man-page-style bold and underlining using termcap information, and Pod::Text::Color for markup with ANSI color sequences) are now standard.

pod2man has been turned into a module, Pod::Man, which also uses Pod::Parser. In the process, several outstanding bugs related to quotes in section headers, quoting of code escapes, and nested lists have been fixed. pod2man is now a wrapper script around this module.

SDBM_File

An EXISTS method has been added to this module (and **sdbm_exists()** has been added to the underlying sdbm library), so one can now call exists on an SDBM_File tied hash and get the correct result, rather than a runtime error.

A bug that may have caused data loss when more than one disk block happens to be read from the database in a single **FETCH()** has been fixed.

Sys::Syslog

Sys::Syslog now uses XSUBs to access facilities from syslog.h so it no longer requires syslog.ph to exist.

Sys::Hostname

Sys::Hostname now uses XSUBs to call the C library’s **gethostname()** or **uname()** if they exist.

Term::ANSIColor

Term::ANSIColor is a very simple module to provide easy and readable access to the ANSI color and highlighting escape sequences, supported by most ANSI terminal emulators. It is now included standard.

Time::Local

The **timelocal()** and **timegm()** functions used to silently return bogus results when the date fell outside the machine’s integer range. They now consistently **croak()** if the date falls in an unsupported range.

Win32

The error return value in list context has been changed for all functions that return a list of values. Previously these functions returned a list with a single element `undef` if an error occurred. Now these functions return the empty list in these situations. This applies to the following functions:

```
Win32::FsType
Win32::GetOSVersion
```

The remaining functions are unchanged and continue to return `undef` on error even in list context.

The Win32::SetLastError(ERROR) function has been added as a complement to the **Win32::GetLastError()** function.

The new Win32::GetFullPathName(FILENAME) returns the full absolute pathname for FILENAME in scalar context. In list context it returns a two-element list containing the fully qualified directory name and the filename. See Win32.

XSLoader
: The XSLoader extension is a simpler alternative to DynaLoader. See XSLoader.

DBM Filters
: A new feature called "DBM Filters" has been added to all the DBM modules--DB_File, GDBM_File, NDBM_File, ODBM_File, and SDBM_File. DBM Filters add four new methods to each DBM module:

```
filter_store_key
filter_store_value
filter_fetch_key
filter_fetch_value
```

These can be used to filter key-value pairs before the pairs are written to the database or just after they are read from the database. See perldbmfilter for further information.

### Pragmata

`use attrs` is now obsolete, and is only provided for backward-compatibility. It's been replaced by the `sub : attributes` syntax. See "Subroutine Attributes" in perlsub and attributes.

Lexical warnings pragma, `use warnings;`, to control optional warnings. See perllexwarn.

`use filetest` to control the behaviour of filetests (`-r -w` ...). Currently only one subpragma implemented, "use filetest 'access';", that uses **access**(2) or equivalent to check permissions instead of using **stat**(2) as usual. This matters in filesystems where there are ACLs (access control lists): the **stat**(2) might lie, but **access**(2) knows better.

The `open` pragma can be used to specify default disciplines for handle constructors (e.g. **open()**) and for qx//. The two pseudo-disciplines `:raw` and `:crlf` are currently supported on DOS-derivative platforms (i.e. where binmode is not a no-op). See also "**binmode()** can be used to set :crlf and :raw modes".

## Utility Changes

### dprofpp

`dprofpp` is used to display profile data generated using `Devel::DProf`. See dprofpp.

### find2perl

The `find2perl` utility now uses the enhanced features of the File::Find module. The -depth and -follow options are supported. Pod documentation is also included in the script.

### h2xs

The `h2xs` tool can now work in conjunction with `C::Scan` (available from CPAN) to automatically parse real-life header files. The `-M`, `-a`, `-k`, and `-o` options are new.

### perlcc

`perlcc` now supports the C and Bytecode backends. By default, it generates output from the simple C backend rather than the optimized C backend.

Support for non-Unix platforms has been improved.

### perldoc

`perldoc` has been reworked to avoid possible security holes. It will not by default let itself be run as the superuser, but you may still use the **-U** switch to try to make it drop privileges first.

### The Perl Debugger

Many bug fixes and enhancements were added to *perl5db.pl*, the Perl debugger. The help documentation was rearranged. New commands include `< ?`, `> ?`, and `{ ?` to list out current actions, `man` *`docpage`* to run your doc viewer on some perl docset, and support for quoted options. The help information was rearranged, and should be viewable once again if you're using **less** as your pager. A serious security hole was plugged — you should immediately remove all older versions of the Perl debugger as installed in previous releases, all the way back to perl3, from your system to avoid being bitten by this.

## Improved Documentation

Many of the platform-specific README files are now part of the perl installation. See perl for the complete list.

perlapi.pod
: The official list of public Perl API functions.

perlboot.pod
: A tutorial for beginners on object-oriented Perl.

perlcompile.pod
: An introduction to using the Perl Compiler suite.

perldbmfilter.pod
: A howto document on using the DBM filter facility.

perldebug.pod
: All material unrelated to running the Perl debugger, plus all low-level guts-like details that risked crushing the casual user of the debugger, have been relocated from the old manpage to the next entry below.

perldebguts.pod
: This new manpage contains excessively low-level material not related to the Perl debugger, but slightly related to debugging Perl itself. It also contains some arcane internal details of how the debugging process works that may only be of interest to developers of Perl debuggers.

perlfork.pod
: Notes on the **fork()** emulation currently available for the Windows platform.

perlfilter.pod
: An introduction to writing Perl source filters.

perlhack.pod
: Some guidelines for hacking the Perl source code.

perlintern.pod
: A list of internal functions in the Perl source code. (List is currently empty.)

perllexwarn.pod
: Introduction and reference information about lexically scoped warning categories.

perlnumber.pod
: Detailed information about numbers as they are represented in Perl.

perlopentut.pod
: A tutorial on using **open()** effectively.

perlreftut.pod
: A tutorial that introduces the essentials of references.

perltootc.pod
: A tutorial on managing class data for object modules.

perltodo.pod
: Discussion of the most often wanted features that may someday be supported in Perl.

perlunicode.pod
: An introduction to Unicode support features in Perl.

## Performance enhancements

### Simple sort() using { $a <=> $b } and the like are optimized

Many common **sort()** operations using a simple inlined block are now optimized for faster performance.

### Optimized assignments to lexical variables

Certain operations in the RHS of assignment statements have been optimized to directly set the lexical variable on the LHS, eliminating redundant copying overheads.

### Faster subroutine calls

Minor changes in how subroutine calls are handled internally provide marginal improvements in performance.

### delete(), each(), values() and hash iteration are faster

The hash values returned by **delete()**, **each()**, **values()** and hashes in a list context are the actual values in the hash, instead of copies. This results in significantly better performance, because it eliminates

needless copying in most situations.

## Installation and Configuration Improvements

### −Dusethreads means something different

The −Dusethreads flag now enables the experimental interpreter-based thread support by default. To get the flavor of experimental threads that was in 5.005 instead, you need to run Configure with ‘‘−Dusethreads −Duse5005threads’’.

As of v5.6.0, interpreter-threads support is still lacking a way to create new threads from Perl (i.e., `use Thread;` will not work with interpreter threads). `use Thread;` continues to be available when you specify the −Duse5005threads option to Configure, bugs and all.

```
NOTE: Support for threads continues to be an experimental feature.
Interfaces and implementation are subject to sudden and drastic changes.
```

### New Configure flags

The following new flags may be enabled on the Configure command line by running Configure with `−Dflag`.

```
usemultiplicity
usethreads useithreads      (new interpreter threads: no Perl API yet)
usethreads use5005threads   (threads as they were in 5.005)

use64bitint                 (equal to now deprecated 'use64bits')
use64bitall

uselongdouble
usemorebits
uselargefiles
usesocks                    (only SOCKS v5 supported)
```

### Threadedness and 64−bitness now more daring

The Configure options enabling the use of threads and the use of 64−bitness are now more daring in the sense that they no more have an explicit list of operating systems of known threads/64−bit capabilities. In other words: if your operating system has the necessary APIs and datatypes, you should be able just to go ahead and use them, for threads by Configure −Dusethreads, and for 64 bits either explicitly by Configure −Duse64bitint or implicitly if your system has 64−bit wide datatypes. See also ‘‘64−bit support’’.

### Long Doubles

Some platforms have ‘‘long doubles’’, floating point numbers of even larger range than ordinary ‘‘doubles’’. To enable using long doubles for Perl’s scalars, use −Duselongdouble.

### −Dusemorebits

You can enable both −Duse64bitint and −Duselongdouble with −Dusemorebits. See also ‘‘64−bit support’’.

### −Duselargefiles

Some platforms support system APIs that are capable of handling large files (typically, files larger than two gigabytes). Perl will try to use these APIs if you ask for −Duselargefiles.

See ‘‘Large file support’’ for more information.

### installusrbinperl

You can use ‘‘Configure −Uinstallusrbinperl’’ which causes installperl to skip installing perl also as /usr/bin/perl. This is useful if you prefer not to modify /usr/bin for some reason or another but harmful because many scripts assume to find Perl in /usr/bin/perl.

### SOCKS support

You can use ‘‘Configure −Dusesocks’’ which causes Perl to probe for the SOCKS proxy protocol library (v5, not v4). For more information on SOCKS, see:

```
http://www.socks.nec.com/
```

### −A flag

You can ‘‘post-edit’’ the Configure variables using the Configure `−A` switch. The editing happens immediately after the platform specific hints files have been processed but before the actual

configuration process starts. Run `Configure -h` to find out the full `-A` syntax.

#### Enhanced Installation Directories

The installation structure has been enriched to improve the support for maintaining multiple versions of perl, to provide locations for vendor-supplied modules, scripts, and manpages, and to ease maintenance of locally-added modules, scripts, and manpages. See the section on Installation Directories in the INSTALL file for complete details. For most users building and installing from source, the defaults should be fine.

If you previously used `Configure -Dsitelib` or `-Dsitearch` to set special values for library directories, you might wish to consider using the new `-Dsiteprefix` setting instead. Also, if you wish to re-use a config.sh file from an earlier version of perl, you should be sure to check that Configure makes sensible choices for the new directories. See INSTALL for complete details.

## Platform specific changes

### Supported platforms

- The Mach CThreads (NEXTSTEP, OPENSTEP) are now supported by the Thread extension.
- GNU/Hurd is now supported.
- Rhapsody/Darwin is now supported.
- EPOC is now supported (on Psion 5).
- The cygwin port (formerly cygwin32) has been greatly improved.

#### DOS

- Perl now works with djgpp 2.02 (and 2.03 alpha).
- Environment variable names are not converted to uppercase any more.
- Incorrect exit codes from backticks have been fixed.
- This port continues to use its own builtin globbing (not File::Glob).

#### OS390 (OpenEdition MVS)

Support for this EBCDIC platform has not been renewed in this release. There are difficulties in reconciling Perl's standardization on UTF-8 as its internal representation for characters with the EBCDIC character set, because the two are incompatible.

It is unclear whether future versions will renew support for this platform, but the possibility exists.

#### VMS

Numerous revisions and extensions to configuration, build, testing, and installation process to accommodate core changes and VMS-specific options.

Expand `%ENV`-handling code to allow runtime mapping to logical names, CLI symbols, and CRTL environ array.

Extension of subprocess invocation code to accept filespecs as command "verbs".

Add to Perl command line processing the ability to use default file types and to recognize Unix-style `2>&1`.

Expansion of File::Spec::VMS routines, and integration into ExtUtils::MM_VMS.

Extension of ExtUtils::MM_VMS to handle complex extensions more flexibly.

Barewords at start of Unix-syntax paths may be treated as text rather than only as logical names.

Optional secure translation of several logical names used internally by Perl.

Miscellaneous bugfixing and porting of new core code to VMS.

Thanks are gladly extended to the many people who have contributed VMS patches, testing, and ideas.

#### Win32

Perl can now emulate **fork()** internally, using multiple interpreters running in different concurrent threads. This support must be enabled at build time. See perlfork for detailed information.

When given a pathname that consists only of a drivename, such as `A:`, **opendir()** and **stat()** now use the current working directory for the drive rather than the drive root.

The builtin XSUB functions in the Win32:: namespace are documented. See Win32.

$^X now contains the full path name of the running executable.

A **Win32::GetLongPathName()** function is provided to complement **Win32::GetFullPathName()** and **Win32::GetShortPathName()**. See Win32.

**POSIX::uname()** is supported.

system(1,...) now returns true process IDs rather than process handles. **kill()** accepts any real process id, rather than strictly return values from system(1,...).

For better compatibility with Unix, `kill(0, $pid)` can now be used to test whether a process exists.

The `Shell` module is supported.

Better support for building Perl under command.com in Windows 95 has been added.

Scripts are read in binary mode by default to allow ByteLoader (and the filter mechanism in general) to work properly. For compatibility, the DATA filehandle will be set to text mode if a carriage return is detected at the end of the line containing the `__END__` or `__DATA__` token; if not, the DATA filehandle will be left open in binary mode. Earlier versions always opened the DATA filehandle in text mode.

The **glob()** operator is implemented via the `File::Glob` extension, which supports glob syntax of the C shell. This increases the flexibility of the **glob()** operator, but there may be compatibility issues for programs that relied on the older globbing syntax. If you want to preserve compatibility with the older syntax, you might want to run perl with `-MFile::DosGlob`. For details and compatibility information, see File::Glob.

## Significant bug fixes

### <HANDLE> on empty files

With `$/` set to `undef`, ''slurping'' an empty file returns a string of zero length (instead of `undef`, as it used to) the first time the HANDLE is read after `$/` is set to `undef`. Further reads yield `undef`.

This means that the following will append ''foo'' to an empty file (it used to do nothing):

```
perl -0777 -pi -e 's/^/foo/' empty_file
```

The behaviour of:

```
perl -pi -e 's/^/foo/' empty_file
```

is unchanged (it continues to leave the file empty).

### `eval '...'` improvements

Line numbers (as reflected by **caller()** and most diagnostics) within `eval '...'` were often incorrect where here documents were involved. This has been corrected.

Lexical lookups for variables appearing in `eval '...'` within functions that were themselves called within an `eval '...'` were searching the wrong place for lexicals. The lexical search now correctly ends at the subroutine's block boundary.

The use of `return` within `eval {...}` caused $@ not to be reset correctly when no exception occurred within the eval. This has been fixed.

Parsing of here documents used to be flawed when they appeared as the replacement expression in `eval 's/.../.../e'`. This has been fixed.

### All compilation errors are true errors

Some ''errors'' encountered at compile time were by necessity generated as warnings followed by eventual termination of the program. This enabled more such errors to be reported in a single run, rather than causing a hard stop at the first error that was encountered.

The mechanism for reporting such errors has been reimplemented to queue compile-time errors and report them at the end of the compilation as true errors rather than as warnings. This fixes cases where error messages leaked through in the form of warnings when code was compiled at run time using `eval STRING`, and also allows such errors to be reliably trapped using `eval "..."`.

### Implicitly closed filehandles are safer

Sometimes implicitly closed filehandles (as when they are localized, and Perl automatically closes them on exiting the scope) could inadvertently set $? or $!. This has been corrected.

### Behavior of list slices is more consistent

When taking a slice of a literal list (as opposed to a slice of an array or hash), Perl used to return an empty list if the result happened to be composed of all undef values.

The new behavior is to produce an empty list if (and only if) the original list was empty. Consider the following example:

```
@a = (1,undef,undef,2)[2,1,2];
```

The old behavior would have resulted in `@a` having no elements. The new behavior ensures it has three undefined elements.

Note in particular that the behavior of slices of the following cases remains unchanged:

```
@a = ()[1,2];
@a = (getpwent)[7,0];
@a = (anything_returning_empty_list())[2,1,2];
@a = @b[2,1,2];
@a = @c{'a','b','c'};
```

See perldata.

### `(\$)` prototype and `$foo{a}`

A scalar reference prototype now correctly allows a hash or array element in that slot.

### `goto &sub` and AUTOLOAD

The `goto &sub` construct works correctly when `&sub` happens to be autoloaded.

### `-bareword` allowed under `use integer`

The autoquoting of barewords preceded by `-` did not work in prior versions when the `integer` pragma was enabled. This has been fixed.

### Failures in DESTROY()

When code in a destructor threw an exception, it went unnoticed in earlier versions of Perl, unless someone happened to be looking in $@ just after the point the destructor happened to run. Such failures are now visible as warnings when warnings are enabled.

### Locale bugs fixed

**printf()** and **sprintf()** previously reset the numeric locale back to the default ''C'' locale. This has been fixed.

Numbers formatted according to the local numeric locale (such as using a decimal comma instead of a decimal dot) caused ''isn't numeric'' warnings, even while the operations accessing those numbers produced correct results. These warnings have been discontinued.

### Memory leaks

The `eval 'return sub {...}'` construct could sometimes leak memory. This has been fixed.

Operations that aren't filehandle constructors used to leak memory when used on invalid filehandles. This has been fixed.

Constructs that modified `@_` could fail to deallocate values in `@_` and thus leak memory. This has been corrected.

### Spurious subroutine stubs after failed subroutine calls

Perl could sometimes create empty subroutine stubs when a subroutine was not found in the package. Such cases stopped later method lookups from progressing into base packages. This has been corrected.

### Taint failures under `-U`

When running in unsafe mode, taint violations could sometimes cause silent failures. This has been fixed.

### END blocks and the `-c` switch

Prior versions used to run BEGIN **and** END blocks when Perl was run in compile-only mode. Since this is typically not the expected behavior, END blocks are not executed anymore when the `-c` switch is used, or if compilation fails.

See ''Support for CHECK blocks'' for how to run things when the compile phase ends.

### Potential to leak DATA filehandles

Using the `__DATA__` token creates an implicit filehandle to the file that contains the token. It is the program's responsibility to close it when it is done reading from it.

This caveat is now better explained in the documentation. See perldata.

## New or Changed Diagnostics

''%s'' variable `%s` masks earlier declaration in same `%s`

(W misc) A ''my'' or ''our'' variable has been redeclared in the current scope or statement, effectively eliminating all access to the previous instance. This is almost always a typographical error. Note that the earlier variable will still exist until the end of the scope or until all closure referents to it are destroyed.

''my sub'' not yet implemented

(F) Lexically scoped subroutines are not yet implemented. Don't try that yet.

''our'' variable `%s` redeclared

(W misc) You seem to have already declared the same global once before in the current lexical scope.

'!' allowed only after types `%s`

(F) The '!' is allowed in **pack()** and **unpack()** only after certain types. See ''pack'' in perlfunc.

/ cannot take a count

(F) You had an unpack template indicating a counted-length string, but you have also specified an explicit size for the string. See ''pack'' in perlfunc.

/ must be followed by a, A or Z

(F) You had an unpack template indicating a counted-length string, which must be followed by one of the letters a, A or Z to indicate what sort of string is to be unpacked. See ''pack'' in perlfunc.

/ must be followed by a*, A* or Z*

(F) You had a pack template indicating a counted-length string, Currently the only things that can have their length counted are a*, A* or Z*. See ''pack'' in perlfunc.

/ must follow a numeric type

(F) You had an unpack template that contained a '#', but this did not follow some numeric unpack specification. See ''pack'' in perlfunc.

/%s/: Unrecognized escape \\%c passed through

(W regexp) You used a backslash-character combination which is not recognized by Perl. This combination appears in an interpolated variable or a `'`−delimited regular expression. The character was understood literally.

/%s/: Unrecognized escape \\%c in character class passed through

(W regexp) You used a backslash-character combination which is not recognized by Perl inside character classes. The character was understood literally.

/%s/ should probably be written as ''%s''

(W syntax) You have used a pattern where Perl expected to find a string, as in the first argument to `join`. Perl will treat the true or false result of matching the pattern against `$_` as the string, which is probably not what you had in mind.

%s() called too early to check prototype

(W prototype) You've called a function that has a prototype before the parser saw a definition or declaration for it, and Perl could not check that the call conforms to the prototype. You need to either add an early prototype declaration for the subroutine in question, or move the subroutine definition ahead of the call to get proper prototype checking. Alternatively, if you are certain that you're calling the function correctly, you may put an ampersand before the name to avoid the warning. See perlsub.

`%s` argument is not a HASH or ARRAY element

(F) The argument to **exists()** must be a hash or array element, such as:

```
$foo{$bar}
$ref->{"susie"}[12]
```

%s argument is not a HASH or ARRAY element or slice
(F) The argument to **delete()** must be either a hash or array element, such as:

```
$foo{$bar}
$ref->{"susie"}[12]
```

or a hash or array slice, such as:

```
@foo[$bar, $baz, $xyzzy]
@{$ref->[12]}{"susie", "queue"}
```

%s argument is not a subroutine name
(F) The argument to **exists()** for `exists &sub` must be a subroutine name, and not a subroutine call. `exists &sub()` will generate this error.

%s package attribute may clash with future reserved word: %s
(W reserved) A lowercase attribute name was used that had a package-specific handler. That name might have a meaning to Perl itself some day, even though it doesn't yet. Perhaps you should use a mixed-case attribute name, instead. See attributes.

(in cleanup) %s
(W misc) This prefix usually indicates that a **DESTROY()** method raised the indicated exception. Since destructors are usually called by the system at arbitrary points during execution, and often a vast number of times, the warning is issued only once for any number of failures that would otherwise result in the same message being repeated.

Failure of user callbacks dispatched using the G_KEEPERR flag could also result in this warning. See ''G_KEEPERR'' in perlcall.

<> should be quotes
(F) You wrote `require <file>` when you should have written `require 'file'`.

Attempt to join self
(F) You tried to join a thread from within itself, which is an impossible task. You may be joining the wrong thread, or you may need to move the **join()** to some other thread.

Bad evalled substitution pattern
(F) You've used the /e switch to evaluate the replacement for a substitution, but perl found a syntax error in the code to evaluate, most likely an unexpected right brace '}'.

Bad **realloc()** ignored
(S) An internal routine called **realloc()** on something that had never been **malloc()**ed in the first place. Mandatory, but can be disabled by setting environment variable PERL_BADFREE to 1.

Bareword found in conditional
(W bareword) The compiler found a bareword where it expected a conditional, which often indicates that an || or && was parsed as part of the last argument of the previous construct, for example:

```
open FOO || die;
```

It may also indicate a misspelled constant that has been interpreted as a bareword:

```
use constant TYPO => 1;
if (TYOP) { print "foo" }
```

The `strict` pragma is useful in avoiding such errors.

Binary number > 0b11111111111111111111111111111111 non-portable
(W portable) The binary number you specified is larger than 2**32−1 (4294967295) and therefore non-portable between systems. See perlport for more on portability concerns.

Bit vector size > 32 non-portable
(W portable) Using bit vector sizes larger than 32 is non-portable.

Buffer overflow in prime_env_iter: %s
(W internal) A warning peculiar to VMS. While Perl was preparing to iterate over %ENV, it encountered a logical name or symbol definition which was too long, so it was truncated to the string shown.

Can't check filesystem of script "%s"
(P) For some reason you can't check the filesystem of the script for nosuid.

Can't declare class for non-scalar %s in "%s"
(S) Currently, only scalar variables can declared with a specific class qualifier in a "my" or "our" declaration. The semantics may be extended for other types of variables in future.

Can't declare %s in "%s"
(F) Only scalar, array, and hash variables may be declared as "my" or "our" variables. They must have ordinary identifiers as names.

Can't ignore signal CHLD, forcing to default
(W signal) Perl has detected that it is being run with the SIGCHLD signal (sometimes known as SIGCLD) disabled. Since disabling this signal will interfere with proper determination of exit status of child processes, Perl has reset the signal to its default value. This situation typically indicates that the parent program under which Perl may be running (e.g., cron) is being very careless.

Can't modify non-lvalue subroutine call
(F) Subroutines meant to be used in lvalue context should be declared as such, see "Lvalue subroutines" in perlsub.

Can't read CRTL environ
(S) A warning peculiar to VMS. Perl tried to read an element of %ENV from the CRTL's internal environment array and discovered the array was missing. You need to figure out where your CRTL misplaced its environ or define *PERL_ENV_TABLES* (see perlvms) so that environ is not searched.

Can't remove %s: %s, skipping file
(S) You requested an inplace edit without creating a backup file. Perl was unable to remove the original file to replace it with the modified file. The file was left unmodified.

Can't return %s from lvalue subroutine
(F) Perl detected an attempt to return illegal lvalues (such as temporary or readonly values) from a subroutine used as an lvalue. This is not allowed.

Can't weaken a nonreference
(F) You attempted to weaken something that was not a reference. Only references can be weakened.

Character class [:%s:] unknown
(F) The class in the character class [: :] syntax is unknown. See perlre.

Character class syntax [%s] belongs inside character classes
(W unsafe) The character class constructs [: :], [= =], and [. .] go *inside* character classes, the [] are part of the construct, for example: /[012[:alpha:]345]/. Note that [= =] and [. .] are not currently implemented; they are simply placeholders for future extensions.

Constant is not %s reference
(F) A constant value (perhaps declared using the `use constant` pragma) is being dereferenced, but it amounts to the wrong type of reference. The message indicates the type of reference that was expected. This usually indicates a syntax error in dereferencing the constant value. See "Constant Functions" in perlsub and constant.

constant(%s): %s
(F) The parser found inconsistencies either while attempting to define an overloaded constant, or when trying to find the character name specified in the `\N{...}` escape. Perhaps you forgot to load the corresponding `overload` or `charnames` pragma? See charnames and overload.

CORE::%s is not a keyword
(F) The CORE:: namespace is reserved for Perl keywords.

defined(@array) is deprecated
(D) **defined()** is not usually useful on arrays because it checks for an undefined *scalar* value. If you want to see if the array is empty, just use `if (@array) { # not empty }` for example.

defined(%hash) is deprecated
: (D) **defined()** is not usually useful on hashes because it checks for an undefined *scalar* value. If you want to see if the hash is empty, just use `if (%hash) { # not empty }` for example.

Did not produce a valid header
: See Server error.

(Did you mean ‘‘local’’ instead of ‘‘our’’?)
: (W misc) Remember that ‘‘our’’ does not localize the declared global variable. You have declared it again in the same lexical scope, which seems superfluous.

Document contains no data
: See Server error.

entering effective `%s` failed
: (F) While under the `use filetest` pragma, switching the real and effective uids or gids failed.

false [] range ‘‘%s’’ in regexp
: (W regexp) A character class range must start and end at a literal character, not another character class like `\d` or `[:alpha:]`. The ‘‘-’’ in your false range is interpreted as a literal ‘‘-’’. Consider quoting the ‘‘-’’, ‘‘\-’’. See perlre.

Filehandle `%s` opened only for output
: (W io) You tried to read from a filehandle opened only for writing. If you intended it to be a read/write filehandle, you needed to open it with ‘‘+<’’ or ‘‘+>’’ or ‘‘+>>’’ instead of with ‘‘<’’ or nothing. If you intended only to read from the file, use ‘‘<’’. See ‘‘open’’ in perlfunc.

**flock()** on closed filehandle `%s`
: (W closed) The filehandle you’re attempting to **flock()** got itself closed some time before now. Check your logic flow. **flock()** operates on filehandles. Are you attempting to call **flock()** on a dirhandle by the same name?

Global symbol ‘‘%s’’ requires explicit package name
: (F) You’ve said ‘‘use strict vars’’, which indicates that all variables must either be lexically scoped (using ‘‘my’’), declared beforehand using ‘‘our’’, or explicitly qualified to say which package the global variable is in (using ‘‘::’’).

Hexadecimal number > 0xffffffff non-portable
: (W portable) The hexadecimal number you specified is larger than 2**32-1 (4294967295) and therefore non-portable between systems. See perlport for more on portability concerns.

Ill-formed CRTL environ value ‘‘%s’’
: (W internal) A warning peculiar to VMS. Perl tried to read the CRTL’s internal environ array, and encountered an element without the = delimiter used to separate keys from values. The element is ignored.

Ill-formed message in prime_env_iter: |%s|
: (W internal) A warning peculiar to VMS. Perl tried to read a logical name or CLI symbol definition when preparing to iterate over `%ENV`, and didn’t see the expected delimiter between key and value, so the line was ignored.

Illegal binary digit `%s`
: (F) You used a digit other than 0 or 1 in a binary number.

Illegal binary digit `%s` ignored
: (W digit) You may have tried to use a digit other than 0 or 1 in a binary number. Interpretation of the binary number stopped before the offending digit.

Illegal number of bits in vec
: (F) The number of bits in **vec()** (the third argument) must be a power of two from 1 to 32 (or 64, if your platform supports that).

Integer overflow in `%s` number
: (W overflow) The hexadecimal, octal or binary number you have specified either as a literal or as an argument to **hex()** or **oct()** is too big for your architecture, and has been converted to a floating point number. On a 32-bit architecture the largest hexadecimal, octal or binary number representable without overflow is 0xFFFFFFFF, 037777777777, or 0b11111111111111111111111111111111 respectively. Note that Perl transparently promotes all

numbers to a floating point representation internally — subject to loss of precision errors in subsequent operations.

Invalid %s attribute: %s
: The indicated attribute for a subroutine or variable was not recognized by Perl or by a user-supplied handler. See attributes.

Invalid %s attributes: %s
: The indicated attributes for a subroutine or variable were not recognized by Perl or by a user-supplied handler. See attributes.

invalid [] range ‘‘%s’’ in regexp
: The offending range is now explicitly displayed.

Invalid separator character %s in attribute list
: (F) Something other than a colon or whitespace was seen between the elements of an attribute list. If the previous attribute had a parenthesised parameter list, perhaps that list was terminated too soon. See attributes.

Invalid separator character %s in subroutine attribute list
: (F) Something other than a colon or whitespace was seen between the elements of a subroutine attribute list. If the previous attribute had a parenthesised parameter list, perhaps that list was terminated too soon.

leaving effective %s failed
: (F) While under the `use filetest` pragma, switching the real and effective uids or gids failed.

Lvalue subs returning %s not implemented yet
: (F) Due to limitations in the current implementation, array and hash values cannot be returned in subroutines used in lvalue context. See ‘‘Lvalue subroutines’’ in perlsub.

Method %s not permitted
: See Server error.

Missing %sbrace%s on \N{}
: (F) Wrong syntax of character name literal `\N{charname}` within double-quotish context.

Missing command in piped open
: (W pipe) You used the `open(FH, "| command")` or `open(FH, "command |")` construction, but the command was missing or blank.

Missing name in ‘‘my sub’’
: (F) The reserved syntax for lexically scoped subroutines requires that they have a name with which they can be found.

No %s specified for −%c
: (F) The indicated command line switch needs a mandatory argument, but you haven’t specified one.

No package name allowed for variable %s in ‘‘our’’
: (F) Fully qualified variable names are not allowed in ‘‘our’’ declarations, because that doesn’t make much sense under existing semantics. Such syntax is reserved for future extensions.

No space allowed after −%c
: (F) The argument to the indicated command line switch must follow immediately after the switch, without intervening spaces.

no UTC offset information; assuming local time is UTC
: (S) A warning peculiar to VMS. Perl was unable to find the local timezone offset, so it’s assuming that local system time is equivalent to UTC. If it’s not, define the logical name *SYS$TIMEZONE_DIFFERENTIAL* to translate to the number of seconds which need to be added to UTC to get local time.

Octal number > 037777777777 non-portable
: (W portable) The octal number you specified is larger than 2**32−1 (4294967295) and therefore non-portable between systems. See perlport for more on portability concerns.

  See also perlport for writing portable code.

panic: del_backref
(P) Failed an internal consistency check while trying to reset a weak reference.

panic: kid popen errno read
(F) forked child returned an incomprehensible message about its errno.

panic: magic_killbackrefs
(P) Failed an internal consistency check while trying to reset all weak references to an object.

Parentheses missing around ‘‘%s’’ list
(W parenthesis) You said something like

```
my $foo, $bar = @_;
```

when you meant

```
my ($foo, $bar) = @_;
```

Remember that ‘‘my’’, ‘‘our’’, and ‘‘local’’ bind tighter than comma.

Possible unintended interpolation of `%s` in string
(W ambiguous) It used to be that Perl would try to guess whether you wanted an array interpolated or a literal @. It no longer does this; arrays are now *always* interpolated into strings. This means that if you try something like:

```
print "fred@example.com";
```

and the array `@example` doesn’t exist, Perl is going to print `fred.com`, which is probably not what you wanted. To get a literal `@` sign in a string, put a backslash before it, just as you would to get a literal `$` sign.

Possible Y2K bug: `%s`
(W y2k) You are concatenating the number 19 with another number, which could be a potential Year 2000 problem.

pragma ‘‘attrs’’ is deprecated, use ‘‘sub NAME : ATTRS’’ instead
(W deprecated) You have written something like this:

```
sub doit
{
    use attrs qw(locked);
}
```

You should use the new declaration syntax instead.

```
sub doit : locked
{
    ...
```

The `use attrs` pragma is now obsolete, and is only provided for backward-compatibility. See ‘‘Subroutine Attributes’’ in perlsub.

Premature end of script headers
See Server error.

Repeat count in pack overflows
(F) You can’t specify a repeat count so large that it overflows your signed integers. See ‘‘pack’’ in perlfunc.

Repeat count in unpack overflows
(F) You can’t specify a repeat count so large that it overflows your signed integers. See ‘‘unpack’’ in perlfunc.

**realloc()** of freed memory ignored
(S) An internal routine called **realloc()** on something that had already been freed.

Reference is already weak
(W misc) You have attempted to weaken a reference that is already weak. Doing so has no effect.

setpgrp can't take arguments
: (F) Your system has the **setpgrp()** from BSD 4.2, which takes no arguments, unlike POSIX **setpgid()**, which takes a process ID and process group ID.

Strange *+?{} on zero-length expression
: (W regexp) You applied a regular expression quantifier in a place where it makes no sense, such as on a zero-width assertion. Try putting the quantifier inside the assertion instead. For example, the way to match ''abc'' provided that it is followed by three repetitions of ''xyz'' is `/abc(?=(?:xyz){3})/`, not `/abc(?=xyz){3}/`.

switching effective %s is not implemented
: (F) While under the `use filetest` pragma, we cannot switch the real and effective uids or gids.

This Perl can't reset CRTL environ elements (%s)

This Perl can't set CRTL environ elements (%s=%s)
: (W internal) Warnings peculiar to VMS. You tried to change or delete an element of the CRTL's internal environ array, but your copy of Perl wasn't built with a CRTL that contained the **setenv()** function. You'll need to rebuild Perl with a CRTL that does, or redefine *PERL_ENV_TABLES* (see perlvms) so that the environ array isn't the target of the change to `%ENV` which produced the warning.

Too late to run %s block
: (W void) A CHECK or INIT block is being defined during run time proper, when the opportunity to run them has already passed. Perhaps you are loading a file with `require` or `do` when you should be using `use` instead. Or perhaps you should put the `require` or `do` inside a BEGIN block.

Unknown **open()** mode '%s'
: (F) The second argument of 3−argument **open()** is not among the list of valid modes: `<`, `>`, `>>`, `+<`, `+>`, `+>>`, `-|`, `|-`.

Unknown process %x sent message to prime_env_iter: %s
: (P) An error peculiar to VMS. Perl was reading values for `%ENV` before iterating over it, and someone else stuck a message in the stream of data Perl expected. Someone's very confused, or perhaps trying to subvert Perl's population of `%ENV` for nefarious purposes.

Unrecognized escape \\%c passed through
: (W misc) You used a backslash-character combination which is not recognized by Perl. The character was understood literally.

Unterminated attribute parameter in attribute list
: (F) The lexer saw an opening (left) parenthesis character while parsing an attribute list, but the matching closing (right) parenthesis character was not found. You may need to add (or remove) a backslash character to get your parentheses to balance. See attributes.

Unterminated attribute list
: (F) The lexer found something other than a simple identifier at the start of an attribute, and it wasn't a semicolon or the start of a block. Perhaps you terminated the parameter list of the previous attribute too soon. See attributes.

Unterminated attribute parameter in subroutine attribute list
: (F) The lexer saw an opening (left) parenthesis character while parsing a subroutine attribute list, but the matching closing (right) parenthesis character was not found. You may need to add (or remove) a backslash character to get your parentheses to balance.

Unterminated subroutine attribute list
: (F) The lexer found something other than a simple identifier at the start of a subroutine attribute, and it wasn't a semicolon or the start of a block. Perhaps you terminated the parameter list of the previous attribute too soon.

Value of CLI symbol ''%s'' too long
: (W misc) A warning peculiar to VMS. Perl tried to read the value of an `%ENV` element from a CLI symbol table, and found a resultant string longer than 1024 characters. The return value has been truncated to 1024 characters.

Version number must be a constant number
: (P) The attempt to translate a `use Module n.n LIST` statement into its equivalent `BEGIN` block found an internal inconsistency with the version number.

## New tests

lib/attrs
: Compatibility tests for `sub : attrs` vs the older `use attrs`.

lib/env
: Tests for new environment scalar capability (e.g., `use Env qw($BAR);`).

lib/env−array
: Tests for new environment array capability (e.g., `use Env qw(@PATH);`).

lib/io_const
: IO constants (SEEK_*, _IO*).

lib/io_dir
: Directory-related IO methods (new, read, close, rewind, tied delete).

lib/io_multihomed
: INET sockets with multi-homed hosts.

lib/io_poll
: IO **poll()**.

lib/io_unix
: UNIX sockets.

op/attrs
: Regression tests for `my ($x,@y,%z) : attrs` and <sub : attrs>.

op/filetest
: File test operators.

op/lex_assign
: Verify operations that access pad objects (lexicals and temporaries).

op/exists_sub
: Verify `exists &sub` operations.

## Incompatible Changes

### Perl Source Incompatibilities

Beware that any new warnings that have been added or old ones that have been enhanced are **not** considered incompatible changes.

Since all new warnings must be explicitly requested via the `−w` switch or the `warnings` pragma, it is ultimately the programmer's responsibility to ensure that warnings are enabled judiciously.

CHECK is a new keyword
: All subroutine definitions named CHECK are now special. See `/"Support for CHECK blocks"` for more information.

Treatment of list slices of undef has changed
: There is a potential incompatibility in the behavior of list slices that are comprised entirely of undefined values. See ''Behavior of list slices is more consistent''.

Format of `$English::PERL_VERSION` is different
: The English module now sets `$PERL_VERSION` to $^V (a string value) rather than `$]` (a numeric value). This is a potential incompatibility. Send us a report via perlbug if you are affected by this.

  See ''Improved Perl version numbering system'' for the reasons for this change.

Literals of the form `1.2.3` parse differently
: Previously, numeric literals with more than one dot in them were interpreted as a floating point number concatenated with one or more numbers. Such ''numbers'' are now parsed as strings composed of the specified ordinals.

  For example, `print 97.98.99` used to output `97.9899` in earlier versions, but now prints

`abc`.

See ‘‘Support for strings represented as a vector of ordinals’’.

Possibly changed pseudo-random number generator
: Perl programs that depend on reproducing a specific set of pseudo-random numbers may now produce different output due to improvements made to the **rand()** builtin. You can use `sh Configure -Drandfunc=rand` to obtain the old behavior.

  See ‘‘Better pseudo-random number generator’’.

Hashing function for hash keys has changed
: Even though Perl hashes are not order preserving, the apparently random order encountered when iterating on the contents of a hash is actually determined by the hashing algorithm used. Improvements in the algorithm may yield a random order that is **different** from that of previous versions, especially when iterating on hashes.

  See ‘‘Better worst-case behavior of hashes’’ for additional information.

`undef` fails on read only values
: Using the `undef` operator on a readonly value (such as `$1`) has the same effect as assigning `undef` to the readonly value — it throws an exception.

Close-on-exec bit may be set on pipe and socket handles
: Pipe and socket handles are also now subject to the close-on-exec behavior determined by the special variable $^F.

  See ‘‘More consistent close-on-exec behavior’’.

Writing `''$$1''` to mean `''${$}1''` is unsupported
: Perl 5.004 deprecated the interpretation of `$$1` and similar within interpolated strings to mean `$$ . "1"`, but still allowed it.

  In Perl 5.6.0 and later, `"$$1"` always means `"${$1}"`.

**delete()**, **each()**, **values()** and `\(%h)`
: operate on aliases to values, not copies

  **delete()**, **each()**, **values()** and hashes (e.g. `\(%h)`) in a list context return the actual values in the hash, instead of copies (as they used to in earlier versions). Typical idioms for using these constructs copy the returned values, but this can make a significant difference when creating references to the returned values. Keys in the hash are still returned as copies when iterating on a hash.

  See also ‘‘**delete()**, **each()**, **values()** and hash iteration are faster’’.

vec(EXPR,OFFSET,BITS) enforces powers-of-two BITS
: **vec()** generates a run-time error if the BITS argument is not a valid power-of-two integer.

Text of some diagnostic output has changed
: Most references to internal Perl operations in diagnostics have been changed to be more descriptive. This may be an issue for programs that may incorrectly rely on the exact text of diagnostics for proper functioning.

%@ has been removed
: The undocumented special variable `%@` that used to accumulate ‘‘background’’ errors (such as those that happen in **DESTROY()**) has been removed, because it could potentially result in memory leaks.

Parenthesized **not()** behaves like a list operator
: The `not` operator now falls under the ‘‘if it looks like a function, it behaves like a function’’ rule.

  As a result, the parenthesized form can be used with `grep` and `map`. The following construct used to be a syntax error before, but it works as expected now:

```
grep not($_), @things;
```

  On the other hand, using `not` with a literal list slice may not work. The following previously allowed construct:

```
print not (1,2,3)[0];
```

needs to be written with additional parentheses now:

```
print not((1,2,3)[0]);
```

The behavior remains unaffected when `not` is not followed by parentheses.

Semantics of bareword prototype `(*)` have changed

The semantics of the bareword prototype `*` have changed. Perl 5.005 always coerced simple scalar arguments to a typeglob, which wasn't useful in situations where the subroutine must distinguish between a simple scalar and a typeglob. The new behavior is to not coerce bareword arguments to a typeglob. The value will always be visible as either a simple scalar or as a reference to a typeglob.

See ''More functional bareword prototype (*)''.

Semantics of bit operators may have changed on 64−bit platforms

If your platform is either natively 64−bit or if Perl has been configured to used 64−bit integers, i.e., `$Config{ivsize}` is 8, there may be a potential incompatibility in the behavior of bitwise numeric operators (& | ^ ~ << >>). These operators used to strictly operate on the lower 32 bits of integers in previous versions, but now operate over the entire native integral width. In particular, note that unary `~` will produce different results on platforms that have different `$Config{ivsize}`. For portability, be sure to mask off the excess bits in the result of unary `~`, e.g., `~$x & 0xffffffff`.

See ''Bit operators support full native integer width''.

More builtins taint their results

As described in ''Improved security features'', there may be more sources of taint in a Perl program.

To avoid these new tainting behaviors, you can build Perl with the Configure option `−Accflags=−DINCOMPLETE_TAINTS`. Beware that the ensuing perl binary may be insecure.

### C Source Incompatibilities

`PERL_POLLUTE`

Release 5.005 grandfathered old global symbol names by providing preprocessor macros for extension source compatibility. As of release 5.6.0, these preprocessor definitions are not available by default. You need to explicitly compile perl with `−DPERL_POLLUTE` to get these definitions. For extensions still using the old symbols, this option can be specified via MakeMaker:

```
perl Makefile.PL POLLUTE=1
```

`PERL_IMPLICIT_CONTEXT`

This new build option provides a set of macros for all API functions such that an implicit interpreter/thread context argument is passed to every API function. As a result of this, something like `sv_setsv(foo,bar)` amounts to a macro invocation that actually translates to something like `Perl_sv_setsv(my_perl,foo,bar)`. While this is generally expected to not have any significant source compatibility issues, the difference between a macro and a real function call will need to be considered.

This means that there **is** a source compatibility issue as a result of this if your extensions attempt to use pointers to any of the Perl API functions.

Note that the above issue is not relevant to the default build of Perl, whose interfaces continue to match those of prior versions (but subject to the other options described here).

See ''Background and PERL_IMPLICIT_CONTEXT'' in perlguts for detailed information on the ramifications of building Perl with this option.

```
NOTE: PERL_IMPLICIT_CONTEXT is automatically enabled whenever Perl is buil
with one of −Dusethreads, −Dusemultiplicity, or both.  It is not
intended to be enabled by users at this time.
```

PERL_POLLUTE_MALLOC

Enabling Perl's malloc in release 5.005 and earlier caused the namespace of the system's malloc family of functions to be usurped by the Perl versions, since by default they used the same names. Besides causing problems on platforms that do not allow these functions to be cleanly replaced, this also meant that the system versions could not be called in programs that used Perl's malloc. Previous versions of Perl have allowed this behaviour to be suppressed with the HIDEMYMALLOC and EMBEDMYMALLOC preprocessor definitions.

As of release 5.6.0, Perl's malloc family of functions have default names distinct from the system versions. You need to explicitly compile perl with `-DPERL_POLLUTE_MALLOC` to get the older behaviour. HIDEMYMALLOC and EMBEDMYMALLOC have no effect, since the behaviour they enabled is now the default.

Note that these functions do **not** constitute Perl's memory allocation API. See ''Memory Allocation'' in perlguts for further information about that.

### Compatible C Source API Changes

PATCHLEVEL is now PERL_VERSION

The cpp macros `PERL_REVISION`, `PERL_VERSION`, and `PERL_SUBVERSION` are now available by default from perl.h, and reflect the base revision, patchlevel, and subversion respectively. `PERL_REVISION` had no prior equivalent, while `PERL_VERSION` and `PERL_SUBVERSION` were previously available as `PATCHLEVEL` and `SUBVERSION`.

The new names cause less pollution of the **cpp** namespace and reflect what the numbers have come to stand for in common practice. For compatibility, the old names are still supported when *patchlevel.h* is explicitly included (as required before), so there is no source incompatibility from the change.

### Binary Incompatibilities

In general, the default build of this release is expected to be binary compatible for extensions built with the 5.005 release or its maintenance versions. However, specific platforms may have broken binary compatibility due to changes in the defaults used in hints files. Therefore, please be sure to always check the platform-specific README files for any notes to the contrary.

The usethreads or usemultiplicity builds are **not** binary compatible with the corresponding builds in 5.005.

On platforms that require an explicit list of exports (AIX, OS/2 and Windows, among others), purely internal symbols such as parser functions and the run time opcodes are not exported by default. Perl 5.005 used to export all functions irrespective of whether they were considered part of the public API or not.

For the full list of public API functions, see perlapi.

## Known Problems

### Thread test failures

The subtests 19 and 20 of lib/thr5005.t test are known to fail due to fundamental problems in the 5.005 threading implementation. These are not new failures — Perl 5.005_0x has the same bugs, but didn't have these tests.

### EBCDIC platforms not supported

In earlier releases of Perl, EBCDIC environments like OS390 (also known as Open Edition MVS) and VM-ESA were supported. Due to changes required by the UTF-8 (Unicode) support, the EBCDIC platforms are not supported in Perl 5.6.0.

### In 64-bit HP-UX the lib/io_multihomed test may hang

The lib/io_multihomed test may hang in HP-UX if Perl has been configured to be 64-bit. Because other 64-bit platforms do not hang in this test, HP-UX is suspect. All other tests pass in 64-bit HP-UX. The test attempts to create and connect to ''multihomed'' sockets (sockets which have multiple IP addresses).

### NEXTSTEP 3.3 POSIX test failure

In NEXTSTEP 3.3p2 the implementation of the **strftime**(3) in the operating system libraries is buggy: the `%j` format numbers the days of a month starting from zero, which, while being logical to programmers, will cause the subtests 19 to 27 of the lib/posix test may fail.

### Tru64 (aka Digital UNIX, aka DEC OSF/1) lib/sdbm test failure with gcc

If compiled with gcc 2.95 the lib/sdbm test will fail (dump core). The cure is to use the vendor cc, it comes with the operating system and produces good code.

### UNICOS/mk CC failures during Configure run

In UNICOS/mk the following errors may appear during the Configure run:

```
Guessing which symbols your C compiler and preprocessor define...
CC-20 cc: ERROR File = try.c, Line = 3
...
  bad switch yylook 79bad switch yylook 79bad switch yylook 79bad switch y
...
4 errors detected in the compilation of "try.c".
```

The culprit is the broken awk of UNICOS/mk. The effect is fortunately rather mild: Perl itself is not adversely affected by the error, only the h2ph utility coming with Perl, and that is rather rarely needed these days.

### Arrow operator and arrays

When the left argument to the arrow operator `->` is an array, or the `scalar` operator operating on an array, the result of the operation must be considered erroneous. For example:

```
@x->[2]
scalar(@x)->[2]
```

These expressions will get run-time errors in some future release of Perl.

### Experimental features

As discussed above, many features are still experimental. Interfaces and implementation of these features are subject to change, and in extreme cases, even subject to removal in some future release of Perl. These features include the following:

Threads
Unicode
64-bit support
Lvalue subroutines
Weak references
The pseudo-hash data type
The Compiler suite
Internal implementation of file globbing
The DB module
The regular expression code constructs:
`(?{ code })` and `(??{ code })`

## Obsolete Diagnostics

Character class syntax [: :] is reserved for future extensions

(W) Within regular expression character classes ([]) the syntax beginning with "[:" and ending with ":]" is reserved for future extensions. If you need to represent those character sequences inside a regular expression character class, just quote the square brackets with the backslash: "\[:" and ":\]".

Ill-formed logical name |%s| in prime_env_iter

(W) A warning peculiar to VMS. A logical name was encountered when preparing to iterate over `%ENV` which violates the syntactic rules governing logical names. Because it cannot be translated normally, it is skipped, and will not appear in `%ENV`. This may be a benign occurrence, as some software packages might directly modify logical name tables and introduce nonstandard names, or it may indicate that a logical name table has been corrupted.

In string, @%s now must be written as \@%s

The description of this error used to say:

```
(Someday it will simply assume that an unbackslashed @
 interpolates an array.)
```

That day has come, and this fatal error has been removed. It has been replaced by a non-fatal warning instead. See "Arrays now always interpolate into double-quoted strings" for details.

Probable precedence problem on %s

(W) The compiler found a bareword where it expected a conditional, which often indicates that an || or && was parsed as part of the last argument of the previous construct, for example:

```
open FOO || die;
```

regexp too big

(F) The current implementation of regular expressions uses shorts as address offsets within a string. Unfortunately this means that if the regular expression compiles to longer than 32767, it'll blow up. Usually when you want a regular expression this big, there is a better way to do it with multiple statements. See perlre.

Use of "$$<digit>" to mean "${$}<digit>" is deprecated

(D) Perl versions before 5.004 misinterpreted any type marker followed by "$" and a digit. For example, "$$0" was incorrectly taken to mean "${$}0" instead of "${$0}". This bug is (mostly) fixed in Perl 5.004.

However, the developers of Perl 5.004 could not fix this bug completely, because at least two widely-used modules depend on the old meaning of "$$0" in a string. So Perl 5.004 still interprets "$$<digit>" in the old (broken) way inside strings; but it generates this message as a warning. And in Perl 5.005, this special treatment will cease.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup. There may also be information at http://www.perl.com/perl/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## HISTORY

Written by Gurusamy Sarathy <*gsar@activestate.com*>, with many contributions from The Perl Porters.

Send omissions or corrections to <*perlbug@perl.org*>.

## NAME

perl581delta − what is new for perl v5.8.1

## DESCRIPTION

This document describes differences between the 5.8.0 release and the 5.8.1 release.

If you are upgrading from an earlier release such as 5.6.1, first read the perl58delta, which describes differences between 5.6.0 and 5.8.0.

In case you are wondering about 5.6.1, it was bug-fix-wise rather identical to the development release 5.7.1. Confused? This timeline hopefully helps a bit: it lists the new major releases, their maintenance releases, and the development releases.

```
New     Maintenance  Development

5.6.0                             2000-Mar-22
                     5.7.0        2000-Sep-02
        5.6.1                     2001-Apr-08
                     5.7.1        2001-Apr-09
                     5.7.2        2001-Jul-13
                     5.7.3        2002-Mar-05
5.8.0                             2002-Jul-18
        5.8.1                     2003-Sep-25
```

## Incompatible Changes

### Hash Randomisation

Mainly due to security reasons, the ''random ordering'' of hashes has been made even more random. Previously while the order of hash elements from **keys()**, **values()**, and **each()** was essentially random, it was still repeatable. Now, however, the order varies between different runs of Perl.

**Perl has never guaranteed any ordering of the hash keys**, and the ordering has already changed several times during the lifetime of Perl 5. Also, the ordering of hash keys has always been, and continues to be, affected by the insertion order.

The added randomness may affect applications.

One possible scenario is when output of an application has included hash data. For example, if you have used the Data::Dumper module to dump data into different files, and then compared the files to see whether the data has changed, now you will have false positives since the order in which hashes are dumped will vary. In general the cure is to sort the keys (or the values); in particular for Data::Dumper to use the `Sortkeys` option. If some particular order is really important, use tied hashes: for example the Tie::IxHash module which by default preserves the order in which the hash elements were added.

More subtle problem is reliance on the order of ''global destruction''. That is what happens at the end of execution: Perl destroys all data structures, including user data. If your destructors (the DESTROY subroutines) have assumed any particular ordering to the global destruction, there might be problems ahead. For example, in a destructor of one object you cannot assume that objects of any other class are still available, unless you hold a reference to them. If the environment variable PERL_DESTRUCT_LEVEL is set to a non-zero value, or if Perl is exiting a spawned thread, it will also destruct the ordinary references and the symbol tables that are no longer in use. You can't call a class method or an ordinary function on a class that has been collected that way.

The hash randomisation is certain to reveal hidden assumptions about some particular ordering of hash elements, and outright bugs: it revealed a few bugs in the Perl core and core modules.

To disable the hash randomisation in runtime, set the environment variable PERL_HASH_SEED to 0 (zero) before running Perl (for more information see ''PERL_HASH_SEED'' in perlrun), or to disable the feature completely in compile time, compile with `-DNO_HASH_SEED` (see *INSTALL*).

See ''Algorithmic Complexity Attacks'' in perlsec for the original rationale behind this change.

### UTF-8 On Filehandles No Longer Activated By Locale

In Perl 5.8.0 all filehandles, including the standard filehandles, were implicitly set to be in Unicode UTF-8 if the locale settings indicated the use of UTF-8. This feature caused too many problems, so the feature was turned off and redesigned: see ''Core Enhancements''.

### Single-number v-strings are no longer v-strings before "=>"

The version strings or v-strings (see "Version Strings" in perldata) feature introduced in Perl 5.6.0 has been a source of some confusion — especially when the user did not want to use it, but Perl thought it knew better. Especially troublesome has been the feature that before a "=>" a version string (a "v" followed by digits) has been interpreted as a v-string instead of a string literal. In other words:

```
%h = ( v65 => 42 );
```

has meant since Perl 5.6.0

```
%h = ( 'A' => 42 );
```

(at least in platforms of ASCII progeny) Perl 5.8.1 restores the more natural interpretation

```
%h = ( 'v65' => 42 );
```

The multi-number v-strings like v65.66 and 65.66.67 still continue to be v-strings in Perl 5.8.

### (Win32) The -C Switch Has Been Repurposed

The -C switch has changed in an incompatible way. The old semantics of this switch only made sense in Win32 and only in the "use utf8" universe in 5.6.x releases, and do not make sense for the Unicode implementation in 5.8.0. Since this switch could not have been used by anyone, it has been repurposed. The behavior that this switch enabled in 5.6.x releases may be supported in a transparent, data-dependent fashion in a future release.

For the new life of this switch, see "UTF-8 no longer default under UTF-8 locales", and "-C" in perlrun.

### (Win32) The /d Switch Of cmd.exe

Perl 5.8.1 uses the /d switch when running the cmd.exe shell internally for **system()**, backticks, and when opening pipes to external programs. The extra switch disables the execution of AutoRun commands from the registry, which is generally considered undesirable when running external programs. If you wish to retain compatibility with the older behavior, set PERL5SHELL in your environment to `cmd /x/c`.

## Core Enhancements

### UTF-8 no longer default under UTF-8 locales

In Perl 5.8.0 many Unicode features were introduced. One of them was found to be of more nuisance than benefit: the automagic (and silent) "UTF-8-ification" of filehandles, including the standard filehandles, if the user's locale settings indicated use of UTF-8.

For example, if you had `en_US.UTF-8` as your locale, your STDIN and STDOUT were automatically "UTF-8", in other words an implicit binmode(..., ":utf8") was made. This meant that trying to print, say, **chr**(0xff), ended up printing the bytes 0xc3 0xbf. Hardly what you had in mind unless you were aware of this feature of Perl 5.8.0. The problem is that the vast majority of people weren't: for example in RedHat releases 8 and 9 the **default** locale setting is UTF-8, so all RedHat users got UTF-8 filehandles, whether they wanted it or not. The pain was intensified by the Unicode implementation of Perl 5.8.0 (still) having nasty bugs, especially related to the use of s/// and tr///. (Bugs that have been fixed in 5.8.1)

Therefore a decision was made to backtrack the feature and change it from implicit silent default to explicit conscious option. The new Perl command line option `-C` and its counterpart environment variable PERL_UNICODE can now be used to control how Perl and Unicode interact at interfaces like I/O and for example the command line arguments. See "-C" in perlrun and "PERL_UNICODE" in perlrun for more information.

### Unsafe signals again available

In Perl 5.8.0 the so-called "safe signals" were introduced. This means that Perl no longer handles signals immediately but instead "between opcodes", when it is safe to do so. The earlier immediate handling easily could corrupt the internal state of Perl, resulting in mysterious crashes.

However, the new safer model has its problems too. Because now an opcode, a basic unit of Perl execution, is never interrupted but instead let to run to completion, certain operations that can take a long time now really do take a long time. For example, certain network operations have their own blocking and timeout mechanisms, and being able to interrupt them immediately would be nice.

Therefore perl 5.8.1 introduces a "backdoor" to restore the pre-5.8.0 (pre-5.7.3, really) signal behaviour. Just set the environment variable PERL_SIGNALS to `unsafe`, and the old immediate (and

unsafe) signal handling behaviour returns. See ''PERL_SIGNALS'' in perlrun and ''Deferred Signals (Safe Signals)'' in perlipc.

In completely unrelated news, you can now use safe signals with POSIX::SigAction. See ''POSIX::SigAction'' in POSIX.

### Tied Arrays with Negative Array Indices

Formerly, the indices passed to `FETCH`, `STORE`, `EXISTS`, and `DELETE` methods in tied array class were always non-negative. If the actual argument was negative, Perl would call FETCHSIZE implicitly and add the result to the index before passing the result to the tied array method. This behaviour is now optional. If the tied array class contains a package variable named `$NEGATIVE_INDICES` which is set to a true value, negative values will be passed to `FETCH`, `STORE`, `EXISTS`, and `DELETE` unchanged.

### local ${$x}

The syntaxes

```
local ${$x}
local @{$x}
local %{$x}
```

now do localise variables, given that the `$x` is a valid variable name.

### Unicode Character Database 4.0.0

The copy of the Unicode Character Database included in Perl 5.8 has been updated to 4.0.0 from 3.2.0. This means for example that the Unicode character properties are as in Unicode 4.0.0.

### Deprecation Warnings

There is one new feature deprecation. Perl 5.8.0 forgot to add some deprecation warnings, these warnings have now been added. Finally, a reminder of an impending feature removal.

*(Reminder) Pseudo-hashes are deprecated (really)*

Pseudo-hashes were deprecated in Perl 5.8.0 and will be removed in Perl 5.10.0, see perl58delta for details. Each attempt to access pseudo-hashes will trigger the warning `Pseudo-hashes are deprecated`. If you really want to continue using pseudo-hashes but not to see the deprecation warnings, use:

```
no warnings 'deprecated';
```

Or you can continue to use the fields pragma, but please don't expect the data structures to be pseudohashes any more.

*(Reminder) 5.005-style threads are deprecated (really)*

5.005-style threads (activated by `use Thread;`) were deprecated in Perl 5.8.0 and will be removed after Perl 5.8, see perl58delta for details. Each 5.005-style thread creation will trigger the warning `5.005 threads are deprecated`. If you really want to continue using the 5.005 threads but not to see the deprecation warnings, use:

```
no warnings 'deprecated';
```

*(Reminder) The $* variable is deprecated (really)*

The `$*` variable controlling multi-line matching has been deprecated and will be removed after 5.8. The variable has been deprecated for a long time, and a deprecation warning `Use of $* is deprecated` is given, now the variable will just finally be removed. The functionality has been supplanted by the `/s` and `/m` modifiers on pattern matching. If you really want to continue using the `$*`-variable but not to see the deprecation warnings, use:

```
no warnings 'deprecated';
```

### Miscellaneous Enhancements

`map` in void context is no longer expensive. `map` is now context aware, and will not construct a list if called in void context.

If a socket gets closed by the server while printing to it, the client now gets a SIGPIPE. While this new feature was not planned, it fell naturally out of PerlIO changes, and is to be considered an accidental feature.

PerlIO::get_layers(FH) returns the names of the PerlIO layers active on a filehandle.

PerlIO::via layers can now have an optional UTF8 method to indicate whether the layer wants to ‘‘auto-:utf8’’ the stream.

**utf8::is_utf8()** has been added as a quick way to test whether a scalar is encoded internally in UTF-8 (Unicode).

## Modules and Pragmata

### Updated Modules And Pragmata

The following modules and pragmata have been updated since Perl 5.8.0:

base

B::Bytecode
: In much better shape than it used to be. Still far from perfect, but maybe worth a try.

B::Concise

B::Deparse

Benchmark
: An optional feature, `:hireswallclock`, now allows for high resolution wall clock times (uses Time::HiRes).

ByteLoader
: See B::Bytecode.

bytes
: Now has bytes::substr.

CGI

charnames
: One can now have custom character name aliases.

CPAN
: There is now a simple command line frontend to the CPAN.pm module called *cpan*.

Data::Dumper
: A new option, Pair, allows choosing the separator between hash keys and values.

DB_File

Devel::PPPort

Digest::MD5

Encode
: Significant updates on the encoding pragma functionality (tr/// and the DATA filehandle, formats).

  If a filehandle has been marked as to have an encoding, unmappable characters are detected already during input, not later (when the corrupted data is being used).

  The ISO 8859-6 conversion table has been corrected (the 0x30..0x39 erroneously mapped to U+0660..U+0669, instead of U+0030..U+0039). The GSM 03.38 conversion did not handle escape sequences correctly. The UTF-7 encoding has been added (making Encode feature-complete with Unicode::String).

fields

libnet

Math::BigInt
: A lot of bugs have been fixed since v1.60, the version included in Perl v5.8.0. Especially noteworthy are the bug in Calc that caused div and mod to fail for some large values, and the fixes to the handling of bad inputs.

  Some new features were added, e.g. the **broot()** method, you can now pass parameters to **config()** to change some settings at runtime, and it is now possible to trap the creation of NaN and infinity.

  As usual, some optimizations took place and made the math overall a tad faster. In some cases, quite a lot faster, actually. Especially alternative libraries like Math::BigInt::GMP benefit from this. In addition, a lot of the quite clunky routines like **fsqrt()** and **flog()** are now much much faster.

MIME::Base64

NEXT
: Diamond inheritance now works.

Net::Ping
PerlIO::scalar
: Reading from non-string scalars (like the special variables, see perlvar) now works.

podlators
Pod::LaTeX
PodParsers
Pod::Perldoc
: Complete rewrite. As a side-effect, no longer refuses to startup when run by root.

Scalar::Util
: New utilities: refaddr, isvstring, looks_like_number, set_prototype.

Storable
: Can now store code references (via B::Deparse, so not foolproof).

strict
: Earlier versions of the strict pragma did not check the parameters implicitly passed to its ''import'' (use) and ''unimport'' (no) routine. This caused the false idiom such as:

```
use strict qw(@ISA);
@ISA = qw(Foo);
```

This however (probably) raised the false expectation that the strict refs, vars and subs were being enforced (and that `@ISA` was somehow ''declared''). But the strict refs, vars, and subs are **not** enforced when using this false idiom.

Starting from Perl 5.8.1, the above **will** cause an error to be raised. This may cause programs which used to execute seemingly correctly without warnings and errors to fail when run under 5.8.1. This happens because

```
use strict qw(@ISA);
```

will now fail with the error:

```
Unknown 'strict' tag(s) '@ISA'
```

The remedy to this problem is to replace this code with the correct idiom:

```
use strict;
use vars qw(@ISA);
@ISA = qw(Foo);
```

Term::ANSIcolor
Test::Harness
: Now much more picky about extra or missing output from test scripts.

Test::More
Test::Simple
Text::Balanced
Time::HiRes
: Use of **nanosleep()**, if available, allows mixing subsecond sleeps with alarms.

threads
: Several fixes, for example for **join()** problems and memory leaks. In some platforms (like Linux) that use glibc the minimum memory footprint of one ithread has been reduced by several hundred kilobytes.

threads::shared
: Many memory leaks have been fixed.

Unicode::Collate
Unicode::Normalize
Win32::GetFolderPath

Win32::GetOSVersion
: Now returns extra information.

## Utility Changes

The `h2xs` utility now produces a more modern layout: *Foo-Bar/lib/Foo/Bar.pm* instead of *Foo/Bar/Bar.pm*. Also, the boilerplate test is now called *t/Foo-Bar.t* instead of *t/1.t*.

The Perl debugger (*lib/perl5db.pl*) has now been extensively documented and bugs found while documenting have been fixed.

`perldoc` has been rewritten from scratch to be more robust and feature rich.

`perlcc -B` works now at least somewhat better, while `perlcc -c` is rather more broken. (The Perl compiler suite as a whole continues to be experimental.)

## New Documentation

perl573delta has been added to list the differences between the (now quite obsolete) development releases 5.7.2 and 5.7.3.

perl58delta has been added: it is the perldelta of 5.8.0, detailing the differences between 5.6.0 and 5.8.0.

perlartistic has been added: it is the Artistic License in pod format, making it easier for modules to refer to it.

perlcheat has been added: it is a Perl cheat sheet.

perlgpl has been added: it is the GNU General Public License in pod format, making it easier for modules to refer to it.

perlmacosx has been added to tell about the installation and use of Perl in Mac OS X.

perlos400 has been added to tell about the installation and use of Perl in OS/400 PASE.

perlreref has been added: it is a regular expressions quick reference.

## Installation and Configuration Improvements

The Unix standard Perl location, */usr/bin/perl*, is no longer overwritten by default if it exists. This change was very prudent because so many Unix vendors already provide a */usr/bin/perl*, but simultaneously many system utilities may depend on that exact version of Perl, so better not to overwrite it.

One can now specify installation directories for site and vendor man and HTML pages, and site and vendor scripts. See *INSTALL*.

One can now specify a destination directory for Perl installation by specifying the DESTDIR variable for `make install`. (This feature is slightly different from the previous `Configure -Dinstallprefix=...`.) See *INSTALL*.

gcc versions 3.x introduced a new warning that caused a lot of noise during Perl compilation: `gcc -Ialreadyknowndirectory (warning: changing search order)`. This warning has now been avoided by Configure weeding out such directories before the compilation.

One can now build subsets of Perl core modules by using the Configure flags `-Dnoextensions=...` and `-Donlyextensions=...`, see *INSTALL*.

### Platform-specific enhancements

In Cygwin Perl can now be built with threads (`Configure -Duseithreads`). This works with both Cygwin 1.3.22 and Cygwin 1.5.3.

In newer FreeBSD releases Perl 5.8.0 compilation failed because of trying to use *malloc.h*, which in FreeBSD is just a dummy file, and a fatal error to even try to use. Now *malloc.h* is not used.

Perl is now known to build also in Hitachi HI-UXMPP.

Perl is now known to build again in LynxOS.

Mac OS X now installs with Perl version number embedded in installation directory names for easier upgrading of user-compiled Perl, and the installation directories in general are more standard. In other words, the default installation no longer breaks the Apple-provided Perl. On the other hand, with `Configure -Dprefix=/usr` you can now really replace the Apple-supplied Perl (**please be careful**).

Mac OS X now builds Perl statically by default. This change was done mainly for faster startup times. The Apple-provided Perl is still dynamically linked and shared, and you can enable the sharedness for your own Perl builds by `Configure -Duseshrplib`.

Perl has been ported to IBM's OS/400 PASE environment. The best way to build a Perl for PASE is to use an AIX host as a cross-compilation environment. See README.os400.

Yet another cross-compilation option has been added: now Perl builds on OpenZaurus, a Linux distribution based on Mandrake + Embedix for the Sharp Zaurus PDA. See the Cross/README file.

Tru64 when using gcc 3 drops the optimisation for *toke.c* to `-O2` because of gigantic memory use with the default `-O3`.

Tru64 can now build Perl with the newer Berkeley DBs.

Building Perl on WinCE has been much enhanced, see *README.ce* and *README.perlce*.

## Selected Bug Fixes

### Closures, eval and lexicals

There have been many fixes in the area of anonymous subs, lexicals and closures. Although this means that Perl is now more ''correct'', it is possible that some existing code will break that happens to rely on the faulty behaviour. In practice this is unlikely unless your code contains a very complex nesting of anonymous subs, evals and lexicals.

### Generic fixes

If an input filehandle is marked `:utf8` and Perl sees illegal UTF-8 coming in when doing `<FH>`, if warnings are enabled a warning is immediately given - instead of being silent about it and Perl being unhappy about the broken data later. (The `:encoding(utf8)` layer also works the same way.)

binmode(SOCKET, ':utf8') only worked on the input side, not on the output side of the socket. Now it works both ways.

For threaded Perls certain system database functions like **getpwent()** and **getgrent()** now grow their result buffer dynamically, instead of failing. This means that at sites with lots of users and groups the functions no longer fail by returning only partial results.

Perl 5.8.0 had accidentally broken the capability for users to define their own uppercase<->lowercase Unicode mappings (as advertised by the Camel). This feature has been fixed and is also documented better.

In 5.8.0 this

```
$some_unicode .= <FH>;
```

didn't work correctly but instead corrupted the data. This has now been fixed.

Tied methods like FETCH etc. may now safely access tied values, i.e. resulting in a recursive call to FETCH etc. Remember to break the recursion, though.

At startup Perl blocks the SIGFPE signal away since there isn't much Perl can do about it. Previously this blocking was in effect also for programs executed from within Perl. Now Perl restores the original SIGFPE handling routine, whatever it was, before running external programs.

Linenumbers in Perl scripts may now be greater than 65536, or 2**16. (Perl scripts have always been able to be larger than that, it's just that the linenumber for reported errors and warnings have ''wrapped around''.) While scripts that large usually indicate a need to rethink your code a bit, such Perl scripts do exist, for example as results from generated code. Now linenumbers can go all the way to 4294967296, or 2**32.

### Platform-specific fixes

Linux

- Setting `$0` works again (with certain limitations that Perl cannot do much about: see ''$0'' in perlvar)

HP-UX

- Setting `$0` now works.

VMS

- Configuration now tests for the presence of `poll()`, and IO::Poll now uses the vendor-supplied function if detected.
- A rare access violation at Perl start-up could occur if the Perl image was installed with privileges or if there was an identifier with the subsystem attribute set in the process's rightslist. Either of these circumstances triggered tainting code that contained a pointer bug. The faulty pointer arithmetic has been fixed.
- The length limit on values (not keys) in the `%ENV` hash has been raised from 255 bytes to 32640 bytes (except when the PERL_ENV_TABLES setting overrides the default use of logical names for `%ENV`). If it is necessary to access these long values from outside Perl, be aware that they are implemented using search list logical names that store the value in pieces, each 255−byte piece (up to 128 of them) being an element in the search list. When doing a lookup in `%ENV` from within Perl, the elements are combined into a single value. The existing VMS-specific ability to access individual elements of a search list logical name via the `$ENV{'foo;N'}` syntax (where N is the search list index) is unimpaired.
- The piping implementation now uses local rather than global DCL symbols for inter-process communication.
- File::Find could become confused when navigating to a relative directory whose name collided with a logical name. This problem has been corrected by adding directory syntax to relative path names, thus preventing logical name translation.

Win32

- A memory leak in the **fork()** emulation has been fixed.
- The return value of the **ioctl()** built-in function was accidentally broken in 5.8.0. This has been corrected.
- The internal message loop executed by perl during blocking operations sometimes interfered with messages that were external to Perl. This often resulted in blocking operations terminating prematurely or returning incorrect results, when Perl was executing under environments that could generate Windows messages. This has been corrected.
- Pipes and sockets are now automatically in binary mode.
- The four-argument form of **select()** did not preserve $! (errno) properly when there were errors in the underlying call. This is now fixed.
- The ''CR CR LF'' problem of has been fixed, binmode(FH, ':crlf'') is now effectively a no-op.

## New or Changed Diagnostics

All the warnings related to **pack()** and **unpack()** were made more informative and consistent.

### Changed "A thread exited while %d threads were running"

The old version

```
A thread exited while %d other threads were still running
```

was misleading because the ''other'' included also the thread giving the warning.

### Removed "Attempt to clear a restricted hash"

It is not illegal to clear a restricted hash, so the warning was removed.

### New "Illegal declaration of anonymous subroutine"

You must specify the block of code for `sub`.

### Changed "Invalid range "%s" in transliteration operator"

The old version

```
Invalid [] range "%s" in transliteration operator
```

was simply wrong because there are no ''[] ranges'' in tr///.

### New "Missing control char name in \c"

Self-explanatory.

### New "Newline in left-justified string for %s"

The padding spaces would appear after the newline, which is probably not what you had in mind.

**New "Possible precedence problem on bitwise %c operator"**
If you think this

```
$x & $y == 0
```

tests whether the bitwise AND of $x and $y is zero, you will like this warning.

**New "Pseudo-hashes are deprecated"**
This warning should have been already in 5.8.0, since they are.

**New "read() on %s filehandle %s"**
You cannot **read()** (or **sysread()**) from a closed or unopened filehandle.

**New "5.005 threads are deprecated"**
This warning should have been already in 5.8.0, since they are.

**New "Tied variable freed while still in use"**
Something pulled the plug on a live tied variable, Perl plays safe by bailing out.

**New "To%s: illegal mapping '%s'"**
An illegal user-defined Unicode casemapping was specified.

**New "Use of freed value in iteration"**
Something modified the values being iterated over. This is not good.

## Changed Internals

These news matter to you only if you either write XS code or like to know about or hack Perl internals (using Devel::Peek or any of the B:: modules counts), or like to run Perl with the -D option.

The embedding examples of perlembed have been reviewed to be up to date and consistent: for example, the correct use of **PERL_SYS_INIT3()** and **PERL_SYS_TERM()**.

Extensive reworking of the pad code (the code responsible for lexical variables) has been conducted by Dave Mitchell.

Extensive work on the v-strings by John Peacock.

UTF-8 length and position cache: to speed up the handling of Unicode (UTF-8) scalars, a cache was introduced. Potential problems exist if an extension bypasses the official APIs and directly modifies the PV of an SV: the UTF-8 cache does not get cleared as it should.

APIs obsoleted in Perl 5.8.0, like sv_2pv, sv_catpvn, sv_catsv, sv_setsv, are again available.

Certain Perl core C APIs like cxinc and regatom are no longer available at all to code outside the Perl core of the Perl core extensions. This is intentional. They never should have been available with the shorter names, and if you application depends on them, you should (be ashamed and) contact perl5-porters to discuss what are the proper APIs.

Certain Perl core C APIs like Perl_list are no longer available without their Perl_ prefix. If your XS module stops working because some functions cannot be found, in many cases a simple fix is to add the Perl_ prefix to the function and the thread context aTHX_ as the first argument of the function call. This is also how it should always have been done: letting the Perl_-less forms to leak from the core was an accident. For cleaner embedding you can also force this for all APIs by defining at compile time the cpp define PERL_NO_SHORT_NAMES.

**Perl_save_bool()** has been added.

Regexp objects (those created with qr) now have S-magic rather than R-magic. This fixed regexps of the form /...(??{...;$x})/ to no longer ignore changes made to $x. The S-magic avoids dropping the caching optimization and making (??{...}) constructs obscenely slow (and consequently useless). See also ''Magic Variables'' in perlguts. Regexp::Copy was affected by this change.

The Perl internal debugging macros **DEBUG()** and **DEB()** have been renamed to **PERL_DEBUG()** and **PERL_DEB()** to avoid namespace conflicts.

-DL removed (the leaktest had been broken and unsupported for years, use alternative debugging mallocs or tools like valgrind and Purify).

Verbose modifier v added for -DXv and -Dsv, see perlrun.

## New Tests

In Perl 5.8.0 there were about 69000 separate tests in about 700 test files, in Perl 5.8.1 there are about 77000 separate tests in about 780 test files. The exact numbers depend on the Perl configuration and on the operating system platform.

## Known Problems

The hash randomisation mentioned in ‘‘Incompatible Changes’’ is definitely problematic: it will wake dormant bugs and shake out bad assumptions.

If you want to use mod_perl 2.x with Perl 5.8.1, you will need mod_perl−1.99_10 or higher. Earlier versions of mod_perl 2.x do not work with the randomised hashes. (mod_perl 1.x works fine.) You will also need Apache::Test 1.04 or higher.

Many of the rarer platforms that worked 100% or pretty close to it with perl 5.8.0 have been left a little bit untended since their maintainers have been otherwise busy lately, and therefore there will be more failures on those platforms. Such platforms include Mac OS Classic, IBM z/OS (and other EBCDIC platforms), and NetWare. The most common Perl platforms (Unix and Unix-like, Microsoft platforms, and VMS) have large enough testing and expert population that they are doing well.

### Tied hashes in scalar context

Tied hashes do not currently return anything useful in scalar context, for example when used as boolean tests:

```
        if (%tied_hash) { ... }
```

The current nonsensical behaviour is always to return false, regardless of whether the hash is empty or has elements.

The root cause is that there is no interface for the implementors of tied hashes to implement the behaviour of a hash in scalar context.

### Net::Ping 450_service and 510_ping_udp failures

The subtests 9 and 18 of lib/Net/Ping/t/450_service.t, and the subtest 2 of lib/Net/Ping/t/510_ping_udp.t might fail if you have an unusual networking setup. For example in the latter case the test is trying to send a UDP ping to the IP address 127.0.0.1.

### B::C

The C−generating compiler backend B::C (the frontend being `perlcc -c`) is even more broken than it used to be because of the extensive lexical variable changes. (The good news is that B::Bytecode and ByteLoader are better than they used to be.)

## Platform Specific Problems

### EBCDIC Platforms

IBM z/OS and other EBCDIC platforms continue to be problematic regarding Unicode support. Many Unicode tests are skipped when they really should be fixed.

### Cygwin 1.5 problems

In Cygwin 1.5 the *io/tell* and *op/sysio* tests have failures for some yet unknown reason. In 1.5.5 the threads tests stress_cv, stress_re, and stress_string are failing unless the environment variable PERLIO is set to ‘‘perlio’’ (which makes also the io/tell failure go away).

Perl 5.8.1 does build and work well with Cygwin 1.3: with (uname −a) `CYGWIN_NT−5.0 ... 1.3.22(0.78/3/2) 2003−03−18 09:20 i686 ...` a 100% ‘‘make test’’ was achieved with `Configure −des −Duseithreads`.

### HP-UX: HP cc warnings about sendfile and sendpath

With certain HP C compiler releases (e.g. B.11.11.02) you will get many warnings like this (lines wrapped for easier reading):

```
cc: "/usr/include/sys/socket.h", line 504: warning 562:
  Redeclaration of "sendfile" with a different storage class specifier:
    "sendfile" will have internal linkage.
cc: "/usr/include/sys/socket.h", line 505: warning 562:
  Redeclaration of "sendpath" with a different storage class specifier:
    "sendpath" will have internal linkage.
```

The warnings show up both during the build of Perl and during certain lib/ExtUtils tests that invoke the C compiler. The warning, however, is not serious and can be ignored.

#### IRIX: t/uni/tr_7jis.t falsely failing

The test t/uni/tr_7jis.t is known to report failure under 'make test' or the test harness with certain releases of IRIX (at least IRIX 6.5 and MIPSpro Compilers Version 7.3.1.1m), but if run manually the test fully passes.

#### Mac OS X: no usemymalloc

The Perl malloc (`-Dusemymalloc`) does not work at all in Mac OS X. This is not that serious, though, since the native malloc works just fine.

#### Tru64: No threaded builds with GNU cc (gcc)

In the latest Tru64 releases (e.g. v5.1B or later) gcc cannot be used to compile a threaded Perl (-Duseithreads) because the system `<pthread.h>` file doesn't know about gcc.

#### Win32: sysopen, sysread, syswrite

As of the 5.8.0 release, **sysopen()**/**sysread()**/**syswrite()** do not behave like they used to in 5.6.1 and earlier with respect to ''text'' mode. These built-ins now always operate in ''binary'' mode (even if **sysopen()** was passed the O_TEXT flag, or if **binmode()** was used on the file handle). Note that this issue should only make a difference for disk files, as sockets and pipes have always been in ''binary'' mode in the Windows port. As this behavior is currently considered a bug, compatible behavior may be re-introduced in a future release. Until then, the use of **sysopen()**, **sysread()** and **syswrite()** is not supported for ''text'' mode operations.

## Future Directions

The following things **might** happen in future. The first publicly available releases having these characteristics will be the developer releases Perl 5.9.x, culminating in the Perl 5.10.0 release. These are our best guesses at the moment: we reserve the right to rethink.

- PerlIO will become The Default. Currently (in Perl 5.8.x) the stdio library is still used if Perl thinks it can use certain tricks to make stdio go **really** fast. For future releases our goal is to make PerlIO go even faster.
- A new feature called *assertions* will be available. This means that one can have code called assertions sprinkled in the code: usually they are optimised away, but they can be enabled with the `-A` option.
- A new operator `//` (defined-or) will be available. This means that one will be able to say

  ```
  $a // $b
  ```

  instead of

  ```
  defined $a ? $a : $b
  ```

  and

  ```
  $c //= $d;
  ```

  instead of

  ```
  $c = $d unless defined $c;
  ```

  The operator will have the same precedence and associativity as `||`. A source code patch against the Perl 5.8.1 sources will be available in CPAN as *authors/id/H/HM/HMBRAND/dor-5.8.1.diff*.
- `unpack()` will default to unpacking the `$_`.
- Various Copy-On-Write techniques will be investigated in hopes of speeding up Perl.
- CPANPLUS, Inline, and Module::Build will become core modules.
- The ability to write true lexically scoped pragmas will be introduced.
- Work will continue on the bytecompiler and byteloader.
- v-strings as they currently exist are scheduled to be deprecated. The v-less form (1.2.3) will become a ''version object'' when used with `use`, `require`, and `$VERSION`. $^V will also be a ''version object'' so the printf(''%vd'',...) construct will no longer be needed. The v-ful version (v1.2.3) will become obsolete. The equivalence of strings and v-strings (e.g. that currently 5.8.0 is equal to ''\5\8\0'') will go away. **There may be no deprecation warning for v-strings**, though: it is quite hard to detect when v-strings are being used safely, and when they are not.

- 5.005 Threads Will Be Removed
- The `$*` Variable Will Be Removed (it was deprecated a long time ago)
- Pseudohashes Will Be Removed

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org/ . There may also be information at http://www.perl.com/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl582delta − what is new for perl v5.8.2

## DESCRIPTION

This document describes differences between the 5.8.1 release and the 5.8.2 release.

If you are upgrading from an earlier release such as 5.6.1, first read the perl58delta, which describes differences between 5.6.0 and 5.8.0, and the perl581delta, which describes differences between 5.8.0 and 5.8.1.

## Incompatible Changes

For threaded builds for modules calling certain re-entrant system calls, binary compatibility was accidentally lost between 5.8.0 and 5.8.1. Binary compatibility with 5.8.0 has been restored in 5.8.2, which necessitates breaking compatibility with 5.8.1. We see this as the lesser of two evils.

This will only affect people who have a threaded perl 5.8.1, and compiled modules which use these calls, and now attempt to run the compiled modules with 5.8.2. The fix is to re-compile and re-install the modules using 5.8.2.

## Core Enhancements

### Hash Randomisation

The hash randomisation introduced with 5.8.1 has been amended. It transpired that although the implementation introduced in 5.8.1 was source compatible with 5.8.0, it was not binary compatible in certain cases. 5.8.2 contains an improved implementation which is both source and binary compatible with both 5.8.0 and 5.8.1, and remains robust against the form of attack which prompted the change for 5.8.1.

We are grateful to the Debian project for their input in this area. See ‘‘Algorithmic Complexity Attacks’’ in perlsec for the original rationale behind this change.

### Threading

Several memory leaks associated with variables shared between threads have been fixed.

## Modules and Pragmata

### Updated Modules And Pragmata

The following modules and pragmata have been updated since Perl 5.8.1:

Devel::PPPort
Digest::MD5
I18N::LangTags
libnet
MIME::Base64
Pod::Perldoc
strict
: Documentation improved

Tie::Hash
: Documentation improved

Time::HiRes
Unicode::Collate
Unicode::Normalize
UNIVERSAL
: Documentation improved

## Selected Bug Fixes

Some syntax errors involving unrecognized filetest operators are now handled correctly by the parser.

## Changed Internals

Interpreter initialization is more complete when −DMULTIPLICITY is off. This should resolve problems with initializing and destroying the Perl interpreter more than once in a single process.

## Platform Specific Problems

Dynamic linker flags have been tweaked for Solaris and OS X, which should solve problems seen while building some XS modules.

Bugs in OS/2 sockets and tmpfile have been fixed.

In OS X `setreuid` and friends are troublesome − perl will now work around their problems as best possible.

## Future Directions

Starting with 5.8.3 we intend to make more frequent maintenance releases, with a smaller number of changes in each. The intent is to propagate bug fixes out to stable releases more rapidly and make upgrading stable releases less of an upheaval. This should give end users more flexibility in their choice of upgrade timing, and allow them easier assessment of the impact of upgrades. The current plan is for code freezes as follows

- 5.8.3 23:59:59 GMT, Wednesday December 31st 2003
- 5.8.4 23:59:59 GMT, Wednesday March 31st 2004
- 5.8.5 23:59:59 GMT, Wednesday June 30th 2004

with the release following soon after, when testing is complete.

See ‘‘Future Directions’’ in perl581delta for more soothsaying.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org/. There may also be information at http://www.perl.com/, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl583delta − what is new for perl v5.8.3

## DESCRIPTION

This document describes differences between the 5.8.2 release and the 5.8.3 release.

If you are upgrading from an earlier release such as 5.6.1, first read the perl58delta, which describes differences between 5.6.0 and 5.8.0, and the perl581delta and perl582delta, which describe differences between 5.8.0, 5.8.1 and 5.8.2

### Incompatible Changes

There are no changes incompatible with 5.8.2.

### Core Enhancements

A SCALAR method is now available for tied hashes. This is called when a tied hash is used in scalar context, such as

```
if (%tied_hash) {
    ...
}
```

The old behaviour was that `%tied_hash` would return whatever would have been returned for that hash before the hash was tied (so usually 0). The new behaviour in the absence of a SCALAR method is to return TRUE if in the middle of an `each` iteration, and otherwise call FIRSTKEY to check if the hash is empty (making sure that a subsequent `each` will also begin by calling FIRSTKEY). Please see "SCALAR" in perltie for the full details and caveats.

### Modules and Pragmata

CGI

Cwd

Digest

Digest::MD5

Encode

File::Spec

FindBin

A function `again` is provided to resolve problems where modules in different directories wish to use FindBin.

List::Util

You can now weaken references to read only values.

Math::BigInt

PodParser

Pod::Perldoc

POSIX

Unicode::Collate

Unicode::Normalize

Test::Harness

threads::shared

`cond_wait` has a new two argument form. `cond_timedwait` has been added.

### Utility Changes

`find2perl` now assumes `-print` as a default action. Previously, it needed to be specified explicitly.

A new utility, `prove`, makes it easy to run an individual regression test at the command line. `prove` is part of Test::Harness, which users of earlier Perl versions can install from CPAN.

### New Documentation

The documentation has been revised in places to produce more standard manpages.

The documentation for the special code blocks (BEGIN, CHECK, INIT, END) has been improved.

### Installation and Configuration Improvements

Perl now builds on OpenVMS I64

### Selected Bug Fixes

Using **substr()** on a UTF8 string could cause subsequent accesses on that string to return garbage. This was due to incorrect UTF8 offsets being cached, and is now fixed.

**join()** could return garbage when the same **join()** statement was used to process 8 bit data having earlier processed UTF8 data, due to the flags on that statement's temporary workspace not being reset correctly. This is now fixed.

`$a .. $b` will now work as expected when either `$a` or `$b` is `undef`

Using Unicode keys with tied hashes should now work correctly.

Reading $^E now preserves $!. Previously, the C code implementing $^E did not preserve `errno`, so reading $^E could cause `errno` and therefore `$!` to change unexpectedly.

Reentrant functions will (once more) work with C++. 5.8.2 introduced a bugfix which accidentally broke the compilation of Perl extensions written in C++

## New or Changed Diagnostics

The fatal error "DESTROY created new reference to dead object" is now documented in perldiag.

## Changed Internals

The hash code has been refactored to reduce source duplication. The external interface is unchanged, and aside from the bug fixes described above, there should be no change in behaviour.

`hv_clear_placeholders` is now part of the perl API

Some C macros have been tidied. In particular macros which create temporary local variables now name these variables more defensively, which should avoid bugs where names clash.

<signal.h> is now always included.

## Configuration and Building

`Configure` now invokes callbacks regardless of the value of the variable they are called for. Previously callbacks were only invoked in the `case $variable $define)` branch. This change should only affect platform maintainers writing configuration hints files.

## Platform Specific Problems

The regression test ext/threads/shared/t/wait.t fails on early RedHat 9 and HP-UX 10.20 due to bugs in their threading implementations. RedHat users should see https://rhn.redhat.com/errata/RHBA−2003−136.html and consider upgrading their glibc.

## Known Problems

Detached threads aren't supported on Windows yet, as they may lead to memory access violation problems.

There is a known race condition opening scripts in `suidperl`. `suidperl` is neither built nor installed by default, and has been deprecated since perl 5.8.0. You are advised to replace use of suidperl with tools such as sudo ( http://www.courtesan.com/sudo/ )

We have a backlog of unresolved bugs. Dealing with bugs and bug reports is unglamorous work; not something ideally suited to volunteer labour, but that is all that we have.

The perl5 development team are implementing changes to help address this problem, which should go live in early 2004.

## Future Directions

Code freeze for the next maintenance release (5.8.4) is on March 31st 2004, with release expected by mid April. Similarly 5.8.5's freeze will be at the end of June, with release by mid July.

## Obituary

Iain 'Spoon' Truskett, Perl hacker, author of perlreref and contributor to CPAN, died suddenly on 29th December 2003, aged 24. He will be missed.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org. There may also be information at http://www.perl.org, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl584delta − what is new for perl v5.8.4

## DESCRIPTION

This document describes differences between the 5.8.3 release and the 5.8.4 release.

## Incompatible Changes

Many minor bugs have been fixed. Scripts which happen to rely on previously erroneous behaviour will consider these fixes as incompatible changes :−) You are advised to perform sufficient acceptance testing on this release to satisfy yourself that this does not affect you, before putting this release into production.

The diagnostic output of Carp has been changed slightly, to add a space after the comma between arguments. This makes it much easier for tools such as web browsers to wrap it, but might confuse any automatic tools which perform detailed parsing of Carp output.

The internal dump output has been improved, so that non-printable characters such as newline and backspace are output in `\x` notation, rather than octal. This might just confuse non-robust tools which parse the output of modules such as Devel::Peek.

## Core Enhancements

### Malloc wrapping

Perl can now be built to detect attempts to assign pathologically large chunks of memory. Previously such assignments would suffer from integer wrap-around during size calculations causing a misallocation, which would crash perl, and could theoretically be used for ‘‘stack smashing’’ attacks. The wrapping defaults to enabled on platforms where we know it works (most AIX configurations, BSDi, Darwin, DEC OSF/1, FreeBSD, HP/UX, GNU Linux, OpenBSD, Solaris, VMS and most Win32 compilers) and defaults to disabled on other platforms.

### Unicode Character Database 4.0.1

The copy of the Unicode Character Database included in Perl 5.8 has been updated to 4.0.1 from 4.0.0.

### suidperl less insecure

Paul Szabo has analysed and patched `suidperl` to remove existing known insecurities. Currently there are no known holes in `suidperl`, but previous experience shows that we cannot be confident that these were the last. You may no longer invoke the set uid perl directly, so to preserve backwards compatibility with scripts that invoke #!/usr/bin/suidperl the only set uid binary is now `sperl5.8.`*n* (`sperl5.8.4` for this release). `suidperl` is installed as a hard link to `perl`; both `suidperl` and `perl` will invoke `sperl5.8.4` automatically the set uid binary, so this change should be completely transparent.

For new projects the core perl team would strongly recommend that you use dedicated, single purpose security tools such as `sudo` in preference to `suidperl`.

### format

In addition to bug fixes, `format`’s features have been enhanced. See perlform

## Modules and Pragmata

The (mis)use of `/tmp` in core modules and documentation has been tidied up. Some modules available both within the perl core and independently from CPAN (‘‘dual-life modules’’) have not yet had these changes applied; the changes will be integrated into future stable perl releases as the modules are updated on CPAN.

### Updated modules

Attribute::Handlers
B
Benchmark
CGI
Carp
Cwd
Exporter
File::Find
IO

IPC::Open3
Local::Maketext
Math::BigFloat
Math::BigInt
Math::BigRat
MIME::Base64
ODBM_File
POSIX
Shell
Socket
: There is experimental support for Linux abstract Unix domain sockets.

Storable
Switch
: Synced with its CPAN version 2.10

Sys::Syslog
: `syslog()` can now use numeric constants for facility names and priorities, in addition to strings.

Term::ANSIColor
Time::HiRes
Unicode::UCD
Win32
: Win32.pm/Win32.xs has moved from the libwin32 module to core Perl

base
open
threads
: Detached threads are now also supported on Windows.

utf8

## Performance Enhancements

- Accelerated Unicode case mappings (`/i`, `lc`, `uc`, etc).
- In place sort optimised (eg `@a = sort @a`)
- Unnecessary assignment optimised away in

```
my $s = undef;
my @a = ();
my %h = ();
```

- Optimised `map` in scalar context

## Utility Changes

The Perl debugger (*lib/perl5db.pl*) can now save all debugger commands for sourcing later, and can display the parent inheritance tree of a given class.

## Installation and Configuration Improvements

The build process on both VMS and Windows has had several minor improvements made. On Windows Borland's C compiler can now compile perl with PerlIO and/or USE_LARGE_FILES enabled.

`perl.exe` on Windows now has a ''Camel'' logo icon. The use of a camel with the topic of Perl is a trademark of O'Reilly and Associates Inc., and is used with their permission (ie distribution of the source, compiling a Windows executable from it, and using that executable locally). Use of the supplied camel for anything other than a perl executable's icon is specifically not covered, and anyone wishing to redistribute perl binaries *with* the icon should check directly with O'Reilly beforehand.

Perl should build cleanly on Stratus VOS once more.

## Selected Bug Fixes

More utf8 bugs fixed, notably in how `chomp`, `chop`, `send`, and `syswrite` and interact with utf8 data. Concatenation now works correctly when `use bytes;` is in scope.

Pragmata are now correctly propagated into (?{...}) constructions in regexps. Code such as

```
my $x = qr{ ... (??{ $x }) ... };
```

will now (correctly) fail under use strict. (As the inner `$x` is and has always referred to `$::x`)

The ''const in void context'' warning has been suppressed for a constant in an optimised-away boolean expression such as `5 || print;`

`perl -i` could `fchmod(stdin)` by mistake. This is serious if stdin is attached to a terminal, and perl is running as root. Now fixed.

## New or Changed Diagnostics

`Carp` and the internal diagnostic routines used by `Devel::Peek` have been made clearer, as described in ''Incompatible Changes''

## Changed Internals

Some bugs have been fixed in the hash internals. Restricted hashes and their place holders are now allocated and deleted at slightly different times, but this should not be visible to user code.

## Future Directions

Code freeze for the next maintenance release (5.8.5) will be on 30th June 2004, with release by mid July.

## Platform Specific Problems

This release is known not to build on Windows 95.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org. There may also be information at http://www.perl.org, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl585delta − what is new for perl v5.8.5

## DESCRIPTION

This document describes differences between the 5.8.4 release and the 5.8.5 release.

## Incompatible Changes

There are no changes incompatible with 5.8.4.

## Core Enhancements

Perl's regular expression engine now contains support for matching on the intersection of two Unicode character classes. You can also now refer to user-defined character classes from within other user defined character classes.

## Modules and Pragmata

- Carp improved to work nicely with Safe. Carp's message reporting should now be anomaly free − it will always print out line number information.
- CGI upgraded to version 3.05
- charnames now avoids clobbering `$_`
- Digest upgraded to version 1.08
- Encode upgraded to version 2.01
- FileCache upgraded to version 1.04
- libnet upgraded to version 1.19
- Pod::Parser upgraded to version 1.28
- Pod::Perldoc upgraded to version 3.13
- Pod::LaTeX upgraded to version 0.57
- Safe now works properly with Carp
- Scalar-List-Utils upgraded to version 1.14
- Shell's documentation has been re-written, and its historical partial auto-quoting of command arguments can now be disabled.
- Test upgraded to version 1.25
- Test::Harness upgraded to version 2.42
- Time::Local upgraded to version 1.10
- Unicode::Collate upgraded to version 0.40
- Unicode::Normalize upgraded to version 0.30

## Utility Changes

### Perl's debugger

The debugger can now emulate stepping backwards, by restarting and rerunning all bar the last command from a saved command history.

### h2ph

*h2ph* is now able to understand a very limited set of C inline functions — basically, the inline functions that look like CPP macros. This has been introduced to deal with some of the headers of the newest versions of the glibc. The standard warning still applies; to quote *h2ph*'s documentation, *you may need to dicker with the files produced*.

## Installation and Configuration Improvements

Perl 5.8.5 should build cleanly from source on LynxOS.

## Selected Bug Fixes

- The in-place sort optimisation introduced in 5.8.4 had a bug. For example, in code such as

  ```
  @a = sort ($b, @a)
  ```

  the result would omit the value `$b`. This is now fixed.

- The optimisation for unnecessary assignments introduced in 5.8.4 could give spurious warnings. This has been fixed.
- Perl should now correctly detect and read BOM-marked and (BOMless) UTF−16 scripts of either endianness.
- Creating a new thread when weak references exist was buggy, and would often cause warnings at interpreter destruction time. The known bug is now fixed.
- Several obscure bugs involving manipulating Unicode strings with `substr` have been fixed.
- Previously if Perl's file globbing function encountered a directory that it did not have permission to open it would return immediately, leading to unexpected truncation of the list of results. This has been fixed, to be consistent with Unix shells' globbing behaviour.
- Thread creation time could vary wildly between identical runs. This was caused by a poor hashing algorithm in the thread cloning routines, which has now been fixed.
- The internals of the ithreads implementation were not checking if OS-level thread creation had failed. threads−>**create()** now returns `undef` in if thread creation fails instead of crashing perl.

## New or Changed Diagnostics

- Perl −V has several improvements
  - correctly outputs local patch names that contain embedded code snippets or other characters that used to confuse it.
  - arguments to −V that look like regexps will give multiple lines of output.
  - a trailing colon suppresses the linefeed and ';' terminator, allowing embedding of queries into shell commands.
  - a leading colon removes the 'name=' part of the response, allowing mapping to any name.
- When perl fails to find the specified script, it now outputs a second line suggesting that the user use the `−S` flag:

```
$ perl5.8.5 missing.pl
Can't open perl script "missing.pl": No such file or directory.
Use −S to search $PATH for it.
```

## Changed Internals

The Unicode character class files used by the regular expression engine are now built at build time from the supplied Unicode consortium data files, instead of being shipped prebuilt. This makes the compressed Perl source tarball about 200K smaller. A side effect is that the layout of files inside lib/unicore has changed.

## Known Problems

The regression test *t/uni/class.t* is now performing considerably more tests, and can take several minutes to run even on a fast machine.

## Platform Specific Problems

This release is known not to build on Windows 95.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org. There may also be information at http://www.perl.org, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl −V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl586delta − what is new for perl v5.8.6

## DESCRIPTION

This document describes differences between the 5.8.5 release and the 5.8.6 release.

## Incompatible Changes

There are no changes incompatible with 5.8.5.

## Core Enhancements

The perl interpreter is now more tolerant of UTF−16−encoded scripts.

On Win32, Perl can now use non-IFS compatible LSPs, which allows Perl to work in conjunction with firewalls such as McAfee Guardian. For full details see the file *README.win32*, particularly if you're running Win95.

## Modules and Pragmata

- With the `base` pragma, an intermediate class with no fields used to messes up private fields in the base class. This has been fixed.
- Cwd upgraded to version 3.01 (as part of the new PathTools distribution)
- Devel::PPPort upgraded to version 3.03
- File::Spec upgraded to version 3.01 (as part of the new PathTools distribution)
- Encode upgraded to version 2.08
- ExtUtils::MakeMaker remains at version 6.17, as later stable releases currently available on CPAN have some issues with core modules on some core platforms.
- I18N::LangTags upgraded to version 0.35
- Math::BigInt upgraded to version 1.73
- Math::BigRat upgraded to version 0.13
- MIME::Base64 upgraded to version 3.05
- POSIX::sigprocmask function can now retrieve the current signal mask without also setting it.
- Time::HiRes upgraded to version 1.65

## Utility Changes

Perl has a new −dt command-line flag, which enables threads support in the debugger.

## Performance Enhancements

`reverse sort ...` is now optimized to sort in reverse, avoiding the generation of a temporary intermediate list.

`for (reverse @foo)` now iterates in reverse, avoiding the generation of a temporary reversed list.

## Selected Bug Fixes

The regexp engine is now more robust when given invalid utf8 input, as is sometimes generated by buggy XS modules.

`foreach` on threads::shared array used to be able to crash Perl. This bug has now been fixed.

A regexp in `STDOUT`'s destructor used to coredump, because the regexp pad was already freed. This has been fixed.

`goto &` is now more robust − bugs in deep recursion and chained `goto &` have been fixed.

Using `delete` on an array no longer leaks memory. A `pop` of an item from a shared array reference no longer causes a leak.

`eval_sv()` failing a taint test could corrupt the stack − this has been fixed.

On platforms with 64 bit pointers numeric comparison operators used to erroneously compare the addresses of references that are overloaded, rather than using the overloaded values. This has been fixed.

`read` into a UTF8−encoded buffer with an offset off the end of the buffer no longer mis-calculates buffer lengths.

Although Perl has promised since version 5.8 that `sort()` would be stable, the two cases `sort {$b cmp $a}` and `sort {$b <=> $a}` could produce non-stable sorts. This is corrected in perl5.8.6.

Localising `$^D` no longer generates a diagnostic message about valid −D flags.

## New or Changed Diagnostics

For −t and −T,

Too late for ‘‘−T’’ option has been changed to the more informative

‘‘−T’’ is on the #! line, it must also be used on the command line

## Changed Internals

From now on all applications embedding perl will behave as if perl were compiled with −DPERL_USE_SAFE_PUTENV. See ‘‘Environment access’’ in the *INSTALL* file for details.

Most `C` source files now have comments at the top explaining their purpose, which should help anyone wishing to get an overview of the implementation.

## New Tests

There are significantly more tests for the `B` suite of modules.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org. There may also be information at http://www.perl.org, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl587delta − what is new for perl v5.8.7

## DESCRIPTION

This document describes differences between the 5.8.6 release and the 5.8.7 release.

## Incompatible Changes

There are no changes incompatible with 5.8.6.

## Core Enhancements

### Unicode Character Database 4.1.0

The copy of the Unicode Character Database included in Perl 5.8 has been updated to 4.1.0 from 4.0.1. See <http://www.unicode.org/versions/Unicode4.1.0/#NotableChanges> for the notable changes.

### suidperl less insecure

A pair of exploits in `suidperl` involving debugging code have been closed.

For new projects the core perl team strongly recommends that you use dedicated, single purpose security tools such as `sudo` in preference to `suidperl`.

### Optional site customization script

The perl interpreter can be built to allow the use of a site customization script. By default this is not enabled, to be consistent with previous perl releases. To use this, add `-Dusesitecustomize` to the command line flags when running the `Configure` script. See also ‘‘−f’’ in perlrun.

### `Config.pm` is now much smaller.

`Config.pm` is now about 3K rather than 32K, with the infrequently used code and `%Config` values loaded on demand. This is transparent to the programmer, but means that most code will save parsing and loading 29K of script (for example, code that uses `File::Find`).

## Modules and Pragmata

- B upgraded to version 1.09
- base upgraded to version 2.07
- bignum upgraded to version 0.17
- bytes upgraded to version 1.02
- Carp upgraded to version 1.04
- CGI upgraded to version 3.10
- Class::ISA upgraded to version 0.33
- Data::Dumper upgraded to version 2.121_02
- DB_File upgraded to version 1.811
- Devel::PPPort upgraded to version 3.06
- Digest upgraded to version 1.10
- Encode upgraded to version 2.10
- FileCache upgraded to version 1.05
- File::Path upgraded to version 1.07
- File::Temp upgraded to version 0.16
- IO::File upgraded to version 1.11
- IO::Socket upgraded to version 1.28
- Math::BigInt upgraded to version 1.77
- Math::BigRat upgraded to version 0.15
- overload upgraded to version 1.03
- PathTools upgraded to version 3.05
- Pod::HTML upgraded to version 1.0503

- Pod::Perldoc upgraded to version 3.14
- Pod::LaTeX upgraded to version 0.58
- Pod::Parser upgraded to version 1.30
- Symbol upgraded to version 1.06
- Term::ANSIColor upgraded to version 1.09
- Test::Harness upgraded to version 2.48
- Test::Simple upgraded to version 0.54
- Text::Wrap upgraded to version 2001.09293, to fix a bug when **wrap()** was called with a non-space separator.
- threads::shared upgraded to version 0.93
- Time::HiRes upgraded to version 1.66
- Time::Local upgraded to version 1.11
- Unicode::Normalize upgraded to version 0.32
- utf8 upgraded to version 1.05
- Win32 upgraded to version 0.24, which provides Win32::GetFileVersion

## Utility Changes

### find2perl enhancements

`find2perl` has new options `-iname`, `-path` and `-ipath`.

## Performance Enhancements

The internal pointer mapping hash used during ithreads cloning now uses an arena for memory allocation. In tests this reduced ithreads cloning time by about 10%.

## Installation and Configuration Improvements

- The Win32 ‘‘dmake’’ makefile.mk has been updated to make it compatible with the latest versions of dmake.
- `PERL_MALLOC`, `DEBUG_MSTATS`, `PERL_HASH_SEED_EXPLICIT` and `NO_HASH_SEED` should now work in Win32 makefiles.

## Selected Bug Fixes

- The **socket()** function on Win32 has been fixed so that it is able to use transport providers which specify a protocol of 0 (meaning any protocol is allowed) once more. (This was broken in 5.8.6, and typically caused the use of ICMP sockets to fail.)
- Another obscure bug involving `substr` and UTF-8 caused by bad internal offset caching has been identified and fixed.
- A bug involving the loading of UTF-8 tables by the regexp engine has been fixed - code such as `"\x{100}" =~ /[[:print:]]/` will no longer give corrupt results.
- Case conversion operations such as `uc` on a long Unicode string could exhaust memory. This has been fixed.
- `index`/`rindex` were buggy for some combinations of Unicode and non-Unicode data. This has been fixed.
- `read` (and presumably `sysread`) would expose the UTF-8 internals when reading from a byte oriented file handle into a UTF-8 scalar. This has been fixed.
- Several `pack`/`unpack` bug fixes:
  - Checksums with `b` or `B` formats were broken.
  - `unpack` checksums could overflow with the `C` format.
  - `U0` and `C0` are now scoped to `()` `pack` sub-templates.
  - Counted length prefixes now don’t change `C0`/`U0` mode.

    - `pack Z0` used to destroy the preceding character.
    - `P`/`p` `pack` formats used to only recognise literal `undef`
- Using closures with ithreads could cause perl to crash. This was due to failure to correctly lock internal OP structures, and has been fixed.
- The return value of `close` now correctly reflects any file errors that occur while flushing the handle's data, instead of just giving failure if the actual underlying file close operation failed.
- `not() || 1` used to segfault. `not()` now behaves like `not(0)`, which was the pre 5.6.0 behaviour.
- `h2ph` has various enhancements to cope with constructs in header files that used to result in incorrect or invalid output.

## New or Changed Diagnostics

There is a new taint error, ''%ENV is aliased to `%s`''. This error is thrown when taint checks are enabled and when `*ENV` has been aliased, so that `%ENV` has no env-magic anymore and hence the environment cannot be verified as taint-free.

The internals of `pack` and `unpack` have been updated. All legitimate templates should work as before, but there may be some changes in the error reported for complex failure cases. Any behaviour changes for non-error cases are bugs, and should be reported.

## Changed Internals

There has been a fair amount of refactoring of the `C` source code, partly to make it tidier and more maintainable. The resulting object code and the `perl` binary may well be smaller than 5.8.6, and hopefully faster in some cases, but apart from this there should be no user-detectable changes.

`${^UTF8LOCALE}` has been added to give perl space access to `PL_utf8locale`.

The size of the arenas used to allocate SV heads and most SV bodies can now be changed at compile time. The old size was 1008 bytes, the new default size is 4080 bytes.

## Known Problems

Unicode strings returned from overloaded operators can be buggy. This is a long standing bug reported since 5.8.6 was released, but we do not yet have a suitable fix for it.

## Platform Specific Problems

On UNICOS, lib/Math/BigInt/t/bigintc.t hangs burning CPU. ext/B/t/bytecode.t and ext/Socket/t/socketpair.t both fail tests. These are unlikely to be resolved, as our valiant UNICOS porter's last Cray is being decommissioned.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org. There may also be information at http://www.perl.org, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl588delta − what is new for perl v5.8.8

## DESCRIPTION

This document describes differences between the 5.8.7 release and the 5.8.8 release.

## Incompatible Changes

There are no changes intentionally incompatible with 5.8.7. If any exist, they are bugs and reports are welcome.

## Core Enhancements

- `chdir`, `chmod` and `chown` can now work on filehandles as well as filenames, if the system supports respectively `fchdir`, `fchmod` and `fchown`, thanks to a patch provided by Gisle Aas.

## Modules and Pragmata

- `Attribute::Handlers` upgraded to version 0.78_02
  - Documentation typo fix
- `attrs` upgraded to version 1.02
  - Internal cleanup only
- `autouse` upgraded to version 1.05
  - Simplified implementation
- `B` upgraded to version 1.09_01
  - The inheritance hierarchy of the `B::` modules has been corrected; `B::NV` now inherits from `B::SV` (instead of `B::IV`).
- `blib` upgraded to version 1.03
  - Documentation typo fix
- `ByteLoader` upgraded to version 0.06
  - Internal cleanup
- `CGI` upgraded to version 3.15
  - Extraneous "?" from `self_url()` removed
  - `scrolling_list()` select attribute fixed
  - `virtual_port` now works properly with the https protocol
  - `upload_hook()` and `append()` now works in function-oriented mode
  - `POST_MAX` doesn't cause the client to hang any more
  - Automatic tab indexes are now disabled and new `-tabindex` pragma has been added to turn automatic indexes back on
  - `end_form()` doesn't emit empty (and non-validating) `<div>`
  - `CGI::Carp` works better in certain mod_perl configurations
  - Setting `$CGI::TMPDIRECTORY` is now effective
  - Enhanced documentation
- `charnames` upgraded to version 1.05
  - `viacode()` now accept hex strings and has been optimized.
- `CPAN` upgraded to version 1.76_02
  - 1 minor bug fix for Win32
- `Cwd` upgraded to version 3.12
  - `canonpath()` on Win32 now collapses *foo*\.. sections correctly.
  - Improved behaviour on Symbian OS.

    - Enhanced documentation and typo fixes
    - Internal cleanup
- `Data::Dumper` upgraded to version 2.121_08
    - A problem where `Data::Dumper` would sometimes update the iterator state of hashes has been fixed
    - Numeric labels now work
    - Internal cleanup
- `DB` upgraded to version 1.01
    - A problem where the state of the regexp engine would sometimes get clobbered when running under the debugger has been fixed.
- `DB_File` upgraded to version 1.814
    - Adds support for Berkeley DB 4.4.
- `Devel::DProf` upgraded to version 20050603.00
    - Internal cleanup
- `Devel::Peek` upgraded to version 1.03
    - Internal cleanup
- `Devel::PPPort` upgraded to version 3.06_01
    - `--compat-version` argument checking has been improved
    - Files passed on the command line are filtered by default
    - `--nofilter` option to override the filtering has been added
    - Enhanced documentation
- `diagnostics` upgraded to version 1.15
    - Documentation typo fix
- `Digest` upgraded to version 1.14
    - The constructor now knows which module implements SHA-224
    - Documentation tweaks and typo fixes
- `Digest::MD5` upgraded to version 2.36
    - `XSLoader` is now used for faster loading
    - Enhanced documentation including MD5 weaknesses discovered lately
- `Dumpvalue` upgraded to version 1.12
    - Documentation fix
- `DynaLoader` upgraded but unfortunately we're not able to increment its version number :-(
    - Implements `dl_unload_file` on Win32
    - Internal cleanup
    - `XSLoader` 0.06 incorporated; small optimisation for calling `bootstrap_inherit()` and documentation enhancements.
- `Encode` upgraded to version 2.12
    - A coderef is now acceptable for `CHECK`!
    - 3 new characters added to the ISO-8859-7 encoding
    - New encoding `MIME-Header-ISO_2022_JP` added
    - Problem with partial characters and `encoding(utf-8-strict)` fixed.

    - Documentation enhancements and typo fixes
- `English` upgraded to version 1.02
    - the `$COMPILING` variable has been added
- `ExtUtils::Constant` upgraded to version 0.17
    - Improved compatibility with older versions of perl
- `ExtUtils::MakeMaker` upgraded to version 6.30 (was 6.17)
    - Too much to list here; see <http://search.cpan.org/dist/ExtUtils−MakeMaker/Changes>
- `File::Basename` upgraded to version 2.74, with changes contributed by Michael Schwern.
    - Documentation clarified and errors corrected.
    - `basename` now strips trailing path separators before processing the name.
    - `basename` now returns `/` for parameter `/`, to make `basename` consistent with the shell utility of the same name.
    - The suffix is no longer stripped if it is identical to the remaining characters in the name, again for consistency with the shell utility.
    - Some internal code cleanup.
- `File::Copy` upgraded to version 2.09
    - Copying a file onto itself used to fail.
    - Moving a file between file systems now preserves the access and modification time stamps
- `File::Find` upgraded to version 1.10
    - Win32 portability fixes
    - Enhanced documentation
- `File::Glob` upgraded to version 1.05
    - Internal cleanup
- `File::Path` upgraded to version 1.08
    - `mkpath` now preserves `errno` when `mkdir` fails
- `File::Spec` upgraded to version 3.12
    - `File::Spec->rootdir()` now returns `\` on Win32, instead of `/`
    - `$^O` could sometimes become tainted. This has been fixed.
    - `canonpath` on Win32 now collapses `foo/..` (or `foo\..`) sections correctly, rather than doing the ''misguided'' work it was previously doing. Note that `canonpath` on Unix still does **not** collapse these sections, as doing so would be incorrect.
    - Some documentation improvements
    - Some internal code cleanup
- `FileCache` upgraded to version 1.06
    - POD formatting errors in the documentation fixed
- `Filter::Simple` upgraded to version 0.82
- `FindBin` upgraded to version 1.47
    - Now works better with directories where access rights are more restrictive than usual.
- `GDBM_File` upgraded to version 1.08
    - Internal cleanup
- `Getopt::Long` upgraded to version 2.35
    - `prefix_pattern` has now been complemented by a new configuration option `long_prefix_pattern` that allows the user to specify what prefix patterns should have long option style semantics applied.

    - Options can now take multiple values at once (experimental)
    - Various bug fixes
- `if` upgraded to version 0.05
    - Give more meaningful error messages from `if` when invoked with a condition in list context.
    - Restore backwards compatibility with earlier versions of perl
- `IO` upgraded to version 1.22
    - Enhanced documentation
    - Internal cleanup
- `IPC::Open2` upgraded to version 1.02
    - Enhanced documentation
- `IPC::Open3` upgraded to version 1.02
    - Enhanced documentation
- `List::Util` upgraded to version 1.18 (was 1.14)
    - Fix pure-perl version of `refaddr` to avoid blessing an un-blessed reference
    - Use `XSLoader` for faster loading
    - Fixed various memory leaks
    - Internal cleanup and portability fixes
- `Math::Complex` upgraded to version 1.35
    - `atan2(0, i)` now works, as do all the (computable) complex argument cases
    - Fixes for certain bugs in `make` and `emake`
    - Support returning the *k*th root directly
    - Support `[2,-3pi/8]` in `emake`
    - Support `inf` for `make`/`emake`
    - Document `make`/`emake` more visibly
- `Math::Trig` upgraded to version 1.03
    - Add more great circle routines: `great_circle_waypoint` and `great_circle_destination`
- `MIME::Base64` upgraded to version 3.07
    - Use `XSLoader` for faster loading
    - Enhanced documentation
    - Internal cleanup
- `NDBM_File` upgraded to version 1.06
    - Enhanced documentation
- `ODBM_File` upgraded to version 1.06
    - Documentation typo fixed
    - Internal cleanup
- `Opcode` upgraded to version 1.06
    - Enhanced documentation
    - Internal cleanup
- `open` upgraded to version 1.05
    - Enhanced documentation

- `overload` upgraded to version 1.04
  - Enhanced documentation
- `PerlIO` upgraded to version 1.04
  - `PerlIO::via` iterate over layers properly now
  - `PerlIO::scalar` understands `$/ = ""` now
  - `encoding(utf-8-strict)` with partial characters now works
  - Enhanced documentation
  - Internal cleanup
- `Pod::Functions` upgraded to version 1.03
  - Documentation typos fixed
- `Pod::Html` upgraded to version 1.0504
  - HTML output will now correctly link to `=item`s on the same page, and should be valid XHTML.
  - Variable names are recognized as intended
  - Documentation typos fixed
- `Pod::Parser` upgraded to version 1.32
  - Allow files that start with `=head` on the first line
  - Win32 portability fix
  - Exit status of `pod2usage` fixed
  - New `-noperldoc` switch for `pod2usage`
  - Arbitrary URL schemes now allowed
  - Documentation typos fixed
- `POSIX` upgraded to version 1.09
  - Documentation typos fixed
  - Internal cleanup
- `re` upgraded to version 0.05
  - Documentation typo fixed
- `Safe` upgraded to version 2.12
  - Minor documentation enhancement
- `SDBM_File` upgraded to version 1.05
  - Documentation typo fixed
  - Internal cleanup
- `Socket` upgraded to version 1.78
  - Internal cleanup
- `Storable` upgraded to version 2.15
  - This includes the `STORABLE_attach` hook functionality added by Adam Kennedy, and more frugal memory requirements when storing under `ithreads`, by using the `ithreads` cloning tracking code.
- `Switch` upgraded to version 2.10_01
  - Documentation typos fixed
- `Sys::Syslog` upgraded to version 0.13

    - Now provides numeric macros and meaningful `Exporter` tags.
    - No longer uses `Sys::Hostname` as it may provide useless values in unconfigured network environments, so instead uses `INADDR_LOOPBACK` directly.
    - `syslog()` now uses local timestamp.
    - `setlogmask()` now behaves like its C counterpart.
    - `setlogsock()` will now `croak()` as documented.
    - Improved error and warnings messages.
    - Improved documentation.
- `Term::ANSIColor` upgraded to version 1.10
    - Fixes a bug in `colored` when `$EACHLINE` is set that caused it to not color lines consisting solely of 0 (literal zero).
    - Improved tests.
- `Term::ReadLine` upgraded to version 1.02
    - Documentation tweaks
- `Test::Harness` upgraded to version 2.56 (was 2.48)
    - The `Test::Harness` timer is now off by default.
    - Now shows elapsed time in milliseconds.
    - Various bug fixes
- `Test::Simple` upgraded to version 0.62 (was 0.54)
    - `is_deeply()` no longer fails to work for many cases
    - Various minor bug fixes
    - Documentation enhancements
- `Text::Tabs` upgraded to version 2005.0824
    - Provides a faster implementation of `expand`
- `Text::Wrap` upgraded to version 2005.082401
    - Adds `$Text::Wrap::separator2`, which allows you to preserve existing newlines but add line-breaks with some other string.
- `threads` upgraded to version 1.07
    - `threads` will now honour `no warnings 'threads'`
    - A thread's interpreter is now freed after `$t->join()` rather than after `undef $t`, which should fix some `ithreads` memory leaks. (Fixed by Dave Mitchell)
    - Some documentation typo fixes.
- `threads::shared` upgraded to version 0.94
    - Documentation changes only
    - Note: An improved implementation of `threads::shared` is available on CPAN − this will be merged into 5.8.9 if it proves stable.
- `Tie::Hash` upgraded to version 1.02
    - Documentation typo fixed
- `Time::HiRes` upgraded to version 1.86 (was 1.66)
    - `clock_nanosleep()` and `clock()` functions added
    - Support for the POSIX `clock_gettime()` and `clock_getres()` has been added
    - Return `undef` or an empty list if the C `gettimeofday()` function fails

- Improved `nanosleep` detection
- Internal cleanup
- Enhanced documentation

- `Unicode::Collate` upgraded to version 0.52
  - Now implements UCA Revision 14 (based on Unicode 4.1.0).
  - `Unicode::Collate->new` method no longer overwrites user's `$_`
  - Enhanced documentation
- `Unicode::UCD` upgraded to version 0.24
  - Documentation typos fixed
- `User::grent` upgraded to version 1.01
  - Documentation typo fixed
- `utf8` upgraded to version 1.06
  - Documentation typos fixed
- `vmsish` upgraded to version 1.02
  - Documentation typos fixed
- `warnings` upgraded to version 1.05
  - Gentler messing with `Carp::` internals
  - Internal cleanup
  - Documentation update
- `Win32` upgraded to version 0.2601
  - Provides Windows Vista support to `Win32::GetOSName`
  - Documentation enhancements
- `XS::Typemap` upgraded to version 0.02
  - Internal cleanup

## Utility Changes

### `h2xs` enhancements

`h2xs` implements new option `--use-xsloader` to force use of `XSLoader` even in backwards compatible modules.

The handling of authors' names that had apostrophes has been fixed.

Any enums with negative values are now skipped.

### `perlivp` enhancements

`perlivp` implements new option `-a` and will not check for *.ph files by default any more. Use the `-a` option to run *all* tests.

## New Documentation

The perlglossary manpage is a glossary of terms used in the Perl documentation, technical and otherwise, kindly provided by O'Reilly Media, inc.

## Performance Enhancements

- Weak reference creation is now *O(1)* rather than *O(n)*, courtesy of Nicholas Clark. Weak reference deletion remains *O(n)*, but if deletion only happens at program exit, it may be skipped completely.
- Salvador Fandiño provided improvements to reduce the memory usage of `sort` and to speed up some cases.
- Jarkko Hietaniemi and Andy Lester worked to mark as much data as possible in the C source files as `static`, to increase the proportion of the executable file that the operating system can share between process, and thus reduce real memory usage on multi-user systems.

## Installation and Configuration Improvements

Parallel makes should work properly now, although there may still be problems if `make test` is instructed to run in parallel.

Building with Borland's compilers on Win32 should work more smoothly. In particular Steve Hay has worked to side step many warnings emitted by their compilers and at least one C compiler internal error.

`Configure` will now detect `clearenv` and `unsetenv`, thanks to a patch from Alan Burlison. It will also probe for `futimes` and whether `sprintf` correctly returns the length of the formatted string, which will both be used in perl 5.8.9.

There are improved hints for next−3.0, vmesa, IX, Darwin, Solaris, Linux, DEC/OSF, HP-UX and MPE/iX

Perl extensions on Windows now can be statically built into the Perl DLL, thanks to a work by Vadim Konovalov. (This improvement was actually in 5.8.7, but was accidentally omitted from perl587delta).

## Selected Bug Fixes

### no warnings 'category' works correctly with −w

Previously when running with warnings enabled globally via `−w`, selective disabling of specific warning categories would actually turn off all warnings. This is now fixed; now `no warnings 'io';` will only turn off warnings in the `io` class. Previously it would erroneously turn off all warnings.

This bug fix may cause some programs to start correctly issuing warnings.

### Remove over-optimisation

Perl 5.8.4 introduced a change so that assignments of `undef` to a scalar, or of an empty list to an array or a hash, were optimised away. As this could cause problems when `goto` jumps were involved, this change has been backed out.

### sprintf() fixes

Using the **sprintf()** function with some formats could lead to a buffer overflow in some specific cases. This has been fixed, along with several other bugs, notably in bounds checking.

In related fixes, it was possible for badly written code that did not follow the documentation of `Sys::Syslog` to have formatting vulnerabilities. `Sys::Syslog` has been changed to protect people from poor quality third party code.

### Debugger and Unicode slowdown

It had been reported that running under perl's debugger when processing Unicode data could cause unexpectedly large slowdowns. The most likely cause of this was identified and fixed by Nicholas Clark.

### Smaller fixes

- `FindBin` now works better with directories where access rights are more restrictive than usual.
- Several memory leaks in ithreads were closed. An improved implementation of `threads::shared` is available on CPAN − this will be merged into 5.8.9 if it proves stable.
- Trailing spaces are now trimmed from `$!` and `$^E`.
- Operations that require perl to read a process's list of groups, such as reads of `$(` and `$)`, now dynamically allocate memory rather than using a fixed sized array. The fixed size array could cause C stack exhaustion on systems configured to use large numbers of groups.
- `PerlIO::scalar` now works better with non-default `$/` settings.
- You can now use the `x` operator to repeat a `qw//` list. This used to raise a syntax error.
- The debugger now traces correctly execution in eval("")uated code that contains #line directives.
- The value of the `open` pragma is no longer ignored for three-argument opens.
- The optimisation of `for (reverse @a)` introduced in perl 5.8.6 could misbehave when the array had undefined elements and was used in LVALUE context. Dave Mitchell provided a fix.
- Some case insensitive matches between UTF−8 encoded data and 8 bit regexps, and vice versa, could give malformed character warnings. These have been fixed by Dave Mitchell and Yves Orton.

- `lcfirst` and `ucfirst` could corrupt the string for certain cases where the length UTF-8 encoding of the string in lower case, upper case or title case differed. This was fixed by Nicholas Clark.
- Perl will now use the C library calls `unsetenv` and `clearenv` if present to delete keys from `%ENV` and delete `%ENV` entirely, thanks to a patch from Alan Burlison.

## New or Changed Diagnostics

### Attempt to set length of freed array

This is a new warning, produced in situations such as this:

```
$r = do {my @a; \$#a};
$$r = 503;
```

### Non-string passed as bitmask

This is a new warning, produced when number has been passed as an argument to **select()**, instead of a bitmask.

```
# Wrong, will now warn
$rin = fileno(STDIN);
($nfound,$timeleft) = select($rout=$rin, undef, undef, $timeout);

# Should be
$rin = '';
vec($rin,fileno(STDIN),1) = 1;
($nfound,$timeleft) = select($rout=$rin, undef, undef, $timeout);
```

### Search pattern not terminated or ternary operator parsed as search pattern

This syntax error indicates that the lexer couldn't find the final delimiter of a `?PATTERN?` construct. Mentioning the ternary operator in this error message makes it easier to diagnose syntax errors.

## Changed Internals

There has been a fair amount of refactoring of the `C` source code, partly to make it tidier and more maintainable. The resulting object code and the `perl` binary may well be smaller than 5.8.7, in particular due to a change contributed by Dave Mitchell which reworked the warnings code to be significantly smaller. Apart from being smaller and possibly faster, there should be no user-detectable changes.

Andy Lester supplied many improvements to determine which function parameters and local variables could actually be declared `const` to the C compiler. Steve Peters provided new `*_set` macros and reworked the core to use these rather than assigning to macros in LVALUE context.

Dave Mitchell improved the lexer debugging output under `-DT`

Nicholas Clark changed the string buffer allocation so that it is now rounded up to the next multiple of 4 (or 8 on platforms with 64 bit pointers). This should reduce the number of calls to `realloc` without actually using any extra memory.

The `HV`'s array of `HE*`s is now allocated at the correct (minimal) size, thanks to another change by Nicholas Clark. Compile with `-DPERL_USE_LARGE_HV_ALLOC` to use the old, sloppier, default.

For XS or embedding debugging purposes, if perl is compiled with `-DDEBUG_LEAKING_SCALARS_FORK_DUMP` in addition to `-DDEBUG_LEAKING_SCALARS` then a child process is `fork`ed just before global destruction, which is used to display the values of any scalars found to have leaked at the end of global destruction. Without this, the scalars have already been freed sufficiently at the point of detection that it is impossible to produce any meaningful dump of their contents. This feature was implemented by the indefatigable Nicholas Clark, based on an idea by Mike Giroux.

## Platform Specific Problems

The optimiser on HP-UX 11.23 (Itanium 2) is currently partly disabled (scaled down to +O1) when using HP C-ANSI-C; the cause of problems at higher optimisation levels is still unclear.

There are a handful of remaining test failures on VMS, mostly due to test fixes and minor module tweaks with too many dependencies to integrate into this release from the development stream, where they have all been corrected. The following is a list of expected failures with the patch number of the fix where that is known:

```
ext/Devel/PPPort/t/ppphtest.t  #26913
ext/List/Util/t/p_tainted.t    #26912
lib/ExtUtils/t/PL_FILES.t      #26813
lib/ExtUtils/t/basic.t         #26813
t/io/fs.t
t/op/cmp.t
```

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org. There may also be information at http://www.perl.org, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl589delta − what is new for perl v5.8.9

## DESCRIPTION

This document describes differences between the 5.8.8 release and the 5.8.9 release.

## Notice

The 5.8.9 release will be the last significant release of the 5.8.x series. Any future releases of 5.8.x will likely only be to deal with security issues, and platform build failures. Hence you should look to migrating to 5.10.x, if you have not started already. See ‘‘Known Problems’’ for more information.

## Incompatible Changes

A particular construction in the source code of extensions written in C++ may need changing. See ‘‘Changed Internals’’ for more details. All extensions written in C, most written in C++, and all existing compiled extensions are unaffected. This was necessary to improve C++ support.

Other than this, there are no changes intentionally incompatible with 5.8.8. If any exist, they are bugs and reports are welcome.

## Core Enhancements

### Unicode Character Database 5.1.0.

The copy of the Unicode Character Database included in Perl 5.8 has been updated to 5.1.0 from 4.1.0. See <http://www.unicode.org/versions/Unicode5.1.0/#NotableChanges> for the notable changes.

### stat and −X on directory handles

It is now possible to call `stat` and the `-X` filestat operators on directory handles. As both directory and file handles are barewords, there can be ambiguities over which was intended. In these situations the file handle semantics are preferred. Both also treat `*FILE{IO}` filehandles like `*FILE` filehandles.

### Source filters in `@INC`

It’s possible to enhance the mechanism of subroutine hooks in `@INC` by adding a source filter on top of the filehandle opened and returned by the hook. This feature was planned a long time ago, but wasn’t quite working until now. See ‘‘require’’ in perlfunc for details. (Nicholas Clark)

### Exceptions in constant folding

The constant folding routine is now wrapped in an exception handler, and if folding throws an exception (such as attempting to evaluate 0/0), perl now retains the current optree, rather than aborting the whole program. Without this change, programs would not compile if they had expressions that happened to generate exceptions, even though those expressions were in code that could never be reached at runtime. (Nicholas Clark, Dave Mitchell)

### `no VERSION`

You can now use `no` followed by a version number to specify that you want to use a version of perl older than the specified one.

### Improved internal UTF−8 caching code

The code that caches calculated UTF−8 byte offsets for character offsets for a string has been re-written. Several bugs have been located and eliminated, and the code now makes better use of the information it has, so should be faster. In particular, it doesn’t scan to the end of a string before calculating an offset within the string, which should speed up some operations on long strings. It is now possible to disable the caching code at run time, to verify that it is not the cause of suspected problems.

### Runtime relocatable installations

There is now *Configure* support for creating a perl tree that is relocatable at run time. see ‘‘Relocatable installations’’.

### New internal variables

`${^CHILD_ERROR_NATIVE}`

This variable gives the native status returned by the last pipe close, backtick command, successful call to `wait` or `waitpid`, or from the `system` operator. See perlvar for details. (Contributed by Gisle Aas.)

`${^UTF8CACHE}`

This variable controls the state of the internal UTF−8 offset caching code. 1 for on (the default), 0 for off, −1 to debug the caching code by checking all its results against linear scans, and panicking on any discrepancy.

### `readpipe` is now overridable

The built-in function `readpipe` is now overridable. Overriding it permits also to override its operator counterpart, `qx//` (also known as ``` `` ```).

### simple exception handling macros

Perl 5.8.9 (and 5.10.0 onwards) now provides a couple of macros to do very basic exception handling in XS modules. You can use these macros if you call code that may `croak`, but you need to do some cleanup before giving control back to Perl. See ''Exception Handling'' in perlguts for more details.

### -D option enhancements

- `-Dq` suppresses the *EXECUTING...* message when running under `-D`
- `-Dl` logs runops loop entry and exit, and jump level popping.
- `-Dv` displays the process id as part of the trace output.

### XS-assisted SWASHGET

Some pure-perl code that the regexp engine was using to retrieve Unicode properties and transliteration mappings has been reimplemented in XS for faster execution. (SADAHIRO Tomoyuki)

### Constant subroutines

The interpreter internals now support a far more memory efficient form of inlineable constants. Storing a reference to a constant value in a symbol table is equivalent to a full typeglob referencing a constant subroutine, but using about 400 bytes less memory. This proxy constant subroutine is automatically upgraded to a real typeglob with subroutine if necessary. The approach taken is analogous to the existing space optimisation for subroutine stub declarations, which are stored as plain scalars in place of the full typeglob.

However, to aid backwards compatibility of existing code, which (wrongly) does not expect anything other than typeglobs in symbol tables, nothing in core uses this feature, other than the regression tests.

Stubs for prototyped subroutines have been stored in symbol tables as plain strings, and stubs for unprototyped subroutines as the number −1, since 5.005, so code which assumes that the core only places typeglobs in symbol tables has been making incorrect assumptions for over 10 years.

## New Platforms

Compile support added for:

- DragonFlyBSD
- MidnightBSD
- MirOS BSD
- RISC OS
- Cray XT4/Catamount

## Modules and Pragmata

### New Modules

- `Module::Pluggable` is a simple framework to create modules that accept pluggable sub-modules. The bundled version is 3.8
- `Module::CoreList` is a hash of hashes that is keyed on perl version as indicated in `$]`. The bundled version is 2.17
- `Win32API::File` now available in core on Microsoft Windows. The bundled version is 0.1001_01
- `Devel::InnerPackage` finds all the packages defined by a single file. It is part of the `Module::Pluggable` distribution. The bundled version is 0.3

### Updated Modules

- `attributes` upgraded to version 0.09
- `AutoLoader` upgraded to version 5.67
- `AutoSplit` upgraded to 1.06
- `autouse` upgraded to version 1.06

- B upgraded from 1.09_01 to 1.19
  - provides new pad related abstraction macros B::NV::COP_SEQ_RANGE_LOW, B::NV::COP_SEQ_RANGE_HIGH, B::NV::PARENT_PAD_INDEX, B::NV::PARENT_FAKELEX_FLAGS, which hides the difference in storage in 5.10.0 and later.
  - provides B::sub_generation, which exposes PL_sub_generation
  - provides B::GV::isGV_with_GP, which on pre−5.10 perls always returns true.
  - New type B::HE added with methods VAL, HASH and SVKEY_force
  - The B::GVf_IMPORTED_CV flag is now set correctly when a proxy constant subroutine is imported.
  - bugs fixed in the handling of PMOPs.
  - B::BM::PREVIOUS returns now U32, not U16. B::CV::START and B:CV::ROOT return now NULL on an XSUB, B::CV::XSUB and B::CV::XSUBANY return 0 on a non-XSUB.
- B::C upgraded to 1.05
- B::Concise upgraded to 0.76
  - new option −src causes the rendering of each statement (starting with the nextstate OP) to be preceded by the first line of source code that generates it.
  - new option −stash="somepackage", requires "somepackage", and then renders each function defined in its namespace.
  - now has documentation of detailed hint symbols.
- B::Debug upgraded to version 1.05
- B::Deparse upgraded to version 0.87
  - properly deparse print readpipe $x, $y.
  - now handles ''−>(), ::(), sub :: {}, *etc.* correctly [RT #43010]. All bugs in parsing these kinds of syntax are now fixed:

```
perl −MO=Deparse −e '"my %h = "−>()'
perl −MO=Deparse −e '::−>()'
perl −MO=Deparse −e 'sub :: {}'
perl −MO=Deparse −e 'package a; sub a::b::c {}'
perl −MO=Deparse −e 'sub the::main::road {}'
```

  - does **not** deparse $^H{v_string}, which is automatically set by the internals.
- B::Lint upgraded to version 1.11
- B::Terse upgraded to version 1.05
- base upgraded to version 2.13
  - loading a module via base.pm would mask a global $SIG{__DIE__} in that module.
  - push all classes at once in @ISA
- Benchmark upgraded to version 1.10
- bigint upgraded to 0.23
- bignum upgraded to 0.23
- bigrat upgraded to 0.23
- blib upgraded to 0.04
- Carp upgraded to version 1.10

  The argument backtrace code now shows undef as undef, instead of a string *"undef"*.

- `CGI` upgraded to version 3.42
- `charnames` upgraded to 1.06
- `constant` upgraded to version 1.17
- `CPAN` upgraded to version 1.9301
- `Cwd` upgraded to version 3.29 with some platform specific improvements (including for VMS).
- `Data::Dumper` upgraded to version 2.121_17
  - Fixes hash iterator current position with the pure Perl version [RT #40668]
  - Performance enhancements, which will be most evident on platforms where repeated calls to C's `realloc()` are slow, such as Win32.
- `DB_File` upgraded to version 1.817
- `DB_Filter` upgraded to version 0.02
- `Devel::DProf` upgraded to version 20080331.00
- `Devel::Peek` upgraded to version 1.04
- `Devel::PPPort` upgraded to version 3.14
- `diagnostics` upgraded to version 1.16
- `Digest` upgraded to version 1.15
- `Digest::MD5` upgraded to version 2.37
- `DirHandle` upgraded to version 1.02
  - now localises `$.`, `$@`, `$!`, `$^E`, and `$?` before closing the directory handle to suppress leaking any side effects of warnings about it already being closed.
- `DynaLoader` upgraded to version 1.09

  `DynaLoader` can now dynamically load a loadable object from a file with a non-default file extension.
- `Encode` upgraded to version 2.26

  `Encode::Alias` includes a fix for encoding ''646'' on Solaris (better known as ASCII).
- `English` upgraded to version 1.03
- `Errno` upgraded to version 1.10
- `Exporter` upgraded to version 5.63
- `ExtUtils::Command` upgraded to version 1.15
- `ExtUtils::Constant` upgraded to version 0.21
- `ExtUtils::Embed` upgraded to version 1.28
- `ExtUtils::Install` upgraded to version 1.50_01
- `ExtUtils::Installed` upgraded to version 1.43
- `ExtUtils::MakeMaker` upgraded to version 6.48
  - support for `INSTALLSITESCRIPT` and `INSTALLVENDORSCRIPT` configuration.
- `ExtUtils::Manifest` upgraded to version 1.55
- `ExtUtils::ParseXS` upgraded to version 2.19
- `Fatal` upgraded to version 1.06
  - allows built-ins in `CORE::GLOBAL` to be made fatal.
- `Fcntl` upgraded to version 1.06
- `fields` upgraded to version 2.12

- `File::Basename` upgraded to version 2.77
- `FileCache` upgraded to version 1.07
- `File::Compare` upgraded to 1.1005
- `File::Copy` upgraded to 2.13
  - now uses 3−arg open.
- `File::DosGlob` upgraded to 1.01
- `File::Find` upgraded to version 1.13
- `File::Glob` upgraded to version 1.06
  - fixes spurious results with brackets inside braces.
- `File::Path` upgraded to version 2.07_02
- `File::Spec` upgraded to version 3.29
  - improved handling of bad arguments.
  - some platform specific improvements (including for VMS and Cygwin), with an optimisation on `abs2rel` when handling both relative arguments.
- `File::stat` upgraded to version 1.01
- `File::Temp` upgraded to version 0.20
- `filetest` upgraded to version 1.02
- `Filter::Util::Call` upgraded to version 1.07
- `Filter::Simple` upgraded to version 0.83
- `FindBin` upgraded to version 1.49
- `GDBM_File` upgraded to version 1.09
- `Getopt::Long` upgraded to version 2.37
- `Getopt::Std` upgraded to version 1.06
- `Hash::Util` upgraded to version 0.06
- `if` upgraded to version 0.05
- `IO` upgraded to version 1.23

  Reduced number of calls to `getpeername` in `IO::Socket`
- `IPC::Open` upgraded to version 1.03
- `IPC::Open3` upgraded to version 1.03
- `IPC::SysV` upgraded to version 2.00
- `lib` upgraded to version 0.61
  - avoid warning about loading *.par* files.
- `libnet` upgraded to version 1.22
- `List::Util` upgraded to 1.19
- `Locale::Maketext` upgraded to 1.13
- `Math::BigFloat` upgraded to version 1.60
- `Math::BigInt` upgraded to version 1.89
- `Math::BigRat` upgraded to version 0.22
  - implements new `as_float` method.
- `Math::Complex` upgraded to version 1.54.
- `Math::Trig` upgraded to version 1.18.

- `NDBM_File` upgraded to version 1.07
  - improve *g++* handling for systems using GDBM compatibility headers.
- `Net::Ping` upgraded to version 2.35
- `NEXT` upgraded to version 0.61
  - fix several bugs with `NEXT` when working with `AUTOLOAD`, `eval` block, and within overloaded stringification.
- `ODBM_File` upgraded to 1.07
- `open` upgraded to 1.06
- `ops` upgraded to 1.02
- `PerlIO::encoding` upgraded to version 0.11
- `PerlIO::scalar` upgraded to version 0.06
  - [RT #40267] `PerlIO::scalar` doesn't respect readonly-ness.
- `PerlIO::via` upgraded to version 0.05
- `Pod::Html` upgraded to version 1.09
- `Pod::Parser` upgraded to version 1.35
- `Pod::Usage` upgraded to version 1.35
- `POSIX` upgraded to version 1.15
  - `POSIX` constants that duplicate those in `Fcntl` are now imported from `Fcntl` and re-exported, rather than being duplicated by `POSIX`
  - `POSIX::remove` can remove empty directories.
  - `POSIX::setlocale` safer to call multiple times.
  - `POSIX::SigRt` added, which provides access to POSIX realtime signal functionality on systems that support it.
- `re` upgraded to version 0.06_01
- `Safe` upgraded to version 2.16
- `Scalar::Util` upgraded to 1.19
- `SDBM_File` upgraded to version 1.06
- `SelfLoader` upgraded to version 1.17
- `Shell` upgraded to version 0.72
- `sigtrap` upgraded to version 1.04
- `Socket` upgraded to version 1.81
  - this fixes an optimistic use of `gethostbyname`
- `Storable` upgraded to 2.19
- `Switch` upgraded to version 2.13
- `Sys::Syslog` upgraded to version 0.27
- `Term::ANSIColor` upgraded to version 1.12
- `Term::Cap` upgraded to version 1.12
- `Term::ReadLine` upgraded to version 1.03
- `Test::Builder` upgraded to version 0.80
- `Test::Harness` upgraded version to 2.64
  - this makes it able to handle newlines.

- `Test::More` upgraded to version 0.80
- `Test::Simple` upgraded to version 0.80
- `Text::Balanced` upgraded to version 1.98
- `Text::ParseWords` upgraded to version 3.27
- `Text::Soundex` upgraded to version 3.03
- `Text::Tabs` upgraded to version 2007.1117
- `Text::Wrap` upgraded to version 2006.1117
- `Thread` upgraded to version 2.01
- `Thread::Semaphore` upgraded to version 2.09
- `Thread::Queue` upgraded to version 2.11
  - added capability to add complex structures (e.g., hash of hashes) to queues.
  - added capability to dequeue multiple items at once.
  - added new methods to inspect and manipulate queues: `peek`, `insert` and `extract`
- `Tie::Handle` upgraded to version 4.2
- `Tie::Hash` upgraded to version 1.03
- `Tie::Memoize` upgraded to version 1.1
  - `Tie::Memoize::EXISTS` now correctly caches its results.
- `Tie::RefHash` upgraded to version 1.38
- `Tie::Scalar` upgraded to version 1.01
- `Tie::StdHandle` upgraded to version 4.2
- `Time::gmtime` upgraded to version 1.03
- `Time::Local` upgraded to version 1.1901
- `Time::HiRes` upgraded to version 1.9715 with various build improvements (including VMS) and minor platform-specific bug fixes (including for HP-UX 11 ia64).
- `threads` upgraded to 1.71
  - new thread state information methods: `is_running`, `is_detached` and `is_joinable`. `list` method enhanced to return running or joinable threads.
  - new thread signal method: `kill`
  - added capability to specify thread stack size.
  - added capability to control thread exiting behavior. Added a new `exit` method.
- `threads::shared` upgraded to version 1.27
  - smaller and faster implementation that eliminates one internal structure and the consequent level of indirection.
  - user locks are now stored in a safer manner.
  - new function `shared_clone` creates a copy of an object leaving shared elements as-is and deep-cloning non-shared elements.
  - added new `is_shared` method.
- `Unicode::Normalize` upgraded to version 1.02
- `Unicode::UCD` upgraded to version 0.25
- `warnings` upgraded to version 1.05_01
- `Win32` upgraded to version 0.38
  - added new function `GetCurrentProcessId` which returns the regular Windows process identifier of the current process, even when called from within a fork.

- `XSLoader` upgraded to version 0.10
- `XS::APItest` and `XS::Typemap` are for internal use only and hence no longer installed. Many more tests have been added to `XS::APItest`.

## Utility Changes

### debugger upgraded to version 1.31

- Andreas König contributed two functions to save and load the debugger history.
- `NEXT::AUTOLOAD` no longer emits warnings under the debugger.
- The debugger should now correctly find tty the device on OS X 10.5 and VMS when the program `fork`s.
- LVALUE subs now work inside the debugger.

*perlthanks*

Perl 5.8.9 adds a new utility *perlthanks*, which is a variant of *perlbug*, but for sending non-bug-reports to the authors and maintainers of Perl. Getting nothing but bug reports can become a bit demoralising − we'll see if this changes things.

*perlbug*

*perlbug* now checks if you're reporting about a non-core module and suggests you report it to the CPAN author instead.

*h2xs*

- won't define an empty string as a constant [RT #25366]
- has examples for `h2xs -X`

*h2ph*

- now attempts to deal sensibly with the difference in path implications between `""` and `<>` quoting in `#include` statements.
- now generates correct code for `#if defined A || defined B` [RT #39130]

## New Documentation

As usual, the documentation received its share of corrections, clarifications and other nitfixes. More tags were added for indexing.

perlunitut is a tutorial written by Juerd Waalboer on Unicode-related terminology and how to correctly handle Unicode in Perl scripts.

perlunicode is updated in section user defined properties.

perluniintro has been updated in the example of detecting data that is not valid in particular encoding.

perlcommunity provides an overview of the Perl Community along with further resources.

CORE documents the pseudo-namespace for Perl's core routines.

## Changes to Existing Documentation

perlglossary adds *deprecated modules and features* and *to be dropped modules*.

perlhack has been updated and added resources on smoke testing.

The Perl FAQs (*perlfaq1..perlfaq9*) have been updated.

perlcheat is updated with better details on `\w`, `\d`, and `\s`.

perldebug is updated with information on how to call the debugger.

perldiag documentation updated with *subroutine with an ampersand* on the argument to `exists` and `delete` and also several terminology updates on warnings.

perlfork documents the limitation of `exec` inside pseudo-processes.

perlfunc:

- Documentation is fixed in section `caller` and `pop`.
- Function `alarm` now mentions `Time::HiRes::ualarm` in preference to `select`.

- Regarding precedence in `-X`, filetest operators are the same as unary operators, but not regarding parsing and parentheses (spotted by Eirik Berg Hanssen).
- `reverse` function documentation received scalar context examples.

perllocale documentation is adjusted for number localization and `POSIX::setlocale` to fix Debian bug #379463.

perlmodlib is updated with `CPAN::API::HOWTO` and `Sys::Syslog::win32::Win32`

perlre documentation updated to reflect the differences between `[[:xxxxx:]]` and `\p{IsXxxxx}` matches. Also added section on `/g` and `/c` modifiers.

perlreguts describe the internals of the regular expressions engine. It has been contributed by Yves Orton.

perlrebackslash describes all perl regular expression backslash and escape sequences.

perlrecharclass describes the syntax and use of character classes in Perl Regular Expressions.

perlrun is updated to clarify on the hash seed *PERL_HASH_SEED*. Also more information in options `-x` and `-u`.

perlsub example is updated to use a lexical variable for `opendir` syntax.

perlvar fixes confusion about real GID `$(` and effective GID `$)`.

Perl thread tutorial example is fixed in section ‘‘Queues: Passing Data Around’’ in perlthrtut and perlthrtut.

perlhack documentation extensively improved by Jarkko Hietaniemi and others.

perltoot provides information on modifying `@UNIVERSAL::ISA`.

perlport documentation extended to include different `kill(-9, ...)` semantics on Windows. It also clearly states `dump` is not supported on Win32 and cygwin.

*INSTALL* has been updated and modernised.

## Performance Enhancements

- The default since perl 5.000 has been for perl to create an empty scalar with every new typeglob. The increased use of lexical variables means that most are now unused. Thanks to Nicholas Clark’s efforts, Perl can now be compiled with `-DPERL_DONT_CREATE_GVSV` to avoid creating these empty scalars. This will significantly decrease the number of scalars allocated for all configurations, and the number of scalars that need to be copied for ithread creation. Whilst this option is binary compatible with existing perl installations, it does change a long-standing assumption about the internals, hence it is not enabled by default, as some third party code may rely on the old behaviour.

  We would recommend testing with this configuration on new deployments of perl, particularly for multi-threaded servers, to see whether all third party code is compatible with it, as this configuration may give useful performance improvements. For existing installations we would not recommend changing to this configuration unless thorough testing is performed before deployment.
- `diagnostics` no longer uses `$&`, which results in large speedups for regexp matching in all code using it.
- Regular expressions classes of a single character are now treated the same as if the character had been used as a literal, meaning that code that uses char-classes as an escaping mechanism will see a speedup. (Yves Orton)
- Creating anonymous array and hash references (ie. `[]` and `{}`) now incurs no more overhead than creating an anonymous list or hash. Nicholas Clark provided changes with a saving of two ops and one stack push, which was measured as a slightly better than 5% improvement for these operations.
- Many calls to `strlen()` have been eliminated, either because the length was already known, or by adopting or enhancing APIs that pass lengths. This has been aided by the adoption of a `my_sprintf()` wrapper, which returns the correct C89 value − the length of the formatted string. Previously we could not rely on the return value of `sprintf()`, because on some ancient

but extant platforms it still returns `char *`.

- `index` is now faster if the search string is stored in UTF-8 but only contains characters in the Latin-1 range.
- The Unicode swatch cache inside the regexp engine is now used. (the lookup had a key mismatch, present since the initial implementation). [RT #42839]

## Installation and Configuration Improvements

### Relocatable installations

There is now *Configure* support for creating a relocatable perl tree. If you *Configure* with `-Duserelocatableinc`, then the paths in `@INC` (and everything else in `%Config`) can be optionally located via the path of the *perl* executable.

At start time, if any paths in `@INC` or `Config` that *Configure* marked as relocatable (by starting them with `"../"`), then they are prefixed the directory of `$^X`. This allows the relocation can be configured on a per-directory basis, although the default with `-Duserelocatableinc` is that everything is relocated. The initial install is done to the original configured prefix.

### Configuration improvements

*Configure* is now better at removing temporary files. Tom Callaway (from RedHat) also contributed patches that complete the set of flags passed to the compiler and the linker, in particular that `-fPIC` is now enabled on Linux. It will also croak when your */dev/null* isn't a device.

A new configuration variable `d_pseudofork` has been to *Configure*, and is available as `$Config{d_pseudofork}` in the `Config` module. This distinguishes real `fork` support from the pseudofork emulation used on Windows platforms.

*Config.pod* and *config.sh* are now placed correctly for cross-compilation.

`$Config{useshrplib}` is now 'true' rather than 'yes' when using a shared perl library.

### Compilation improvements

Parallel makes should work properly now, although there may still be problems if `make test` is instructed to run in parallel.

Many compilation warnings have been cleaned up. A very stubborn compiler warning in `S_emulate_eaccess()` was killed after six attempts. *g++* support has been tuned, especially for FreeBSD.

*mkppport* has been integrated, and all *ppport.h* files in the core will now be autogenerated at build time (and removed during cleanup).

### Installation improvements.

*installman* now works with `-Duserelocatableinc` and `DESTDIR`.

*installperl* no longer installs:

- static library files of statically linked extensions when a shared perl library is being used. (They are not needed. See ''Windows'' below).
- *SIGNATURE* and *PAUSE*.pub* (CPAN files)
- *NOTES* and *PATCHING* (ExtUtils files)
- *perlld* and *ld2* (Cygwin files)

### Platform Specific Changes

There are improved hints for AIX, Cygwin, DEC/OSF, FreeBSD, HP/UX, Irix 6 Linux, MachTen, NetBSD, OS/390, QNX, SCO, Solaris, SunOS, System V Release 5.x (UnixWare 7, OpenUNIX 8), Ultrix, UMIPS, uts and VOS.

*FreeBSD*

- Drop `-std=c89` and `-ansi` if using `long long` as the main integral type, else in FreeBSD 6.2 (and perhaps other releases), system headers do not declare some functions required by perl.

*Solaris*

- Starting with Solaris 10, we do not want versioned shared libraries, because those often indicate a private use only library. These problems could often be triggered when SUNWbdb (Berkeley DB) was installed. Hence if Solaris 10 is detected set `ignore_versioned_solibs=y`.

*VMS*

- Allow IEEE math to be deselected on OpenVMS I64 (but it remains the default).
- Record IEEE usage in `config.h`
- Help older VMS compilers by using `ccflags` when building `munchconfig.exe`.
- Don't try to build old `Thread` extension on VMS when `-Duseithreads` has been chosen.
- Passing a raw string of ''NaN'' to *nawk* causes a core dump − so the string has been changed to ''*NaN*''
- *t/op/stat.t* tests will now test hard links on VMS if they are supported.

*Windows*

- When using a shared perl library *installperl* no longer installs static library files, import library files and export library files (of statically linked extensions) and empty bootstrap files (of dynamically linked extensions). This fixes a problem building PAR-Packer on Win32 with a debug build of perl.
- Various improvements to the win32 build process, including support for Visual C++ 2005 Express Edition (aka Visual C++ 8.x).
- *perl.exe* will now have an icon if built with MinGW or Borland.
- Improvements to the perl−static.exe build process.
- Add Win32 makefile option to link all extensions statically.
- The *WinCE* directory has been merged into the *Win32* directory.
- `setlocale` tests have been re-enabled for Windows XP onwards.

## Selected Bug Fixes

### Unicode

Many many bugs related to the internal Unicode implementation (UTF−8) have been fixed. In particular, long standing bugs related to returning Unicode via `tie`, overloading or `$@` are now gone, some of which were never reported.

`unpack` will internally convert the string back from UTF−8 on numeric types. This is a compromise between the full consistency now in 5.10, and the current behaviour, which is often used as a ''feature'' on string types.

Using `:crlf` and `UTF-16` IO layers together will now work.

Fixed problems with `split`, Unicode `/\s+/` and `/ \0/`.

Fixed bug RT #40641 − encoding of Unicode characters in regular expressions.

Fixed a bug where using certain patterns in a regexp led to a panic. [RT #45337]

Perl no longer segfaults (due to infinite internal recursion) if the locale's character is not UTF−8 [RT #41442]:

```
use open ':locale';
print STDERR "\x{201e}"; # &bdquo;
```

### PerlIO

Inconsistencies have been fixed in the reference counting PerlIO uses to keep track of Unix file descriptors, and the API used by XS code to manage getting and releasing `FILE *`s

### Magic

Several bugs have been fixed in Magic, the internal system used to implement features such as `tie`, tainting and threads sharing.

`undef @array` on a tied array now correctly calls the `CLEAR` method.

Some of the bitwise ops were not checking whether their arguments were magical before using them. [RT #24816]

Magic is no longer invoked twice by the expression `\&$x`

A bug with assigning large numbers and tainting has been resolved. [RT #40708]

A new entry has been added to the MAGIC vtable − `svt_local`. This is used when copying magic to the new value during `local`, allowing certain problems with localising shared variables to be resolved.

For the implementation details, see ‘‘Magic Virtual Tables’’ in perlguts.

### Reblessing overloaded objects now works

Internally, perl object-ness is on the referent, not the reference, even though methods can only be called via a reference. However, the original implementation of overloading stored flags related to overloading on the reference, relying on the flags being copied when the reference was copied, or set at the creation of a new reference. This manifests in a bug − if you rebless an object from a class that has overloading, into one that does not, then any other existing references think that they (still) point to an overloaded object, choose these C code paths, and then throw errors. Analogously, blessing into an overloaded class when other references exist will result in them not using overloading.

The implementation has been fixed for 5.10, but this fix changes the semantics of flag bits, so is not binary compatible, so can’t be applied to 5.8.9. However, 5.8.9 has a work-around that implements the same bug fix. If the referent has multiple references, then all the other references are located and corrected. A full search is avoided whenever possible by scanning lexicals outwards from the current subroutine, and the argument stack.

A certain well known Linux vendor applied incomplete versions of this bug fix to their */usr/bin/perl* and then prematurely closed bug reports about performance issues without consulting back upstream. This not being enough, they then proceeded to ignore the necessary fixes to these unreleased changes for 11 months, until massive pressure was applied by their long-suffering paying customers, catalysed by the failings being featured on a prominent blog and Slashdot.

### `strict` now propagates correctly into string evals

Under 5.8.8 and earlier:

```
$ perl5.8.8 −e 'use strict; eval "use foo bar" or die $@'
Can't locate foo.pm in @INC (@INC contains: ... .) at (eval 1) line 2.
BEGIN failed−−compilation aborted at (eval 1) line 2.
```

Under 5.8.9 and later:

```
$ perl5.8.9 −e 'use strict; eval "use foo bar" or die $@'
Bareword "bar" not allowed while "strict subs" in use at (eval 1) line 1.
```

This may cause problems with programs that parse the error message and rely on the buggy behaviour.

### Other fixes

- The tokenizer no longer treats `=cute` (and other words beginning with `=cut`) as a synonym for `=cut`.
- Calling `CORE::require`

  `CORE::require` and `CORE::do` were always parsed as `require` and `do` when they were overridden. This is now fixed.
- Stopped memory leak on long */etc/groups* entries.
- `while (my $x ...) { ...; redo }` shouldn’t `undef $x`.

  In the presence of `my` in the conditional of a `while()`, `until()`, or `for(;;)` loop, we now add an extra scope to the body so that `redo` doesn’t `undef` the lexical.
- The `encoding` pragma now correctly ignores anything following an `@` character in the `LC_ALL` and `LANG` environment variables. [RT # 49646]
- A segfault observed with some *gcc* 3.3 optimisations is resolved.
- A possible segfault when `unpack` used in scalar context with `()` groups is resolved. [RT #50256]
- Resolved issue where `$!` could be changed by a signal handler interrupting a `system` call.
- Fixed bug RT #37886, symbolic dereferencing was allowed in the argument of `defined` even under the influence of `use strict 'refs'`.
- Fixed bug RT #43207, where `lc`/`uc` inside `sort` affected the return value.

- Fixed bug RT #45607, where `*{"BONK"} = \&{"BONK"}` didn't work correctly.
- Fixed bug RT #35878, croaking from a XSUB called via `goto &xsub` corrupts perl internals.
- Fixed bug RT #32539, *DynaLoader.o* is moved into *libperl.so* to avoid the need to statically link DynaLoader into the stub perl executable. With this *libperl.so* provides everything needed to get a functional embedded perl interpreter to run.
- Fix bug RT #36267 so that assigning to a tied hash doesn't change the underlying hash.
- Fix bug RT #6006, regexp replaces using large replacement variables fail some of the time, *i.e.* when substitution contains something like `${10}` (note the bracket) instead of just `$10`.
- Fix bug RT #45053, `Perl_newCONSTSUB()` is now thread safe.

### Platform Specific Fixes

*Darwin / MacOS X*

- Various improvements to 64 bit builds.
- Mutex protection added in `PerlIOStdio_close()` to avoid race conditions. Hopefully this fixes failures in the threads tests *free.t* and *blocks.t*.
- Added forked terminal support to the debugger, with the ability to update the window title.

*OS/2*

- A build problem with specifying `USE_MULTI` and `USE_ITHREADS` but without `USE_IMP_SYS` has been fixed.
- `OS2::REXX` upgraded to version 1.04

*Tru64*

- Aligned floating point build policies for *cc* and *gcc*.

*RedHat Linux*

- Revisited a patch from 5.6.1 for RH7.2 for Intel's *icc* [RT #7916], added an additional check for `$Config{gccversion}`.

*Solaris/i386*

- Use `-DPTR_IS_LONG` when using 64 bit integers

*VMS*

- Fixed `PerlIO::Scalar` in-memory file record-style reads.
- pipe shutdown at process exit should now be more robust.
- Bugs in VMS exit handling tickled by `Test::Harness` 2.64 have been fixed.
- Fix `fcntl()` locking capability test in *configure.com*.
- Replaced `shrplib='define'` with `useshrplib='true'` on VMS.

*Windows*

- `File::Find` used to fail when the target directory is a bare drive letter and `no_chdir` is 1 (the default is 0). [RT #41555]
- A build problem with specifying `USE_MULTI` and `USE_ITHREADS` but without `USE_IMP_SYS` has been fixed.
- The process id is no longer truncated to 16 bits on some Windows platforms ( http://bugs.activestate.com/show_bug.cgi?id=72443 )
- Fixed bug RT #54828 in *perlio.c* where calling `binmode` on Win32 and Cygwin may cause a segmentation fault.

### Smaller fixes

- It is now possible to overload `eq` when using `nomethod`.
- Various problems using `overload` with 64 bit integers corrected.

- The reference count of `PerlIO` file descriptors is now correctly handled.
- On VMS, escaped dots will be preserved when converted to Unix syntax.
- `keys %+` no longer throws an `'ambiguous'` warning.
- Using `#!perl -d` could trigger an assertion, which has been fixed.
- Don't stringify tied code references in `@INC` when calling `require`.
- Code references in `@INC` report the correct file name when `__FILE__` is used.
- Width and precision in sprintf didn't handle characters above 255 correctly. [RT #40473]
- List slices with indices out of range now work more consistently. [RT #39882]
- A change introduced with perl 5.8.1 broke the parsing of arguments of the form `-foo=bar` with the `-s` on the <#!> line. This has been fixed. See http://bugs.activestate.com/show_bug.cgi?id=43483
- `tr///` is now threadsafe. Previously it was storing a swash inside its OP, rather than in a pad.
- *pod2html* labels anchors more consistently and handles nested definition lists better.
- `threads` cleanup veto has been extended to include `perl_free()` and `perl_destruct()`
- On some systems, changes to `$ENV{TZ}` would not always be respected by the underlying calls to `localtime_r()`. Perl now forces the inspection of the environment on these systems.
- The special variable `$^R` is now more consistently set when executing regexps using the `(?{...})` construct. In particular, it will still be set even if backreferences or optional sub-patterns `(?:...)?` are used.

## New or Changed Diagnostics

**panic: sv_chop** %s

This new fatal error occurs when the C routine `Perl_sv_chop()` was passed a position that is not within the scalar's string buffer. This is caused by buggy XS code, and at this point recovery is not possible.

**Maximal count of pending signals (%s) exceeded**

This new fatal error occurs when the perl process has to abort due to too many pending signals, which is bound to prevent perl from being able to handle further incoming signals safely.

**panic: attempt to call** %s **in** %s

This new fatal error occurs when the ACL version file test operator is used where it is not available on the current platform. Earlier checks mean that it should never be possible to get this.

**FETCHSIZE returned a negative value**

New error indicating that a tied array has claimed to have a negative number of elements.

**Can't upgrade** %s **(%d) to** %d

Previously the internal error from the SV upgrade code was the less informative *Can't upgrade that kind of scalar*. It now reports the current internal type, and the new type requested.

%s **argument is not a HASH or ARRAY element or a subroutine**

This error, thrown if an invalid argument is provided to `exists` now correctly includes ''or a subroutine''. [RT #38955]

**Cannot make the non-overridable builtin** %s **fatal**

This error in `Fatal` previously did not show the name of the builtin in question (now represented by %s above).

**Unrecognized character '%s' in column** %d

This error previously did not state the column.

**Offset outside string**

This can now also be generated by a `seek` on a file handle using `PerlIO::scalar`.

**Invalid escape in the specified encoding in regexp; marked by <-- HERE in m/%s/**

New error, introduced as part of the fix to RT #40641 to handle encoding of Unicode characters in regular expression comments.

**Your machine doesn't support dump/undump.**

A more informative fatal error issued when calling `dump` on Win32 and Cygwin. (Given that the purpose of `dump` is to abort with a core dump, and core dumps can't be produced on these platforms, this is more useful than silently exiting.)

## Changed Internals

The perl sources can now be compiled with a C++ compiler instead of a C compiler. A necessary implementation details is that under C++, the macro `XS` used to define XSUBs now includes an `extern "C"` definition. A side effect of this is that **C++** code that used the construction

```
typedef XS(SwigPerlWrapper);
```

now needs to be written

```
typedef XSPROTO(SwigPerlWrapper);
```

using the new `XSPROTO` macro, in order to compile. C extensions are unaffected, although C extensions are encouraged to use `XSPROTO` too. This change was present in the 5.10.0 release of perl, so any actively maintained code that happened to use this construction should already have been adapted. Code that needs changing will fail with a compilation error.

`set` magic on localizing/assigning to a magic variable will now only trigger for *container magics*, i.e. it will for `%ENV` or `%SIG` but not for `$#array`.

The new API macro `newSVpvs()` can be used in place of constructions such as `newSVpvn("ISA", 3)`. It takes a single string constant, and at C compile time determines its length.

The new API function `Perl_newSV_type()` can be used as a more efficient replacement of the common idiom

```
sv = newSV(0);
sv_upgrade(sv, type);
```

Similarly `Perl_newSVpvn_flags()` can be used to combine `Perl_newSVpv()` with `Perl_sv_2mortal()` or the equivalent `Perl_sv_newmortal()` with `Perl_sv_setpvn()`

Two new macros `mPUSHs()` and `mXPUSHs()` are added, to make it easier to push mortal SVs onto the stack. They were then used to fix several bugs where values on the stack had not been mortalised.

A `Perl_signbit()` function was added to test the sign of an `NV`. It maps to the system one when available.

`Perl_av_reify()`, `Perl_lex_end()`, `Perl_mod()`, `Perl_op_clear()`, `Perl_pop_return()`, `Perl_qerror()`, `Perl_setdefout()`, `Perl_vivify_defelem()` and `Perl_yylex()` are now visible to extensions. This was required to allow `Data::Alias` to work on Windows.

`Perl_find_runcv()` is now visible to perl core extensions. This was required to allow `Sub::Current` to work on Windows.

`ptr_table*` functions are now available in unthreaded perl. `Storable` takes advantage of this.

There have been many small cleanups made to the internals. In particular, `Perl_sv_upgrade()` has been simplified considerably, with a straight-through code path that uses `memset()` and `memcpy()` to initialise the new body, rather than assignment via multiple temporary variables. It has also benefited from simplification and de-duplication of the arena management code.

A lot of small improvements in the code base were made due to reports from the Coverity static code analyzer.

Corrected use and documentation of `Perl_gv_stashpv()`, `Perl_gv_stashpvn()`, `Perl_gv_stashsv()` functions (last parameter is a bitmask, not boolean).

`PERL_SYS_INIT`, `PERL_SYS_INIT3` and `PERL_SYS_TERM` macros have been changed into functions.

`PERLSYS_TERM` no longer requires a context. `PerlIO_teardown()` is now called without a context, and debugging output in this function has been disabled because that required that an interpreter was present, an invalid assumption at termination time.

All compile time options which affect binary compatibility have been grouped together into a global variable (`PL_bincompat_options`).

The values of `PERL_REVISION`, `PERL_VERSION` and `PERL_SUBVERSION` are now baked into global variables (and hence into any shared perl library). Additionally under `MULTIPLICITY`, the perl executable now records the size of the interpreter structure (total, and for this version). Coupled with `PL_bincompat_options` this will allow 5.8.10 (and later), when compiled with a shared perl library, to perform sanity checks in `main()` to verify that the shared library is indeed binary compatible.

Symbolic references can now have embedded NULs. The new public function `Perl_get_cvn_flags()` can be used in extensions if you have to handle them.

### Macro cleanups

The core code, and XS code in *ext* that is not dual-lived on CPAN, no longer uses the macros `PL_na`, `NEWSV()`, `Null()`, `Nullav`, `Nullcv`, `Nullhv`, `Nullhv` *etc*. Their use is discouraged in new code, particularly `PL_na`, which is a small performance hit.

## New Tests

Many modules updated from CPAN incorporate new tests. Some core specific tests have been added:

ext/DynaLoader/t/DynaLoader.t
: Tests for the `DynaLoader` module.

t/comp/fold.t
: Tests for compile-time constant folding.

t/io/pvbm.t
: Tests incorporated from 5.10.0 which check that there is no unexpected interaction between the internal types `PVBM` and `PVGV`.

t/lib/proxy_constant_subs.t
: Tests for the new form of constant subroutines.

t/op/attrhand.t
: Tests for `Attribute::Handlers`.

t/op/dbm.t
: Tests for `dbmopen`.

t/op/inccode−tie.t
: Calls all tests in *t/op/inccode.t* after first tying `@INC`.

t/op/incfilter.t
: Tests for source filters returned from code references in `@INC`.

t/op/kill0.t
: Tests for RT #30970.

t/op/qrstack.t
: Tests for RT #41484.

t/op/qr.t
: Tests for the `qr//` construct.

t/op/regexp_qr_embed.t
: Tests for the `qr//` construct within another regexp.

t/op/regexp_qr.t
: Tests for the `qr//` construct.

t/op/rxcode.t
: Tests for RT #32840.

t/op/studytied.t
: Tests for `study` on tied scalars.

t/op/substT.t
: Tests for `subst` run under `−T` mode.

t/op/symbolcache.t
: Tests for `undef` and `delete` on stash entries that are bound to subroutines or methods.

t/op/upgrade.t
: Tests for `Perl_sv_upgrade()`.

t/mro/package_aliases.t
: MRO tests for `isa` and package aliases.

t/pod/twice.t
: Tests for calling `Pod::Parser` twice.

t/run/cloexec.t
: Tests for inheriting file descriptors across `exec` (close-on-exec).

t/uni/cache.t
: Tests for the UTF-8 caching code.

t/uni/chr.t
: Test that strange encodings do not upset `Perl_pp_chr()`.

t/uni/greek.t
: Tests for RT #40641.

t/uni/latin2.t
: Tests for RT #40641.

t/uni/overload.t
: Tests for returning Unicode from overloaded values.

t/uni/tie.t
: Tests for returning Unicode from tied variables.

## Known Problems

There are no known new bugs.

However, programs that rely on bugs that have been fixed will have problems. Also, many bug fixes present in 5.10.0 can't be back-ported to the 5.8.x branch, because they require changes that are binary incompatible, or because the code changes are too large and hence too risky to incorporate.

We have only limited volunteer labour, and the maintenance burden is getting increasingly complex. Hence this will be the last significant release of the 5.8.x series. Any future releases of 5.8.x will likely only be to deal with security issues, and platform build failures. Hence you should look to migrating to 5.10.x, if you have not started already. Alternatively, if business requirements constrain you to continue to use 5.8.x, you may wish to consider commercial support from firms such as ActiveState.

## Platform Specific Notes

### Win32

`readdir()`, `cwd()`, `$^X` and `@INC` now use the alternate (short) filename if the long name is outside the current codepage (Jan Dubois).

*Updated Modules*

- `Win32` upgraded to version 0.38. Now has a documented 'WinVista' response from `GetOSName` and support for Vista's privilege elevation in `IsAdminUser`. Support for Unicode characters in path names. Improved cygwin and Win64 compatibility.
- `Win32API` updated to 0.1001_01
- `killpg()` support added to `MSWin32` (Jan Dubois).
- `File::Spec::Win32` upgraded to version 3.2701

### OS/2

*Updated Modules*

- `OS2::Process` upgraded to 1.03

  Ilya Zakharevich has added and documented several `Window*` and `Clipbrd*` functions.

- `OS2::REXX::DLL`, `OS2::REXX` updated to version 1.03

#### VMS

*Updated Modules*

- `DCLsym` upgraded to version 1.03
- `Stdio` upgraded to version 2.4
- `VMS::XSSymSet` upgraded to 1.1.

## Obituary

Nick Ing-Simmons, long time Perl hacker, author of the `Tk` and `Encode` modules, *perlio.c* in the core, and 5.003_02 pumpking, died of a heart attack on 25th September 2006. He will be missed.

## Acknowledgements

Some of the work in this release was funded by a TPF grant.

Steve Hay worked behind the scenes working out the causes of the differences between core modules, their CPAN releases, and previous core releases, and the best way to rectify them. He doesn't want to do it again. I know this feeling, and I'm very glad he did it this time, instead of me.

Paul Fenwick assembled a team of 18 volunteers, who broke the back of writing this document. In particular, Bradley Dean, Eddy Tan, and Vincent Pit provided half the team's contribution.

Schwern verified the list of updated module versions, correcting quite a few errors that I (and everyone else) had missed, both wrongly stated module versions, and changed modules that had not been listed.

The crack Berlin-based QA team of Andreas König and Slaven Rezic tirelessly re-built snapshots, tested most everything CPAN against them, and then identified the changes responsible for any module regressions, ensuring that several show-stopper bugs were stomped before the first release candidate was cut.

The other core committers contributed most of the changes, and applied most of the patches sent in by the hundreds of contributors listed in *AUTHORS*.

And obviously, Larry Wall, without whom we wouldn't have Perl.

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org. There may also be information at http://www.perl.org, the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team. You can browse and search the Perl 5 bugs at http://bugs.perl.org/

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5−security−report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who will be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## NAME

perl58delta − what is new for perl v5.8.0

## DESCRIPTION

This document describes differences between the 5.6.0 release and the 5.8.0 release.

Many of the bug fixes in 5.8.0 were already seen in the 5.6.1 maintenance release since the two releases were kept closely coordinated (while 5.8.0 was still called 5.7.something).

Changes that were integrated into the 5.6.1 release are marked `[561]`. Many of these changes have been further developed since 5.6.1 was released, those are marked `[561+]`.

You can see the list of changes in the 5.6.1 release (both from the 5.005_03 release and the 5.6.0 release) by reading perl561delta.

## Highlights In 5.8.0

- Better Unicode support
- New IO Implementation
- New Thread Implementation
- Better Numeric Accuracy
- Safe Signals
- Many New Modules
- More Extensive Regression Testing

## Incompatible Changes

### Binary Incompatibility

**Perl 5.8 is not binary compatible with earlier releases of Perl.**

**You have to recompile your XS modules.**

(Pure Perl modules should continue to work.)

The major reason for the discontinuity is the new IO architecture called PerlIO. PerlIO is the default configuration because without it many new features of Perl 5.8 cannot be used. In other words: you just have to recompile your modules containing XS code, sorry about that.

In future releases of Perl, non-PerlIO aware XS modules may become completely unsupported. This shouldn't be too difficult for module authors, however: PerlIO has been designed as a drop-in replacement (at the source code level) for the stdio interface.

Depending on your platform, there are also other reasons why we decided to break binary compatibility, please read on.

### 64−bit platforms and malloc

If your pointers are 64 bits wide, the Perl malloc is no longer being used because it does not work well with 8−byte pointers. Also, usually the system mallocs on such platforms are much better optimized for such large memory models than the Perl malloc. Some memory-hungry Perl applications like the PDL don't work well with Perl's malloc. Finally, other applications than Perl (such as mod_perl) tend to prefer the system malloc. Such platforms include Alpha and 64−bit HPPA, MIPS, PPC, and Sparc.

### AIX Dynaloading

The AIX dynaloading now uses in AIX releases 4.3 and newer the native dlopen interface of AIX instead of the old emulated interface. This change will probably break backward compatibility with compiled modules. The change was made to make Perl more compliant with other applications like mod_perl which are using the AIX native interface.

### Attributes for `my` variables now handled at run-time

The `my EXPR : ATTRS` syntax now applies variable attributes at run-time. (Subroutine and `our` variables still get attributes applied at compile-time.) See attributes for additional details. In particular, however, this allows variable attributes to be useful for `tie` interfaces, which was a deficiency of earlier releases. Note that the new semantics doesn't work with the Attribute::Handlers module (as of version 0.76).

### Socket Extension Dynamic in VMS

The Socket extension is now dynamically loaded instead of being statically built in. This may or may not be a problem with ancient TCP/IP stacks of VMS: we do not know since we weren't able to test Perl in such configurations.

### IEEE-format Floating Point Default on OpenVMS Alpha

Perl now uses IEEE format (T_FLOAT) as the default internal floating point format on OpenVMS Alpha, potentially breaking binary compatibility with external libraries or existing data. G_FLOAT is still available as a configuration option. The default on VAX (D_FLOAT) has not changed.

### New Unicode Semantics (no more `use utf8`, almost)

Previously in Perl 5.6 to use Unicode one would say ''use utf8'' and then the operations (like string concatenation) were Unicode-aware in that lexical scope.

This was found to be an inconvenient interface, and in Perl 5.8 the Unicode model has completely changed: now the ''Unicodeness'' is bound to the data itself, and for most of the time ''use utf8'' is not needed at all. The only remaining use of ''use utf8'' is when the Perl script itself has been written in the UTF-8 encoding of Unicode. (UTF-8 has not been made the default since there are many Perl scripts out there that are using various national eight-bit character sets, which would be illegal in UTF-8.)

See perluniintro for the explanation of the current model, and utf8 for the current use of the utf8 pragma.

### New Unicode Properties

Unicode *scripts* are now supported. Scripts are similar to (and superior to) Unicode *blocks*. The difference between scripts and blocks is that scripts are the glyphs used by a language or a group of languages, while the blocks are more artificial groupings of (mostly) 256 characters based on the Unicode numbering.

In general, scripts are more inclusive, but not universally so. For example, while the script `Latin` includes all the Latin characters and their various diacritic-adorned versions, it does not include the various punctuation or digits (since they are not solely `Latin`).

A number of other properties are now supported, including `\p{L&}`, `\p{Any}` `\p{Assigned}`, `\p{Unassigned}`, `\p{Blank}` [561] and `\p{SpacePerl}` [561] (along with their `\P{...}` versions, of course). See perlunicode for details, and more additions.

The `In` or `Is` prefix to names used with the `\p{...}` and `\P{...}` are now almost always optional. The only exception is that a `In` prefix is required to signify a Unicode block when a block name conflicts with a script name. For example, `\p{Tibetan}` refers to the script, while `\p{InTibetan}` refers to the block. When there is no name conflict, you can omit the `In` from the block name (e.g. `\p{BraillePatterns}`), but to be safe, it's probably best to always use the `In`).

### REF(...) Instead Of SCALAR(...)

A reference to a reference now stringifies as ''REF(0x81485ec)'' instead of ''SCALAR(0x81485ec)'' in order to be more consistent with the return value of **ref()**.

### pack/unpack D/F recycled

The undocumented pack/unpack template letters D/F have been recycled for better use: now they stand for long double (if supported by the platform) and NV (Perl internal floating point type). (They used to be aliases for d/f, but you never knew that.)

### glob() now returns filenames in alphabetical order

The list of filenames from **glob()** (or <...>) is now by default sorted alphabetically to be csh-compliant (which is what happened before in most Unix platforms). (**bsd_glob()** does still sort platform natively, ASCII or EBCDIC, unless GLOB_ALPHASORT is specified.) [561]

### Deprecations

- The semantics of bless(REF, REF) were unclear and until someone proves it to make some sense, it is forbidden.
- The obsolete chat2 library that should never have been allowed to escape the laboratory has been decommissioned.

- Using chdir("") or chdir(undef) instead of explicit **chdir()** is doubtful. A failure (think chdir(**some_function()**) can lead into unintended **chdir()** to the home directory, therefore this behaviour is deprecated.
- The builtin **dump()** function has probably outlived most of its usefulness. The core-dumping functionality will remain in future available as an explicit call to `CORE::dump()`, but in future releases the behaviour of an unqualified `dump()` call may change.
- The very dusty examples in the eg/ directory have been removed. Suggestions for new shiny examples welcome but the main issue is that the examples need to be documented, tested and (most importantly) maintained.
- The (bogus) escape sequences \8 and \9 now give an optional warning (''Unrecognized escape passed through''). There is no need to \−escape any `\w` character.
- The *glob{FILEHANDLE} is deprecated, use *glob{IO} instead.
- The `package;` syntax (`package` without an argument) has been deprecated. Its semantics were never that clear and its implementation even less so. If you have used that feature to disallow all but fully qualified variables, `use strict;` instead.
- The unimplemented POSIX regex features [[.cc.]] and [[=c=]] are still recognised but now cause fatal errors. The previous behaviour of ignoring them by default and warning if requested was unacceptable since it, in a way, falsely promised that the features could be used.
- In future releases, non-PerlIO aware XS modules may become completely unsupported. Since PerlIO is a drop-in replacement for stdio at the source code level, this shouldn't be that drastic a change.
- Previous versions of perl and some readings of some sections of Camel III implied that the `:raw` ''discipline'' was the inverse of `:crlf`. Turning off ''clrfness'' is no longer enough to make a stream truly binary. So the PerlIO `:raw` layer (or ''discipline'', to use the Camel book's older terminology) is now formally defined as being equivalent to binmode(FH) − which is in turn defined as doing whatever is necessary to pass each byte as-is without any translation. In particular binmode(FH) − and hence `:raw` − will now turn off both CRLF and UTF−8 translation and remove other layers (e.g. :**encoding()**) which would modify byte stream.
- The current user-visible implementation of pseudo-hashes (the weird use of the first array element) is deprecated starting from Perl 5.8.0 and will be removed in Perl 5.10.0, and the feature will be implemented differently. Not only is the current interface rather ugly, but the current implementation slows down normal array and hash use quite noticeably. The `fields` pragma interface will remain available. The *restricted hashes* interface is expected to be the replacement interface (see Hash::Util). If your existing programs depends on the underlying implementation, consider using Class::PseudoHash from CPAN.
- The syntaxes `@a−>[...]` and `%h−>{...}` have now been deprecated.
- After years of trying, suidperl is considered to be too complex to ever be considered truly secure. The suidperl functionality is likely to be removed in a future release.
- The 5.005 threads model (module `Thread`) is deprecated and expected to be removed in Perl 5.10. Multithreaded code should be migrated to the new ithreads model (see threads, threads::shared and perlthrtut).
- The long deprecated uppercase aliases for the string comparison operators (EQ, NE, LT, LE, GE, GT) have now been removed.
- The tr///C and tr///U features have been removed and will not return; the interface was a mistake. Sorry about that. For similar functionality, see pack('U0', ...) and pack('C0', ...). [561]
- Earlier Perls treated ''sub foo (@bar)'' as equivalent to ''sub foo (@)''. The prototypes are now checked better at compile-time for invalid syntax. An optional warning is generated (''Illegal character in prototype...'') but this may be upgraded to a fatal error in a future release.
- The `exec LIST` and `system LIST` operations now produce warnings on tainted data and in some future release they will produce fatal errors.

- The existing behaviour when localising tied arrays and hashes is wrong, and will be changed in a future release, so do not rely on the existing behaviour. See ''Localising Tied Arrays and Hashes Is Broken''.

## Core Enhancements

### Unicode Overhaul

Unicode in general should be now much more usable than in Perl 5.6.0 (or even in 5.6.1). Unicode can be used in hash keys, Unicode in regular expressions should work now, Unicode in tr/// should work now, Unicode in I/O should work now. See perluniintro for introduction and perlunicode for details.

- The Unicode Character Database coming with Perl has been upgraded to Unicode 3.2.0. For more information, see http://www.unicode.org/ . [561+] (5.6.1 has UCD 3.0.1.)
- For developers interested in enhancing Perl's Unicode capabilities: almost all the UCD files are included with the Perl distribution in the *lib/unicore* subdirectory. The most notable omission, for space considerations, is the Unihan database.
- The properties \p{Blank} and \p{SpacePerl} have been added. ''Blank'' is like C **isblank()**, that is, it contains only ''horizontal whitespace'' (the space character is, the newline isn't), and the ''SpacePerl'' is the Unicode equivalent of `\s` (\p{Space} isn't, since that includes the vertical tabulator character, whereas `\s` doesn't.)

  See ''New Unicode Properties'' earlier in this document for additional information on changes with Unicode properties.

### PerlIO is Now The Default

- IO is now by default done via PerlIO rather than system's ''stdio''. PerlIO allows ''layers'' to be ''pushed'' onto a file handle to alter the handle's behaviour. Layers can be specified at open time via 3−arg form of open:

  ```
  open($fh,'>:crlf :utf8', $path) || ...
  ```

  or on already opened handles via extended `binmode`:

  ```
  binmode($fh,':encoding(iso−8859−7)');
  ```

  The built-in layers are: unix (low level read/write), stdio (as in previous Perls), perlio (re-implementation of stdio buffering in a portable manner), crlf (does CRLF <=> ''\n'' translation as on Win32, but available on any platform). A mmap layer may be available if platform supports it (mostly Unixes).

  Layers to be applied by default may be specified via the 'open' pragma.

  See ''Installation and Configuration Improvements'' for the effects of PerlIO on your architecture name.

- If your platform supports **fork()**, you can use the list form of `open` for pipes. For example:

  ```
  open KID_PS, "−|", "ps", "aux" or die $!;
  ```

  forks the **ps**(1) command (without spawning a shell, as there are more than three arguments to **open()**), and reads its standard output via the `KID_PS` filehandle. See perlipc.

- File handles can be marked as accepting Perl's internal encoding of Unicode (UTF−8 or UTF-EBCDIC depending on platform) by a pseudo layer ''':utf8''' :

  ```
  open($fh,">:utf8","Uni.txt");
  ```

  Note for EBCDIC users: the pseudo layer ''':utf8''' is erroneously named for you since it's not UTF−8 what you will be getting but instead UTF-EBCDIC. See perlunicode, utf8, and http://www.unicode.org/reports/tr16/ for more information. In future releases this naming may change. See perluniintro for more information about UTF−8.

- If your environment variables (LC_ALL, LC_CTYPE, LANG) look like you want to use UTF−8 (any of the variables match `/utf−?8/i`), your STDIN, STDOUT, STDERR handles and the default open layer (see open) are marked as UTF−8. (This feature, like other new features that combine Unicode and I/O, work only if you are using PerlIO, but that's the default.)

  Note that after this Perl really does assume that everything is UTF−8: for example if some input

handle is not, Perl will probably very soon complain about the input data like this ‘‘Malformed UTF−8 ...’’ since any old eight-bit data is not legal UTF−8.

Note for code authors: if you want to enable your users to use UTF−8 as their default encoding but in your code still have eight-bit I/O streams (such as images or zip files), you need to explicitly **open()** or **binmode()** with `:bytes` (see ‘‘open’’ in perlfunc and ‘‘binmode’’ in perlfunc), or you can just use `binmode(FH)` (nice for pre−5.8.0 backward compatibility).

- File handles can translate character encodings from/to Perl’s internal Unicode form on read/write via the ‘‘:**encoding()**’’ layer.
- File handles can be opened to ‘‘in memory’’ files held in Perl scalars via:

```
open($fh,'>', \$variable) || ...
```

- Anonymous temporary files are available without need to ’use FileHandle’ or other module via

```
open($fh,"+>", undef) || ...
```

That is a literal undef, not an undefined value.

### ithreads

The new interpreter threads (‘‘ithreads’’ for short) implementation of multithreading, by Arthur Bergman, replaces the old ‘‘5.005 threads’’ implementation. In the ithreads model any data sharing between threads must be explicit, as opposed to the model where data sharing was implicit. See threads and threads::shared, and perlthrtut.

As a part of the ithreads implementation Perl will also use any necessary and detectable reentrant libc interfaces.

### Restricted Hashes

A restricted hash is restricted to a certain set of keys, no keys outside the set can be added. Also individual keys can be restricted so that the key cannot be deleted and the value cannot be changed. No new syntax is involved: the Hash::Util module is the interface.

### Safe Signals

Perl used to be fragile in that signals arriving at inopportune moments could corrupt Perl’s internal state. Now Perl postpones handling of signals until it’s safe (between opcodes).

This change may have surprising side effects because signals no longer interrupt Perl instantly. Perl will now first finish whatever it was doing, like finishing an internal operation (like **sort()**) or an external operation (like an I/O operation), and only then look at any arrived signals (and before starting the next operation). No more corrupt internal state since the current operation is always finished first, but the signal may take more time to get heard. Note that breaking out from potentially blocking operations should still work, though.

### Understanding of Numbers

In general a lot of fixing has happened in the area of Perl’s understanding of numbers, both integer and floating point. Since in many systems the standard number parsing functions like `strtoul()` and `atof()` seem to have bugs, Perl tries to work around their deficiencies. This results hopefully in more accurate numbers.

Perl now tries internally to use integer values in numeric conversions and basic arithmetics (+ − * /) if the arguments are integers, and tries also to keep the results stored internally as integers. This change leads to often slightly faster and always less lossy arithmetics. (Previously Perl always preferred floating point numbers in its math.)

### Arrays now always interpolate into double-quoted strings [561]

In double-quoted strings, arrays now interpolate, no matter what. The behavior in earlier versions of perl 5 was that arrays would interpolate into strings if the array had been mentioned before the string was compiled, and otherwise Perl would raise a fatal compile-time error. In versions 5.000 through 5.003, the error was

```
Literal @example now requires backslash
```

In versions 5.004_01 through 5.6.0, the error was

```
In string, @example now must be written as \@example
```

The idea here was to get people into the habit of writing `"fred\@example.com"` when they

wanted a literal @ sign, just as they have always written `"Give me back my \$5"` when they wanted a literal $ sign.

Starting with 5.6.1, when Perl now sees an @ sign in a double-quoted string, it *always* attempts to interpolate an array, regardless of whether or not the array has been used or declared already. The fatal error has been downgraded to an optional warning:

```
Possible unintended interpolation of @example in string
```

This warns you that `"fred@example.com"` is going to turn into `fred.com` if you don't backslash the @. See http://perl.plover.com/at-error.html for more details about the history here.

### Miscellaneous Changes

- AUTOLOAD is now lvaluable, meaning that you can add the :lvalue attribute to AUTOLOAD subroutines and you can assign to the AUTOLOAD return value.
- The `$Config{byteorder}` (and corresponding BYTEORDER in config.h) was previously wrong in platforms if sizeof(long) was 4, but sizeof(IV) was 8. The byteorder was only sizeof(long) bytes long (1234 or 4321), but now it is correctly sizeof(IV) bytes long, (12345678 or 87654321). (This problem didn't affect Windows platforms.)

  Also, `$Config{byteorder}` is now computed dynamically — this is more robust with ''fat binaries'' where an executable image contains binaries for more than one binary platform, and when cross-compiling.
- `perl -d:Module=arg,arg,arg` now works (previously one couldn't pass in multiple arguments.)
- `do` followed by a bareword now ensures that this bareword isn't a keyword (to avoid a bug where `do q(foo.pl)` tried to call a subroutine called `q`). This means that for example instead of `do format()` you must write `do &format()`.
- The builtin **dump()** now gives an optional warning `dump() better written as CORE::dump()`, meaning that by default `dump(...)` is resolved as the builtin **dump()** which dumps core and aborts, not as (possibly) user-defined `sub dump`. To call the latter, qualify the call as `&dump(...)`. (The whole **dump()** feature is to considered deprecated, and possibly removed/changed in future releases.)
- **chomp()** and **chop()** are now overridable. Note, however, that their prototype (as given by `prototype("CORE::chomp")`) is undefined, because it cannot be expressed and therefore one cannot really write replacements to override these builtins.
- END blocks are now run even if you exit/die in a BEGIN block. Internally, the execution of END blocks is now controlled by PL_exit_flags & PERL_EXIT_DESTRUCT_END. This enables the new behaviour for Perl embedders. This will default in 5.10. See perlembed.
- Formats now support zero-padded decimal fields.
- Although ''you shouldn't do that'', it was possible to write code that depends on Perl's hashed key order (Data::Dumper does this). The new algorithm ''One-at-a-Time'' produces a different hashed key order. More details are in ''Performance Enhancements''.
- lstat(FILEHANDLE) now gives a warning because the operation makes no sense. In future releases this may become a fatal error.
- Spurious syntax errors generated in certain situations, when **glob()** caused File::Glob to be loaded for the first time, have been fixed. [561]
- Lvalue subroutines can now return `undef` in list context. However, the lvalue subroutine feature still remains experimental. [561+]
- A lost warning ''Can't declare ... dereference in my'' has been restored (Perl had it earlier but it became lost in later releases.)
- A new special regular expression variable has been introduced: `$^N`, which contains the most-recently closed group (submatch).
- `no Module;` does not produce an error even if Module does not have an **unimport()** method. This parallels the behavior of `use` vis-a-vis `import`. [561]

- The numerical comparison operators return `undef` if either operand is a NaN. Previously the behaviour was unspecified.
- `our` can now have an experimental optional attribute `unique` that affects how global variables are shared among multiple interpreters, see ‘‘our’’ in perlfunc.
- The following builtin functions are now overridable: **each()**, **keys()**, **pop()**, **push()**, **shift()**, **splice()**, **unshift()**. [561]
- `pack() / unpack()` can now group template letters with `()` and then apply repetition/count modifiers on the groups.
- `pack() / unpack()` can now process the Perl internal numeric types: IVs, UVs, NVs — and also long doubles, if supported by the platform. The template letters are `j`, `J`, `F`, and `D`.
- `pack('U0a*', ...)` can now be used to force a string to UTF−8.
- my __PACKAGE__ `$obj` now works. [561]
- **POSIX::sleep()** now returns the number of *unslept* seconds (as the POSIX standard says), as opposed to **CORE::sleep()** which returns the number of slept seconds.
- **printf()** and **sprintf()** now support parameter reordering using the `%\d+\$` and `*\d+\$` syntaxes. For example

  ```
  printf "%2\$s %1\$s\n", "foo", "bar";
  ```

  will print ‘‘bar foo\n’’. This feature helps in writing internationalised software, and in general when the order of the parameters can vary.
- The (\&) prototype now works properly. [561]
- prototype(\[$@%&]) is now available to implicitly create references (useful for example if you want to emulate the **tie()** interface).
- A new command-line option, `−t` is available. It is the little brother of `−T`: instead of dying on taint violations, lexical warnings are given. **This is only meant as a temporary debugging aid while securing the code of old legacy applications. This is not a substitute for −T.**
- In other taint news, the `exec LIST` and `system LIST` have now been considered too risky (think `exec @ARGV`: it can start any program with any arguments), and now the said forms cause a warning under lexical warnings. You should carefully launder the arguments to guarantee their validity. In future releases of Perl the forms will become fatal errors so consider starting laundering now.
- Tied hash interfaces are now required to have the EXISTS and DELETE methods (either own or inherited).
- If tr/// is just counting characters, it doesn’t attempt to modify its target.
- **untie()** will now call an **UNTIE()** hook if it exists. See perltie for details. [561]
- ‘‘utime’’ in perlfunc now supports `utime undef, undef, @files` to change the file timestamps to the current time.
- The rules for allowing underscores (underbars) in numeric constants have been relaxed and simplified: now you can have an underscore simply **between digits**.
- Rather than relying on C’s argv[0] (which may not contain a full pathname) where possible $^X is now set by asking the operating system. (eg by reading */proc/self/exe* on Linux, */proc/curproc/file* on FreeBSD)
- A new variable, `${^TAINT}`, indicates whether taint mode is enabled.
- You can now override the **readline()** builtin, and this overrides also the <FILEHANDLE> angle bracket operator.
- The command-line options −s and −F are now recognized on the shebang (#!) line.
- Use of the `/c` match modifier without an accompanying `/g` modifier elicits a new warning: `Use of /c modifier is meaningless without /g`.

  Use of `/c` in substitutions, even with `/g`, elicits `Use of /c modifier is meaningless`

`in s///`.

Use of `/g` with `split` elicits `Use of /g modifier is meaningless in split`.

- Support for the `CLONE` special subroutine had been added. With ithreads, when a new thread is created, all Perl data is cloned, however non-Perl data cannot be cloned automatically. In `CLONE` you can do whatever you need to do, like for example handle the cloning of non-Perl data, if necessary. `CLONE` will be executed once for every package that has it defined or inherited. It will be called in the context of the new thread, so all modifications are made in the new area.

  See perlmod

## Modules and Pragmata

### New Modules and Pragmata

- `Attribute::Handlers`, originally by Damian Conway and now maintained by Arthur Bergman, allows a class to define attribute handlers.

```
package MyPack;
use Attribute::Handlers;
sub Wolf :ATTR(SCALAR) { print "howl!\n" }

# later, in some package using or inheriting from MyPack...

my MyPack $Fluffy : Wolf; # the attribute handler Wolf will be called
```

  Both variables and routines can have attribute handlers. Handlers can be specific to type (SCALAR, ARRAY, HASH, or CODE), or specific to the exact compilation phase (BEGIN, CHECK, INIT, or END). See Attribute::Handlers.

- `B::Concise`, by Stephen McCamant, is a new compiler backend for walking the Perl syntax tree, printing concise info about ops. The output is highly customisable. See B::Concise. [561+]

- The new bignum, bigint, and bigrat pragmas, by Tels, implement transparent bignum support (using the Math::BigInt, Math::BigFloat, and Math::BigRat backends).

- `Class::ISA`, by Sean Burke, is a module for reporting the search path for a class's ISA tree. See Class::ISA.

- `Cwd` now has a split personality: if possible, an XS extension is used, (this will hopefully be faster, more secure, and more robust) but if not possible, the familiar Perl implementation is used.

- `Devel::PPPort`, originally by Kenneth Albanowski and now maintained by Paul Marquess, has been added. It is primarily used by `h2xs` to enhance portability of XS modules between different versions of Perl. See Devel::PPPort.

- `Digest`, frontend module for calculating digests (checksums), from Gisle Aas, has been added. See Digest.

- `Digest::MD5` for calculating MD5 digests (checksums) as defined in RFC 1321, from Gisle Aas, has been added. See Digest::MD5.

```
use Digest::MD5 'md5_hex';

$digest = md5_hex("Thirsty Camel");

print $digest, "\n"; # 01d19d9d2045e005c3f1b80e8b164de1
```

  NOTE: the `MD5` backward compatibility module is deliberately not included since its further use is discouraged.

  See also PerlIO::via::QuotedPrint.

- `Encode`, originally by Nick Ing-Simmons and now maintained by Dan Kogai, provides a mechanism to translate between different character encodings. Support for Unicode, ISO−8859−1, and ASCII are compiled in to the module. Several other encodings (like the rest of the ISO−8859, CP*/Win*, Mac, KOI8−R, three variants EBCDIC, Chinese, Japanese, and Korean encodings) are included and can be loaded at runtime. (For space considerations, the largest Chinese encodings have been separated into their own CPAN module, Encode::HanExtra, which Encode will use if

available). See Encode.

Any encoding supported by Encode module is also available to the ‘‘:**encoding()**’’ layer if PerlIO is used.

- `Hash::Util` is the interface to the new *restricted hashes* feature. (Implemented by Jeffrey Friedl, Nick Ing-Simmons, and Michael Schwern.) See Hash::Util.
- `I18N::Langinfo` can be used to query locale information. See I18N::Langinfo.
- `I18N::LangTags`, by Sean Burke, has functions for dealing with RFC3066−style language tags. See I18N::LangTags.
- `ExtUtils::Constant`, by Nicholas Clark, is a new tool for extension writers for generating XS code to import C header constants. See ExtUtils::Constant.
- `Filter::Simple`, by Damian Conway, is an easy-to-use frontend to Filter::Util::Call. See Filter::Simple.

```
# in MyFilter.pm:

package MyFilter;

use Filter::Simple sub {
    while (my ($from, $to) = splice @_, 0, 2) {
            s/$from/$to/g;
    }
};

1;

# in user's code:

use MyFilter qr/red/ => 'green';

print "red\n";   # this code is filtered, will print "green\n"
print "bored\n"; # this code is filtered, will print "bogreen\n"

no MyFilter;

print "red\n";   # this code is not filtered, will print "red\n"
```

- `File::Temp`, by Tim Jenness, allows one to create temporary files and directories in an easy, portable, and secure way. See File::Temp. [561+]
- `Filter::Util::Call`, by Paul Marquess, provides you with the framework to write *source filters* in Perl. For most uses, the frontend Filter::Simple is to be preferred. See Filter::Util::Call.
- `if`, by Ilya Zakharevich, is a new pragma for conditional inclusion of modules.
- libnet, by Graham Barr, is a collection of perl5 modules related to network programming. See Net::FTP, Net::NNTP, Net::Ping (not part of libnet, but related), Net::POP3, Net::SMTP, and Net::Time.

  Perl installation leaves libnet unconfigured; use *libnetcfg* to configure it.
- `List::Util`, by Graham Barr, is a selection of general-utility list subroutines, such as **sum()**, **min()**, **first()**, and **shuffle()**. See List::Util.
- `Locale::Constants`, `Locale::Country`, `Locale::Currency` `Locale::Language`, and Locale::Script, by Neil Bowers, have been added. They provide the codes for various locale standards, such as ‘‘fr’’ for France, ‘‘usd’’ for US Dollar, and ‘‘ja’’ for Japanese.

```
use Locale::Country;

$country = code2country('jp');                # $country gets 'Japan'
```

```
    $code    = country2code('Norway');           # $code gets 'no'
```

See Locale::Constants, Locale::Country, Locale::Currency, and Locale::Language.

- `Locale::Maketext`, by Sean Burke, is a localization framework. See Locale::Maketext, and Locale::Maketext::TPJ13. The latter is an article about software localization, originally published in The Perl Journal #13, and republished here with kind permission.
- `Math::BigRat` for big rational numbers, to accompany Math::BigInt and Math::BigFloat, from Tels. See Math::BigRat.
- `Memoize` can make your functions faster by trading space for time, from Mark-Jason Dominus. See Memoize.
- `MIME::Base64`, by Gisle Aas, allows you to encode data in base64, as defined in RFC 2045 − *MIME (Multipurpose Internet Mail Extensions)*.

```
    use MIME::Base64;

    $encoded = encode_base64('Aladdin:open sesame');
    $decoded = decode_base64($encoded);

    print $encoded, "\n"; # "QWxhZGRpbjpvcGVuIHNlc2FtZQ=="
```

See MIME::Base64.

- `MIME::QuotedPrint`, by Gisle Aas, allows you to encode data in quoted-printable encoding, as defined in RFC 2045 − *MIME (Multipurpose Internet Mail Extensions)*.

```
    use MIME::QuotedPrint;

    $encoded = encode_qp("\xDE\xAD\xBE\xEF");
    $decoded = decode_qp($encoded);

    print $encoded, "\n"; # "=DE=AD=BE=EF\n"
    print $decoded, "\n"; # "\xDE\xAD\xBE\xEF\n"
```

See also PerlIO::via::QuotedPrint.

- `NEXT`, by Damian Conway, is a pseudo-class for method redispatch. See NEXT.
- `open` is a new pragma for setting the default I/O layers for **open()**.
- `PerlIO::scalar`, by Nick Ing-Simmons, provides the implementation of IO to ‘‘in memory’’ Perl scalars as discussed above. It also serves as an example of a loadable PerlIO layer. Other future possibilities include PerlIO::Array and PerlIO::Code. See PerlIO::scalar.
- `PerlIO::via`, by Nick Ing-Simmons, acts as a PerlIO layer and wraps PerlIO layer functionality provided by a class (typically implemented in Perl code).
- `PerlIO::via::QuotedPrint`, by Elizabeth Mattijsen, is an example of a `PerlIO::via` class:

```
    use PerlIO::via::QuotedPrint;
    open($fh,">:via(QuotedPrint)",$path);
```

This will automatically convert everything output to `$fh` to Quoted-Printable. See PerlIO::via and PerlIO::via::QuotedPrint.

- `Pod::ParseLink`, by Russ Allbery, has been added, to parse L<> links in pods as described in the new perlpodspec.
- `Pod::Text::Overstrike`, by Joe Smith, has been added. It converts POD data to formatted overstrike text. See Pod::Text::Overstrike. [561+]
- `Scalar::Util` is a selection of general-utility scalar subroutines, such as **blessed()**, **reftype()**, and **tainted()**. See Scalar::Util.
- `sort` is a new pragma for controlling the behaviour of **sort()**.

- `Storable` gives persistence to Perl data structures by allowing the storage and retrieval of Perl data to and from files in a fast and compact binary format. Because in effect Storable does serialisation of Perl data structures, with it you can also clone deep, hierarchical datastructures. Storable was originally created by Raphael Manfredi, but it is now maintained by Abhijit Menon-Sen. Storable has been enhanced to understand the two new hash features, Unicode keys and restricted hashes. See Storable.
- `Switch`, by Damian Conway, has been added. Just by saying

  ```
  use Switch;
  ```

  you have `switch` and `case` available in Perl.

  ```
  use Switch;

  switch ($val) {

              case 1          { print "number 1" }
              case "a"        { print "string a" }
              case [1..10,42] { print "number in list" }
              case (@array)   { print "number in list" }
              case /\w+/      { print "pattern" }
              case qr/\w+/    { print "pattern" }
              case (%hash)    { print "entry in hash" }
              case (\%hash)   { print "entry in hash" }
              case (\&sub)    { print "arg to subroutine" }
              else            { print "previous case not true" }
  }
  ```

  See Switch.
- `Test::More`, by Michael Schwern, is yet another framework for writing test scripts, more extensive than Test::Simple. See Test::More.
- `Test::Simple`, by Michael Schwern, has basic utilities for writing tests. See Test::Simple.
- `Text::Balanced`, by Damian Conway, has been added, for extracting delimited text sequences from strings.

  ```
  use Text::Balanced 'extract_delimited';

  ($a, $b) = extract_delimited("'never say never', he never said", "'", '');
  ```

  `$a` will be ‘’never say never’’’, `$b` will be ’, he never said’.

  In addition to **extract_delimited()**, there are also **extract_bracketed()**, **extract_quotelike()**, **extract_codeblock()**, **extract_variable()**, **extract_tagged()**, **extract_multiple()**, **gen_delimited_pat()**, and **gen_extract_tagged()**. With these, you can implement rather advanced parsing algorithms. See Text::Balanced.
- `threads`, by Arthur Bergman, is an interface to interpreter threads. Interpreter threads (ithreads) is the new thread model introduced in Perl 5.6 but only available as an internal interface for extension writers (and for Win32 Perl for `fork()` emulation). See threads, threads::shared, and perlthrtut.
- `threads::shared`, by Arthur Bergman, allows data sharing for interpreter threads. See threads::shared.
- `Tie::File`, by Mark-Jason Dominus, associates a Perl array with the lines of a file. See Tie::File.
- `Tie::Memoize`, by Ilya Zakharevich, provides on-demand loaded hashes. See Tie::Memoize.
- `Tie::RefHash::Nestable`, by Edward Avis, allows storing hash references (unlike the standard Tie::RefHash) The module is contained within Tie::RefHash. See Tie::RefHash.

- `Time::HiRes`, by Douglas E. Wegscheid, provides high resolution timing (ualarm, usleep, and gettimeofday). See Time::HiRes.
- `Unicode::UCD` offers a querying interface to the Unicode Character Database. See Unicode::UCD.
- `Unicode::Collate`, by SADAHIRO Tomoyuki, implements the UCA (Unicode Collation Algorithm) for sorting Unicode strings. See Unicode::Collate.
- `Unicode::Normalize`, by SADAHIRO Tomoyuki, implements the various Unicode normalization forms. See Unicode::Normalize.
- `XS::APItest`, by Tim Jenness, is a test extension that exercises XS APIs. Currently only `printf()` is tested: how to output various basic data types from XS.
- `XS::Typemap`, by Tim Jenness, is a test extension that exercises XS typemaps. Nothing gets installed, but the code is worth studying for extension writers.

### Updated And Improved Modules and Pragmata

- The following independently supported modules have been updated to the newest versions from CPAN: CGI, CPAN, DB_File, File::Spec, File::Temp, Getopt::Long, Math::BigFloat, Math::BigInt, the podlators bundle (Pod::Man, Pod::Text), Pod::LaTeX [561+], Pod::Parser, Storable, Term::ANSIColor, Test, Text−Tabs+Wrap.
- **attributes::reftype()** now works on tied arguments.
- AutoLoader can now be disabled with `no AutoLoader;`.
- B::Deparse has been significantly enhanced by Robin Houston. It can now deparse almost all of the standard test suite (so that the tests still succeed). There is a make target ‘‘test.deparse’’ for trying this out.
- Carp now has better interface documentation, and the `@CARP_NOT` interface has been added to get optional control over where errors are reported independently of `@ISA`, by Ben Tilly.
- Class::Struct can now define the classes in compile time.
- Class::Struct now assigns the array/hash element if the accessor is called with an array/hash element as the **sole** argument.
- The return value of **Cwd::fastcwd()** is now tainted.
- Data::Dumper now has an option to sort hashes.
- Data::Dumper now has an option to dump code references using B::Deparse.
- DB_File now supports newer Berkeley DB versions, among other improvements.
- Devel::Peek now has an interface for the Perl memory statistics (this works only if you are using perl’s malloc, and if you have compiled with debugging).
- The English module can now be used without the infamous performance hit by saying

  ```
  use English '−no_match_vars';
  ```

  (Assuming, of course, that you don’t need the troublesome variables `$``, `$&`, or `$'`.) Also, introduced `@LAST_MATCH_START` and `@LAST_MATCH_END` English aliases for `@−` and `@+`.
- ExtUtils::MakeMaker has been significantly cleaned up and fixed. The enhanced version has also been backported to earlier releases of Perl and submitted to CPAN so that the earlier releases can enjoy the fixes.
- The arguments of **WriteMakefile()** in Makefile.PL are now checked for sanity much more carefully than before. This may cause new warnings when modules are being installed. See ExtUtils::MakeMaker for more details.
- ExtUtils::MakeMaker now uses File::Spec internally, which hopefully leads to better portability.
- Fcntl, Socket, and Sys::Syslog have been rewritten by Nicholas Clark to use the new-style constant dispatch section (see ExtUtils::Constant). This means that they will be more robust and hopefully faster.

- File::Find now **chdir()**s correctly when chasing symbolic links. [561]
- File::Find now has pre− and post-processing callbacks. It also correctly changes directories when chasing symbolic links. Callbacks (naughtily) exiting with ‘‘next;’’ instead of ‘‘return;’’ now work.
- File::Find is now (again) reentrant. It also has been made more portable.
- The warnings issued by File::Find now belong to their own category. You can enable/disable them with `use/no warnings 'File::Find';`.
- **File::Glob::glob()** has been renamed to **File::Glob::bsd_glob()** because the name clashes with the builtin **glob()**. The older name is still available for compatibility, but is deprecated. [561]
- File::Glob now supports `GLOB_LIMIT` constant to limit the size of the returned list of filenames.
- IPC::Open3 now allows the use of numeric file descriptors.
- IO::Socket now has an **atmark()** method, which returns true if the socket is positioned at the out-of-band mark. The method is also exportable as a **sockatmark()** function.
- IO::Socket::INET failed to open the specified port if the service name was not known. It now correctly uses the supplied port number as is. [561]
- IO::Socket::INET has support for the ReusePort option (if your platform supports it). The Reuse option now has an alias, ReuseAddr. For clarity, you may want to prefer ReuseAddr.
- IO::Socket::INET now supports a value of zero for `LocalPort` (usually meaning that the operating system will make one up.)
- ’use lib’ now works identically to `@INC`. Removing directories with ’no lib’ now works.
- Math::BigFloat and Math::BigInt have undergone a full rewrite by Tels. They are now magnitudes faster, and they support various bignum libraries such as GMP and PARI as their backends.
- Math::Complex handles inf, NaN etc., better.
- Net::Ping has been considerably enhanced by Rob Brown: multihoming is now supported, Win32 functionality is better, there is now time measuring functionality (optionally high-resolution using Time::HiRes), and there is now ‘‘external’’ protocol which uses Net::Ping::External module which runs your external ping utility and parses the output. A version of Net::Ping::External is available in CPAN.

  Note that some of the Net::Ping tests are disabled when running under the Perl distribution since one cannot assume one or more of the following: enabled echo port at localhost, full Internet connectivity, or sympathetic firewalls. You can set the environment variable PERL_TEST_Net_Ping to ‘‘1’’ (one) before running the Perl test suite to enable all the Net::Ping tests.
- **POSIX::sigaction()** is now much more flexible and robust. You can now install coderef handlers, ’DEFAULT’, and ’IGNORE’ handlers, installing new handlers was not atomic.
- In Safe, `%INC` is now localised in a Safe compartment so that use/require work.
- In SDBM_File on DOSish platforms, some keys went missing because of lack of support for files with ‘‘holes’’. A workaround for the problem has been added.
- In Search::Dict one can now have a pre-processing hook for the lines being searched.
- The Shell module now has an OO interface.
- In Sys::Syslog there is now a failover mechanism that will go through alternative connection mechanisms until the message is successfully logged.
- The Test module has been significantly enhanced.
- **Time::Local::timelocal()** does not handle fractional seconds anymore. The rationale is that neither does **localtime()**, and **timelocal()** and **localtime()** are supposed to be inverses of each other.
- The vars pragma now supports declaring fully qualified variables. (Something that `our()` does not and will not support.)

- The `utf8::` name space (as in the pragma) provides various Perl-callable functions to provide low level access to Perl's internal Unicode representation. At the moment only **length()** has been implemented.

## Utility Changes

- Emacs perl mode (emacs/cperl−mode.el) has been updated to version 4.31.
- *emacs/e2ctags.pl* is now much faster.
- `enc2xs` is a tool for people adding their own encodings to the Encode module.
- `h2ph` now supports C trigraphs.
- `h2xs` now produces a template README.
- `h2xs` now uses `Devel::PPPort` for better portability between different versions of Perl.
- `h2xs` uses the new ExtUtils::Constant module which will affect newly created extensions that define constants. Since the new code is more correct (if you have two constants where the first one is a prefix of the second one, the first constant **never** got defined), less lossy (it uses integers for integer constant, as opposed to the old code that used floating point numbers even for integer constants), and slightly faster, you might want to consider regenerating your extension code (the new scheme makes regenerating easy). h2xs now also supports C trigraphs.
- `libnetcfg` has been added to configure libnet.
- `perlbug` is now much more robust. It also sends the bug report to perl.org, not perl.com.
- `perlcc` has been rewritten and its user interface (that is, command line) is much more like that of the Unix C compiler, cc. (The perlbc tools has been removed. Use `perlcc -B` instead.) **Note that perlcc is still considered very experimental and unsupported.** [561]
- `perlivp` is a new Installation Verification Procedure utility for running any time after installing Perl.
- `piconv` is an implementation of the character conversion utility `iconv`, demonstrating the new Encode module.
- `pod2html` now allows specifying a cache directory.
- `pod2html` now produces XHTML 1.0.
- `pod2html` now understands POD written using different line endings (PC-like CRLF versus Unix-like LF versus MacClassic-like CR).
- `s2p` has been completely rewritten in Perl. (It is in fact a full implementation of sed in Perl: you can use the sed functionality by using the `psed` utility.)
- `xsubpp` now understands POD documentation embedded in the *.xs files. [561]
- `xsubpp` now supports the OUT keyword.

## New Documentation

- perl56delta details the changes between the 5.005 release and the 5.6.0 release.
- perlclib documents the internal replacements for standard C library functions. (Interesting only for extension writers and Perl core hackers.) [561+]
- perldebtut is a Perl debugging tutorial. [561+]
- perlebcdic contains considerations for running Perl on EBCDIC platforms. [561+]
- perlintro is a gentle introduction to Perl.
- perliol documents the internals of PerlIO with layers.
- perlmodstyle is a style guide for writing modules.
- perlnewmod tells about writing and submitting a new module. [561+]
- perlpacktut is a **pack()** tutorial.
- perlpod has been rewritten to be clearer and to record the best practices gathered over the years.

- perlpodspec is a more formal specification of the pod format, mainly of interest for writers of pod applications, not to people writing in pod.
- perlretut is a regular expression tutorial. [561+]
- perlrequick is a regular expressions quick-start guide. Yes, much quicker than perlretut. [561]
- perltodo has been updated.
- perltootc has been renamed as perltooc (to not to conflict with perltoot in filesystems restricted to ''8.3'' names).
- perluniintro is an introduction to using Unicode in Perl. (perlunicode is more of a detailed reference and background information)
- perlutil explains the command line utilities packaged with the Perl distribution. [561+]

The following platform-specific documents are available before the installation as README.*platform*, and after the installation as perl*platform*:

```
perlaix perlamiga perlapollo perlbeos perlbs2000
perlce perlcygwin perldgux perldos perlepoc perlfreebsd perlhpux
perlhurd perlirix perlmachten perlmacos perlmint perlmpeix
perlnetware perlos2 perlos390 perlplan9 perlqnx perlsolaris
perltru64 perluts perlvmesa perlvms perlvos perlwin32
```

These documents usually detail one or more of the following subjects: configuring, building, testing, installing, and sometimes also using Perl on the said platform.

Eastern Asian Perl users are now welcomed in their own languages: README.jp (Japanese), README.ko (Korean), README.cn (simplified Chinese) and README.tw (traditional Chinese), which are written in normal pod but encoded in EUC-JP, EUC-KR, EUC-CN and Big5. These will get installed as

```
perljp perlko perlcn perltw
```

- The documentation for the POSIX-BC platform is called ''BS2000'', to avoid confusion with the Perl POSIX module.
- The documentation for the WinCE platform is called perlce (README.ce in the source code kit), to avoid confusion with the perlwin32 documentation on 8.3−restricted filesystems.

## Performance Enhancements

- **map()** could get pathologically slow when the result list it generates is larger than the source list. The performance has been improved for common scenarios. [561]
- **sort()** is also fully reentrant, in the sense that the sort function can itself call **sort()**. This did not work reliably in previous releases. [561]
- **sort()** has been changed to use primarily mergesort internally as opposed to the earlier quicksort. For very small lists this may result in slightly slower sorting times, but in general the speedup should be at least 20%. Additional bonuses are that the worst case behaviour of **sort()** is now better (in computer science terms it now runs in time O(N log N), as opposed to quicksort's Theta(N**2) worst-case run time behaviour), and that **sort()** is now stable (meaning that elements with identical keys will stay ordered as they were before the sort). See the `sort` pragma for information.

  The story in more detail: suppose you want to serve yourself a little slice of Pi.

  ```
  @digits = ( 3,1,4,1,5,9 );
  ```

  A numerical sort of the digits will yield (1,1,3,4,5,9), as expected. Which `1` comes first is hard to know, since one `1` looks pretty much like any other. You can regard this as totally trivial, or somewhat profound. However, if you just want to sort the even digits ahead of the odd ones, then what will

  ```
  sort { ($a % 2) <=> ($b % 2) } @digits;
  ```

  yield? The only even digit, `4`, will come first. But how about the odd numbers, which all compare equal? With the quicksort algorithm used to implement Perl 5.6 and earlier, the order of ties is left up to the sort. So, as you add more and more digits of Pi, the order in which the sorted even and

odd digits appear will change. and, for sufficiently large slices of Pi, the quicksort algorithm in Perl 5.8 won't return the same results even if reinvoked with the same input. The justification for this rests with quicksort's worst case behavior. If you run

```
sort { $a <=> $b } ( 1 .. $N , 1 .. $N );
```

(something you might approximate if you wanted to merge two sorted arrays using sort), doubling `$N` doesn't just double the quicksort time, it *quadruples* it. Quicksort has a worst case run time that can grow like N**2, so-called *quadratic* behaviour, and it can happen on patterns that may well arise in normal use. You won't notice this for small arrays, but you *will* notice it with larger arrays, and you may not live long enough for the sort to complete on arrays of a million elements. So the 5.8 quicksort scrambles large arrays before sorting them, as a statistical defence against quadratic behaviour. But that means if you sort the same large array twice, ties may be broken in different ways.

Because of the unpredictability of tie-breaking order, and the quadratic worst-case behaviour, quicksort was *almost* replaced completely with a stable mergesort. *Stable* means that ties are broken to preserve the original order of appearance in the input array. So

```
sort { ($a % 2) <=> ($b % 2) } (3,1,4,1,5,9);
```

will yield (4,3,1,1,5,9), guaranteed. The even and odd numbers appear in the output in the same order they appeared in the input. Mergesort has worst case O(N log N) behaviour, the best value attainable. And, ironically, this mergesort does particularly well where quicksort goes quadratic: mergesort sorts (1..$N, 1..$N) in O(N) time. But quicksort was rescued at the last moment because it is faster than mergesort on certain inputs and platforms. For example, if you really *don't* care about the order of even and odd digits, quicksort will run in O(N) time; it's very good at sorting many repetitions of a small number of distinct elements. The quicksort divide and conquer strategy works well on platforms with relatively small, very fast, caches. Eventually, the problem gets whittled down to one that fits in the cache, from which point it benefits from the increased memory speed.

Quicksort was rescued by implementing a sort pragma to control aspects of the sort. The **stable** subpragma forces stable behaviour, regardless of algorithm. The **_quicksort** and **_mergesort** subpragmas are heavy-handed ways to select the underlying implementation. The leading _ is a reminder that these subpragmas may not survive beyond 5.8. More appropriate mechanisms for selecting the implementation exist, but they wouldn't have arrived in time to save quicksort.

- Hashes now use Bob Jenkins ''One-at-a-Time'' hashing key algorithm ( http://burtleburtle.net/bob/hash/doobs.html ). This algorithm is reasonably fast while producing a much better spread of values than the old hashing algorithm (originally by Chris Torek, later tweaked by Ilya Zakharevich). Hash values output from the algorithm on a hash of all 3−char printable ASCII keys comes much closer to passing the DIEHARD random number generation tests. According to perlbench, this change has not affected the overall speed of Perl.
- **unshift()** should now be noticeably faster.

## Installation and Configuration Improvements

### Generic Improvements

- INSTALL now explains how you can configure Perl to use 64−bit integers even on non−64−bit platforms.
- Policy.sh policy change: if you are reusing a Policy.sh file (see INSTALL) and you use Configure −Dprefix=/foo/bar and in the old Policy `$prefix` eq `$siteprefix` and `$prefix` eq `$vendorprefix`, all of them will now be changed to the new prefix, /foo/bar. (Previously only `$prefix` changed.) If you do not like this new behaviour, specify prefix, siteprefix, and vendorprefix explicitly.
- A new optional location for Perl libraries, otherlibdirs, is available. It can be used for example for vendor add-ons without disturbing Perl's own library directories.
- In many platforms, the vendor-supplied 'cc' is too stripped-down to build Perl (basically, 'cc' doesn't do ANSI C). If this seems to be the case and 'cc' does not seem to be the GNU C compiler 'gcc', an automatic attempt is made to find and use 'gcc' instead.

- gcc needs to closely track the operating system release to avoid build problems. If Configure finds that gcc was built for a different operating system release than is running, it now gives a clearly visible warning that there may be trouble ahead.
- Since Perl 5.8 is not binary-compatible with previous releases of Perl, Configure no longer suggests including the 5.005 modules in `@INC`.
- Configure `-S` can now run non-interactively. [561]
- Configure support for pdp11-style memory models has been removed due to obsolescence. [561]
- configure.gnu now works with options with whitespace in them.
- installperl now outputs everything to STDERR.
- Because PerlIO is now the default on most platforms, ''-perlio'' doesn't get appended to the `$Config`{archname} (also known as $^O) anymore. Instead, if you explicitly choose not to use perlio (Configure command line option -Uuseperlio), you will get ''-stdio'' appended.
- Another change related to the architecture name is that ''-64all'' (-Duse64bitall, or ''maximally 64-bit'') is appended only if your pointers are 64 bits wide. (To be exact, the use64bitall is ignored.)
- In AFS installations, one can configure the root of the AFS to be somewhere else than the default */afs* by using the Configure parameter `-Dafsroot=/some/where/else`.
- APPLLIB_EXP, a lesser-known configuration-time definition, has been documented. It can be used to prepend site-specific directories to Perl's default search path (@INC); see INSTALL for information.
- The version of Berkeley DB used when the Perl (and, presumably, the DB_File extension) was built is now available as `@Config{qw(db_version_major db_version_minor db_version_patch)}` from Perl and as `DB_VERSION_MAJOR_CFG DB_VERSION_MINOR_CFG DB_VERSION_PATCH_CFG` from C.
- Building Berkeley DB3 for compatibility modes for DB, NDBM, and ODBM has been documented in INSTALL.
- If you have CPAN access (either network or a local copy such as a CD-ROM) you can during specify extra modules to Configure to build and install with Perl using the -Dextras=... option. See INSTALL for more details.
- In addition to config.over, a new override file, config.arch, is available. This file is supposed to be used by hints file writers for architecture-wide changes (as opposed to config.over which is for site-wide changes).
- If your file system supports symbolic links, you can build Perl outside of the source directory by

  ```
  mkdir perl/build/directory
  cd perl/build/directory
  sh /path/to/perl/source/Configure -Dmksymlinks ...
  ```

  This will create in perl/build/directory a tree of symbolic links pointing to files in /path/to/perl/source. The original files are left unaffected. After Configure has finished, you can just say

  ```
  make all test
  ```

  and Perl will be built and tested, all in perl/build/directory. [561]
- For Perl developers, several new make targets for profiling and debugging have been added; see perlhack.
  - Use of the *gprof* tool to profile Perl has been documented in perlhack. There is a make target called ''perl.gprof'' for generating a gprofiled Perl executable.
  - If you have GCC 3, there is a make target called ''perl.gcov'' for creating a gcoved Perl executable for coverage analysis. See perlhack.

- If you are on IRIX or Tru64 platforms, new profiling/debugging options have been added; see perlhack for more information about pixie and Third Degree.

- Guidelines of how to construct minimal Perl installations have been added to INSTALL.
- The Thread extension is now not built at all under ithreads (`Configure -Duseithreads`) because it wouldn't work anyway (the Thread extension requires being Configured with `-Duse5005threads`).

  **Note that the 5.005 threads are unsupported and deprecated: if you have code written for the old threads you should migrate it to the new ithreads model.**
- The Gconvert macro ($Config{d_Gconvert}) used by perl for stringifying floating-point numbers is now more picky about using sprintf %.*g rules for the conversion. Some platforms that used to use gcvt may now resort to the slower sprintf.
- The obsolete method of making a special (e.g., debugging) flavor of perl by saying

  ```
  make LIBPERL=libperld.a
  ```

  has been removed. Use -DDEBUGGING instead.

### New Or Improved Platforms

For the list of platforms known to support Perl, see ''Supported Platforms'' in perlport.

- AIX dynamic loading should be now better supported.
- AIX should now work better with gcc, threads, and 64-bitness. Also the long doubles support in AIX should be better now. See perlaix.
- AtheOS ( http://www.atheos.cx/ ) is a new platform.
- BeOS has been reclaimed.
- The DG/UX platform now supports 5.005-style threads. See perldgux.
- The DYNIX/ptx platform (also known as dynixptx) is supported at or near osvers 4.5.2.
- EBCDIC platforms (z/OS (also known as OS/390), POSIX-BC, and VM/ESA) have been regained. Many test suite tests still fail and the co-existence of Unicode and EBCDIC isn't quite settled, but the situation is much better than with Perl 5.6. See perlos390, perlbs2000 (for POSIX-BC), and perlvmesa for more information. (**Note:** support for VM/ESA was removed in Perl v5.18.0. The relevant information was in *README.vmesa*)
- Building perl with -Duseithreads or -Duse5005threads now works under HP-UX 10.20 (previously it only worked under 10.30 or later). You will need a thread library package installed. See README.hpux. [561]
- Mac OS Classic is now supported in the mainstream source package (MacPerl has of course been available since perl 5.004 but now the source code bases of standard Perl and MacPerl have been synchronised) [561]
- Mac OS X (or Darwin) should now be able to build Perl even on HFS+ filesystems. (The case-insensitivity used to confuse the Perl build process.)
- NCR MP-RAS is now supported. [561]
- All the NetBSD specific patches (except for the installation specific ones) have been merged back to the main distribution.
- NetWare from Novell is now supported. See perlnetware.
- NonStop-UX is now supported. [561]
- NEC SUPER-UX is now supported.
- All the OpenBSD specific patches (except for the installation specific ones) have been merged back to the main distribution.
- Perl has been tested with the GNU pth userlevel thread package ( http://www.gnu.org/software/pth/pth.html ). All thread tests of Perl now work, but not without adding some **yield()**s to the tests, so while pth (and other userlevel thread implementations) can be considered to be ''working'' with Perl ithreads, keep in mind the possible non-preemptability of

the underlying thread implementation.

- Stratus VOS is now supported using Perl's native build method (Configure). This is the recommended method to build Perl on VOS. The older methods, which build miniperl, are still available. See perlvos. [561+]
- The Amdahl UTS Unix mainframe platform is now supported. [561]
- WinCE is now supported. See perlce.
- z/OS (formerly known as OS/390, formerly known as MVS OE) now has support for dynamic loading. This is not selected by default, however, you must specify −Dusedl in the arguments of Configure. [561]

## Selected Bug Fixes

Numerous memory leaks and uninitialized memory accesses have been hunted down. Most importantly, anonymous subs used to leak quite a bit. [561]

- The autouse pragma didn't work for Multi::Part::Function::Names.
- **caller()** could cause core dumps in certain situations. Carp was sometimes affected by this problem. In particular, **caller()** now returns a subroutine name of `(unknown)` for subroutines that have been removed from the symbol table.
- chop(@list) in list context returned the characters chopped in reverse order. This has been reversed to be in the right order. [561]
- Configure no longer includes the DBM libraries (dbm, gdbm, db, ndbm) when building the Perl binary. The only exception to this is SunOS 4.x, which needs them. [561]
- The behaviour of non-decimal but numeric string constants such as ‘‘0x23’’ was platform-dependent: in some platforms that was seen as 35, in some as 0, in some as a floating point number (don't ask). This was caused by Perl's using the operating system libraries in a situation where the result of the string to number conversion is undefined: now Perl consistently handles such strings as zero in numeric contexts.
- Several debugger fixes: exit code now reflects the script exit code, condition `"0"` now treated correctly, the `d` command now checks line number, `$.` no longer gets corrupted, and all debugger output now goes correctly to the socket if RemotePort is set. [561]
- The debugger (perl5db.pl) has been modified to present a more consistent commands interface, via (CommandSet=580). perl5db.t was also added to test the changes, and as a placeholder for further tests.

  See perldebug.
- The debugger has a new `dumpDepth` option to control the maximum depth to which nested structures are dumped. The `x` command has been extended so that `x N EXPR` dumps out the value of *EXPR* to a depth of at most *N* levels.
- The debugger can now show lexical variables if you have the CPAN module PadWalker installed.
- The order of DESTROYs has been made more predictable.
- Perl 5.6.0 could emit spurious warnings about redefinition of **dl_error()** when statically building extensions into perl. This has been corrected. [561]
- dprofpp −R didn't work.
- `*foo{FORMAT}` now works.
- Infinity is now recognized as a number.
- UNIVERSAL::isa no longer caches methods incorrectly. (This broke the Tk extension with 5.6.0.) [561]
- Lexicals I: lexicals outside an eval "‘ weren't resolved correctly inside a subroutine definition inside the eval ’’‘ if they were not already referenced in the top level of the eval’’"ed code.
- Lexicals II: lexicals leaked at file scope into subroutines that were declared before the lexicals.

- Lexical warnings now propagating correctly between scopes and into `eval "..."`.
- `use warnings qw(FATAL all)` did not work as intended. This has been corrected. [561]
- **warnings::enabled()** now reports the state of $^W correctly if the caller isn't using lexical warnings. [561]
- Line renumbering with eval and `#line` now works. [561]
- Fixed numerous memory leaks, especially in eval "".
- Localised tied variables no longer leak memory

```
use Tie::Hash;
tie my %tied_hash => 'Tie::StdHash';

...

# Used to leak memory every time local() was called;
# in a loop, this added up.
local($tied_hash{Foo}) = 1;
```

- Localised hash elements (and `%ENV`) are correctly unlocalised to not exist, if they didn't before they were localised.

```
use Tie::Hash;
tie my %tied_hash => 'Tie::StdHash';

...

# Nothing has set the FOO element so far

{ local $tied_hash{FOO} = 'Bar' }

# This used to print, but not now.
print "exists!\n" if exists $tied_hash{FOO};
```

  As a side effect of this fix, tied hash interfaces **must** define the EXISTS and DELETE methods.
- **mkdir()** now ignores trailing slashes in the directory name, as mandated by POSIX.
- Some versions of glibc have a broken **modfl()**. This affects builds with `-Duselongdouble`. This version of Perl detects this brokenness and has a workaround for it. The glibc release 2.2.2 is known to have fixed the **modfl()** bug.
- Modulus of unsigned numbers now works (4063328477 % 65535 used to return 27406, instead of 27047). [561]
- Some ''not a number'' warnings introduced in 5.6.0 eliminated to be more compatible with 5.005. Infinity is now recognised as a number. [561]
- Numeric conversions did not recognize changes in the string value properly in certain circumstances. [561]
- Attributes (such as :shared) didn't work with **our()**.
- **our()** variables will not cause bogus ''Variable will not stay shared'' warnings. [561]
- ''our'' variables of the same name declared in two sibling blocks resulted in bogus warnings about ''redeclaration'' of the variables. The problem has been corrected. [561]
- pack ''Z'' now correctly terminates the string with ''\0''.
- Fix password routines which in some shadow password platforms (e.g. HP-UX) caused **getpwent()** to return every other entry.
- The PERL5OPT environment variable (for passing command line arguments to Perl) didn't work for more than a single group of options. [561]

- PERL5OPT with embedded spaces didn't work.
- **printf()** no longer resets the numeric locale to ''C''.
- `qw(a\\b)` now parses correctly as `'a\\b'`: that is, as three characters, not four. [561]
- **pos()** did not return the correct value within s///ge in earlier versions. This is now handled correctly. [561]
- Printing quads (64−bit integers) with printf/sprintf now works without the q L ll prefixes (assuming you are on a quad-capable platform).
- Regular expressions on references and overloaded scalars now work. [561+]
- Right-hand side magic (GMAGIC) could in many cases such as string concatenation be invoked too many times.
- **scalar()** now forces scalar context even when used in void context.
- SOCKS support is now much more robust.
- **sort()** arguments are now compiled in the right wantarray context (they were accidentally using the context of the **sort()** itself). The comparison block is now run in scalar context, and the arguments to be sorted are always provided list context. [561]
- Changed the POSIX character class `[[:space:]]` to include the (very rarely used) vertical tab character. Added a new POSIX-ish character class `[[:blank:]]` which stands for horizontal whitespace (currently, the space and the tab).
- The tainting behaviour of **sprintf()** has been rationalized. It does not taint the result of floating point formats anymore, making the behaviour consistent with that of string interpolation. [561]
- Some cases of inconsistent taint propagation (such as within hash values) have been fixed.
- The RE engine found in Perl 5.6.0 accidentally pessimised certain kinds of simple pattern matches. These are now handled better. [561]
- Regular expression debug output (whether through `use re 'debug'` or via `-Dr`) now looks better. [561]
- Multi-line matches like `"a\nxb\n" =~ /(?!\A)x/m` were flawed. The bug has been fixed. [561]
- Use of $& could trigger a core dump under some situations. This is now avoided. [561]
- The regular expression captured submatches ($1, `$2`, ...) are now more consistently unset if the match fails, instead of leaving false data lying around in them. [561]
- **readline()** on files opened in ''slurp'' mode could return an extra "" (blank line) at the end in certain situations. This has been corrected. [561]
- Autovivification of symbolic references of special variables described in perlvar (as in `${$num}`) was accidentally disabled. This works again now. [561]
- Sys::Syslog ignored the `LOG_AUTH` constant.
- `$AUTOLOAD`, **sort()**, **lock()**, and spawning subprocesses in multiple threads simultaneously are now thread-safe.
- Tie::Array's SPLICE method was broken.
- Allow a read-only string on the left-hand side of a non-modifying tr///.
- If `STDERR` is tied, warnings caused by `warn` and `die` now correctly pass to it.
- Several Unicode fixes.
    - BOMs (byte order marks) at the beginning of Perl files (scripts, modules) should now be transparently skipped. UTF−16 and UCS−2 encoded Perl files should now be read correctly.
    - The character tables have been updated to Unicode 3.2.0.

- Comparing with utf8 data does not magically upgrade non−utf8 data into utf8. (This was a problem for example if you were mixing data from I/O and Unicode data: your output might have got magically encoded as UTF−8.)
- Generating illegal Unicode code points such as U+FFFE, or the UTF−16 surrogates, now also generates an optional warning.
- `IsAlnum`, `IsAlpha`, and `IsWord` now match titlecase.
- Concatenation with the `.` operator or via variable interpolation, `eq`, `substr`, `reverse`, `quotemeta`, the `x` operator, substitution with `s///`, single-quoted UTF−8, should now work.
- The `tr///` operator now works. Note that the `tr///CU` functionality has been removed (but see pack('U0', ...)).
- `eval "v200"` now works.
- Perl 5.6.0 parsed m/\x{ab}/ incorrectly, leading to spurious warnings. This has been corrected. [561]
- Zero entries were missing from the Unicode classes such as `IsDigit`.

- Large unsigned numbers (those above 2**31) could sometimes lose their unsignedness, causing bogus results in arithmetic operations. [561]
- The Perl parser has been stress tested using both random input and Markov chain input and the few found crashes and lockups have been fixed.

### Platform Specific Changes and Fixes

- BSDI 4.*

  Perl now works on post−4.0 BSD/OSes.

- All BSDs

  Setting `$0` now works (as much as possible; see perlvar for details).

- Cygwin

  Numerous updates; currently synchronised with Cygwin 1.3.10.

- Previously DYNIX/ptx had problems in its Configure probe for non-blocking I/O.

- EPOC

  EPOC now better supported. See README.epoc. [561]

- FreeBSD 3.*

  Perl now works on post−3.0 FreeBSDs.

- HP-UX

  README.hpux updated; `Configure -Duse64bitall` now works; now uses HP-UX malloc instead of Perl malloc.

- IRIX

  Numerous compilation flag and hint enhancements; accidental mixing of 32−bit and 64−bit libraries (a doomed attempt) made much harder.

- Linux
  - Long doubles should now work (see INSTALL). [561]
  - Linux previously had problems related to sockaddrlen when using **accept()**, **recvfrom()** (in Perl: **recv()**), **getpeername()**, and **getsockname()**.

- Mac OS Classic

  Compilation of the standard Perl distribution in Mac OS Classic should now work if you have the Metrowerks development environment and the missing Mac-specific toolkit bits. Contact the macperl mailing list for details.

- MPE/iX

  MPE/iX update after Perl 5.6.0. See README.mpeix. [561]

- NetBSD/threads: try installing the GNU pth (should be in the packages collection, or http://www.gnu.org/software/pth/), and Configure with −Duseithreads.

- NetBSD/sparc

  Perl now works on NetBSD/sparc.

- OS/2

  Now works with usethreads (see INSTALL). [561]

- Solaris

  64−bitness using the Sun Workshop compiler now works.

- Stratus VOS

  The native build method requires at least VOS Release 14.5.0 and GNU C++/GNU Tools 2.0.1 or later. The Perl pack function now maps overflowed values to +infinity and underflowed values to −infinity.

- Tru64 (aka Digital UNIX, aka DEC OSF/1)

  The operating system version letter now recorded in `$Config{osvers}`. Allow compiling with gcc (previously explicitly forbidden). Compiling with gcc still not recommended because buggy code results, even with gcc 2.95.2.

- Unicos

  Fixed various alignment problems that lead into core dumps either during build or later; no longer dies on math errors at runtime; now using full quad integers (64 bits), previously was using only 46 bit integers for speed.

- VMS

  See ‘‘Socket Extension Dynamic in VMS’’ and ‘‘IEEE-format Floating Point Default on OpenVMS Alpha’’ for important changes not otherwise listed here.

  **chdir()** now works better despite a CRT bug; now works with MULTIPLICITY (see INSTALL); now works with Perl’s malloc.

  The tainting of `%ENV` elements via `keys` or `values` was previously unimplemented. It now works as documented.

  The `waitpid` emulation has been improved. The worst bug (now fixed) was that a pid of −1 would cause a wildcard search of all processes on the system.

  POSIX-style signals are now emulated much better on VMS versions prior to 7.0.

  The `system` function and backticks operator have improved functionality and better error handling. [561]

  File access tests now use current process privileges rather than the user’s default privileges, which could sometimes result in a mismatch between reported access and actual access. This improvement is only available on VMS v6.0 and later.

  There is a new `kill` implementation based on `sys$sigprc` that allows older VMS systems (pre−7.0) to use `kill` to send signals rather than simply force exit. This implementation also allows later systems to call `kill` from within a signal handler.

  Iterative logical name translations are now limited to 10 iterations in imitation of SHOW LOGICAL and other OpenVMS facilities.

- Windows

  - Signal handling now works better than it used to. It is now implemented using a Windows message loop, and is therefore less prone to random crashes.

- **fork()** emulation is now more robust, but still continues to have a few esoteric bugs and caveats. See perlfork for details. [561+]
- A failed (pseudo)fork now returns undef and sets errno to EAGAIN. [561]
- The following modules now work on Windows:

```
ExtUtils::Embed         [561]
IO::Pipe
IO::Poll
Net::Ping
```

- **IO::File::new_tmpfile()** is no longer limited to 32767 invocations per-process.
- Better **chdir()** return value for a non-existent directory.
- Compiling perl using the 64−bit Platform SDK tools is now supported.
- The **Win32::SetChildShowWindow()** builtin can be used to control the visibility of windows created by child processes. See Win32 for details.
- Non-blocking waits for child processes (or pseudo-processes) are supported via `waitpid($pid, &POSIX::WNOHANG)`.
- The behavior of **system()** with multiple arguments has been rationalized. Each unquoted argument will be automatically quoted to protect whitespace, and any existing whitespace in the arguments will be preserved. This improves the portability of system(@args) by avoiding the need for Windows `cmd` shell specific quoting in perl programs.

  Note that this means that some scripts that may have relied on earlier buggy behavior may no longer work correctly. For example, `system("nmake /nologo", @args)` will now attempt to run the file `nmake /nologo` and will fail when such a file isn't found. On the other hand, perl will now execute code such as `system("c:/Program Files/MyApp/foo.exe", @args)` correctly.
- The perl header files no longer suppress common warnings from the Microsoft Visual C++ compiler. This means that additional warnings may now show up when compiling XS code.
- Borland C++ v5.5 is now a supported compiler that can build Perl. However, the generated binaries continue to be incompatible with those generated by the other supported compilers (GCC and Visual C++). [561]
- Duping socket handles with open(F, ''>&MYSOCK'') now works under Windows 9x. [561]
- Current directory entries in `%ENV` are now correctly propagated to child processes. [561]
- New `%ENV` entries now propagate to subprocesses. [561]
- **Win32::GetCwd()** correctly returns C:\ instead of C: when at the drive root. Other bugs in **chdir()** and **Cwd::cwd()** have also been fixed. [561]
- The makefiles now default to the features enabled in ActiveState ActivePerl (a popular Win32 binary distribution). [561]
- HTML files will now be installed in c:\perl\html instead of c:\perl\lib\pod\html
- REG_EXPAND_SZ keys are now allowed in registry settings used by perl. [561]
- Can now **send()** from all threads, not just the first one. [561]
- ExtUtils::MakeMaker now uses `$ENV{LIB}` to search for libraries. [561]
- Less stack reserved per thread so that more threads can run concurrently. (Still 16M per thread.) [561]
- `File::Spec->tmpdir()` now prefers C:/temp over /tmp (works better when perl is running as service).

- Better UNC path handling under ithreads. [561]
- **wait()**, **waitpid()**, and backticks now return the correct exit status under Windows 9x. [561]
- A socket handle leak in **accept()** has been fixed. [561]

## New or Changed Diagnostics

Please see perldiag for more details.

- Ambiguous range in the transliteration operator (like a−z−9) now gives a warning.
- chdir("") and chdir(undef) now give a deprecation warning because they cause a possible unintentional chdir to the home directory. Say **chdir()** if you really mean that.
- Two new debugging options have been added: if you have compiled your Perl with debugging, you can use the −DT [561] and −DR options to trace tokenising and to add reference counts to displaying variables, respectively.
- The lexical warnings category ‘‘deprecated’’ is no longer a sub-category of the ‘‘syntax’’ category. It is now a top-level category in its own right.
- Unadorned **dump()** will now give a warning suggesting to use explicit **CORE::dump()** if that’s what really is meant.
- The ‘‘Unrecognized escape’’ warning has been extended to include `\8`, `\9`, and `\_`. There is no need to escape any of the `\w` characters.
- All regular expression compilation error messages are now hopefully easier to understand both because the error message now comes before the failed regex and because the point of failure is now clearly marked by a `<-- HERE` marker.
- Various I/O (and socket) functions like **binmode()**, **close()**, and so forth now more consistently warn if they are used illogically either on a yet unopened or on an already closed filehandle (or socket).
- Using **lstat()** on a filehandle now gives a warning. (It’s a non-sensical thing to do.)
- The `-M` and `-m` options now warn if you didn’t supply the module name.
- If you in `use` specify a required minimum version, modules matching the name and but not defining a `$VERSION` will cause a fatal failure.
- Using negative offset for **vec()** in lvalue context is now a warnable offense.
- Odd number of arguments to overload::constant now elicits a warning.
- Odd number of elements in anonymous hash now elicits a warning.
- The various ‘‘opened only for’’, ‘‘on closed’’, ‘‘never opened’’ warnings drop the `main::` prefix for filehandles in the `main` package, for example `STDIN` instead of `main::STDIN`.
- Subroutine prototypes are now checked more carefully, you may get warnings for example if you have used non-prototype characters.
- If an attempt to use a (non-blessed) reference as an array index is made, a warning is given.
- `push @a;` and `unshift @a;` (with no values to push or unshift) now give a warning. This may be a problem for generated and eval’ed code.
- If you try to ‘‘pack’’ in perlfunc a number less than 0 or larger than 255 using the `"C"` format you will get an optional warning. Similarly for the `"c"` format and a number less than −128 or more than 127.
- pack `P` format now demands an explicit size.
- unpack `w` now warns of unterminated compressed integers.
- Warnings relating to the use of PerlIO have been added.
- Certain regex modifiers such as `(?o)` make sense only if applied to the entire regex. You will get an optional warning if you try to do otherwise.

- Variable length lookbehind has not yet been implemented, trying to use it will tell that.
- Using arrays or hashes as references (e.g. `%foo->{bar}` has been deprecated for a while. Now you will get an optional warning.
- Warnings relating to the use of the new restricted hashes feature have been added.
- Self-ties of arrays and hashes are not supported and fatal errors will happen even at an attempt to do so.
- Using `sort` in scalar context now issues an optional warning. This didn't do anything useful, as the sort was not performed.
- Using the /g modifier in **split()** is meaningless and will cause a warning.
- Using **splice()** past the end of an array now causes a warning.
- Malformed Unicode encodings (UTF-8 and UTF-16) cause a lot of warnings, as does trying to use UTF-16 surrogates (which are unimplemented).
- Trying to use Unicode characters on an I/O stream without marking the stream's encoding (using **open()** or **binmode()**) will cause ''Wide character'' warnings.
- Use of v-strings in use/require causes a (backward) portability warning.
- Warnings relating to the use interpreter threads and their shared data have been added.

## Changed Internals

- PerlIO is now the default.
- perlapi.pod (a companion to perlguts) now attempts to document the internal API.
- You can now build a really minimal perl called microperl. Building microperl does not require even running Configure; `make -f Makefile.micro` should be enough. Beware: microperl makes many assumptions, some of which may be too bold; the resulting executable may crash or otherwise misbehave in wondrous ways. For careful hackers only.
- Added **rsignal()**, **whichsig()**, **do_join()**, op_clear, op_null, **ptr_table_clear()**, **ptr_table_free()**, **sv_setref_uv()**, and several UTF-8 interfaces to the publicised API. For the full list of the available APIs see perlapi.
- Made possible to propagate customised exceptions via **croak()**ing.
- Now xsubs can have attributes just like subs. (Well, at least the built-in attributes.)
- dTHR and djSP have been obsoleted; the former removed (because it's a no-op) and the latter replaced with dSP.
- PERL_OBJECT has been completely removed.
- The MAGIC constants (e.g. `'P'`) have been macrofied (e.g. `PERL_MAGIC_TIED`) for better source code readability and maintainability.
- The regex compiler now maintains a structure that identifies nodes in the compiled bytecode with the corresponding syntactic features of the original regex expression. The information is attached to the new `offsets` member of the `struct regexp`. See perldebguts for more complete information.
- The C code has been made much more `gcc -Wall` clean. Some warning messages still remain in some platforms, so if you are compiling with gcc you may see some warnings about dubious practices. The warnings are being worked on.
- *perly.c*, *sv.c*, and *sv.h* have now been extensively commented.
- Documentation on how to use the Perl source repository has been added to *Porting/repository.pod*.
- There are now several profiling make targets.

## Security Vulnerability Closed [561]

(This change was already made in 5.7.0 but bears repeating here.) (5.7.0 came out before 5.6.1: the development branch 5.7 released earlier than the maintenance branch 5.6)

A potential security vulnerability in the optional suidperl component of Perl was identified in August 2000. suidperl is neither built nor installed by default. As of November 2001 the only known

vulnerable platform is Linux, most likely all Linux distributions. CERT and various vendors and distributors have been alerted about the vulnerability. See http://www.cpan.org/src/5.0/sperl−2000−08−05/sperl−2000−08−05.txt for more information.

The problem was caused by Perl trying to report a suspected security exploit attempt using an external program, /bin/mail. On Linux platforms the /bin/mail program had an undocumented feature which when combined with suidperl gave access to a root shell, resulting in a serious compromise instead of reporting the exploit attempt. If you don't have /bin/mail, or if you have 'safe setuid scripts', or if suidperl is not installed, you are safe.

The exploit attempt reporting feature has been completely removed from Perl 5.8.0 (and the maintenance release 5.6.1, and it was removed also from all the Perl 5.7 releases), so that particular vulnerability isn't there anymore. However, further security vulnerabilities are, unfortunately, always possible. The suidperl functionality is most probably going to be removed in Perl 5.10. In any case, suidperl should only be used by security experts who know exactly what they are doing and why they are using suidperl instead of some other solution such as sudo ( see http://www.courtesan.com/sudo/ ).

## New Tests

Several new tests have been added, especially for the *lib* and *ext* subsections. There are now about 69 000 individual tests (spread over about 700 test scripts), in the regression suite (5.6.1 has about 11 700 tests, in 258 test scripts) The exact numbers depend on the platform and Perl configuration used. Many of the new tests are of course introduced by the new modules, but still in general Perl is now more thoroughly tested.

Because of the large number of tests, running the regression suite will take considerably longer time than it used to: expect the suite to take up to 4−5 times longer to run than in perl 5.6. On a really fast machine you can hope to finish the suite in about 6−8 minutes (wallclock time).

The tests are now reported in a different order than in earlier Perls. (This happens because the test scripts from under t/lib have been moved to be closer to the library/extension they are testing.)

## Known Problems

### The Compiler Suite Is Still Very Experimental

The compiler suite is slowly getting better but it continues to be highly experimental. Use in production environments is discouraged.

### Localising Tied Arrays and Hashes Is Broken

```
local %tied_array;
```

doesn't work as one would expect: the old value is restored incorrectly. This will be changed in a future release, but we don't know yet what the new semantics will exactly be. In any case, the change will break existing code that relies on the current (ill-defined) semantics, so just avoid doing this in general.

### Building Extensions Can Fail Because Of Largefiles

Some extensions like mod_perl are known to have issues with 'largefiles', a change brought by Perl 5.6.0 in which file offsets default to 64 bits wide, where supported. Modules may fail to compile at all, or they may compile and work incorrectly. Currently, there is no good solution for the problem, but Configure now provides appropriate non-largefile ccflags, ldflags, libswanted, and libs in the `%Config` hash (e.g., `$Config{ccflags_nolargefiles}`) so the extensions that are having problems can try configuring themselves without the largefileness. This is admittedly not a clean solution, and the solution may not even work at all. One potential failure is whether one can (or, if one can, whether it's a good idea to) link together at all binaries with different ideas about file offsets; all this is platform-dependent.

### Modifying $_ Inside for(..)

```
for (1..5) { $_++ }
```

works without complaint. It shouldn't. (You should be able to modify only lvalue elements inside the loops.) You can see the correct behaviour by replacing the 1..5 with 1, 2, 3, 4, 5.

### mod_perl 1.26 Doesn't Build With Threaded Perl

Use mod_perl 1.27 or higher.

### lib/ftmp−security tests warn 'system possibly insecure'

Don't panic. Read the 'make test' section of INSTALL instead.

### libwww-perl (LWP) fails base/date #51

Use libwww-perl 5.65 or later.

### PDL failing some tests

Use PDL 2.3.4 or later.

### Perl_get_sv

You may get errors like 'Undefined symbol ‘‘Perl_get_sv’’' or ‘‘can't resolve symbol 'Perl_get_sv’’', or the symbol may be ‘‘Perl_sv_2pv’’. This probably means that you are trying to use an older shared Perl library (or extensions linked with such) with Perl 5.8.0 executable. Perl used to have such a subroutine, but that is no more the case. Check your shared library path, and any shared Perl libraries in those directories.

Sometimes this problem may also indicate a partial Perl 5.8.0 installation, see ‘‘Mac OS X dyld undefined symbols’’ for an example and how to deal with it.

### Self-tying Problems

Self-tying of arrays and hashes is broken in rather deep and hard-to-fix ways. As a stop-gap measure to avoid people from getting frustrated at the mysterious results (core dumps, most often), it is forbidden for now (you will get a fatal error even from an attempt).

A change to self-tying of globs has caused them to be recursively referenced (see: ‘‘Two-Phased Garbage Collection’’ in perlobj). You will now need an explicit untie to destroy a self-tied glob. This behaviour may be fixed at a later date.

Self-tying of scalars and IO thingies works.

### ext/threads/t/libc

If this test fails, it indicates that your libc (C library) is not threadsafe. This particular test stress tests the **localtime()** call to find out whether it is threadsafe. See perlthrtut for more information.

### Failure of Thread (5.005−style) tests

**Note that support for 5.005−style threading is deprecated, experimental and practically unsupported. In 5.10, it is expected to be removed. You should migrate your code to ithreads.**

The following tests are known to fail due to fundamental problems in the 5.005 threading implementation. These are not new failures — Perl 5.005_0x has the same bugs, but didn't have these tests.

```
../ext/B/t/xref.t                   255 65280    14   12  85.71%  3−14
../ext/List/Util/t/first.t          255 65280     7    4  57.14%  2 5−7
../lib/English.t                      2   512    54    2   3.70%  2−3
../lib/FileCache.t                              5    1  20.00%  5
../lib/Filter/Simple/t/data.t                   6    3  50.00%  1−3
../lib/Filter/Simple/t/filter_only.             9    3  33.33%  1−2 5
../lib/Math/BigInt/t/bare_mbf.t              1627    4   0.25%  8 11 1626−162
../lib/Math/BigInt/t/bigfltpm.t              1629    4   0.25%  10 13 1628−
                                                                 1629
../lib/Math/BigInt/t/sub_mbf.t               1633    4   0.24%  8 11 1632−163
../lib/Math/BigInt/t/with_sub.t              1628    4   0.25%  9 12 1627−162
../lib/Tie/File/t/31_autodefer.t    255 65280    65   32  49.23%  34−65
../lib/autouse.t                                10    1  10.00%  4
op/flip.t                                       15    1   6.67%  15
```

These failures are unlikely to get fixed as 5.005−style threads are considered fundamentally broken. (Basically what happens is that competing threads can corrupt shared global state, one good example being regular expression engine's state.)

### Timing problems

The following tests may fail intermittently because of timing problems, for example if the system is heavily loaded.

```
t/op/alarm.t
ext/Time/HiRes/HiRes.t
lib/Benchmark.t
lib/Memoize/t/expmod_t.t
lib/Memoize/t/speed.t
```

In case of failure please try running them manually, for example

```
./perl -Ilib ext/Time/HiRes/HiRes.t
```

### Tied/Magical Array/Hash Elements Do Not Autovivify

For normal arrays `$foo = \$bar[1]` will assign `undef` to `$bar[1]` (assuming that it didn't exist before), but for tied/magical arrays and hashes such autovivification does not happen because there is currently no way to catch the reference creation. The same problem affects slicing over non-existent indices/keys of a tied/magical array/hash.

### Unicode in package/class and subroutine names does not work

One can have Unicode in identifier names, but not in package/class or subroutine names. While some limited functionality towards this does exist as of Perl 5.8.0, that is more accidental than designed; use of Unicode for the said purposes is unsupported.

One reason of this unfinishedness is its (currently) inherent unportability: since both package names and subroutine names may need to be mapped to file and directory names, the Unicode capability of the filesystem becomes important — and there unfortunately aren't portable answers.

## Platform Specific Problems

### AIX

- If using the AIX native make command, instead of just ''make'' issue ''make all''. In some setups the former has been known to spuriously also try to run ''make install''. Alternatively, you may want to use GNU make.
- In AIX 4.2, Perl extensions that use C++ functions that use statics may have problems in that the statics are not getting initialized. In newer AIX releases, this has been solved by linking Perl with the libC_r library, but unfortunately in AIX 4.2 the said library has an obscure bug where the various functions related to time (such as **time()** and **gettimeofday()**) return broken values, and therefore in AIX 4.2 Perl is not linked against libC_r.
- vac 5.0.0.0 May Produce Buggy Code For Perl

  The AIX C compiler vac version 5.0.0.0 may produce buggy code, resulting in a few random tests failing when run as part of ''make test'', but when the failing tests are run by hand, they succeed. We suggest upgrading to at least vac version 5.0.1.0, that has been known to compile Perl correctly. ''lslpp -L|grep vac.C'' will tell you the vac version. See README.aix.
- If building threaded Perl, you may get compilation warning from pp_sys.c:

  ```
  "pp_sys.c", line 4651.39: 1506-280 (W) Function argument assignment between
  ```

  This is harmless; it is caused by the **getnetbyaddr()** and **getnetbyaddr_r()** having slightly different types for their first argument.

### Alpha systems with old gccs fail several tests

If you see op/pack, op/pat, op/regexp, or ext/Storable tests failing in a Linux/alpha or *BSD/Alpha, it's probably time to upgrade your gcc. gccs prior to 2.95.3 are definitely not good enough, and gcc 3.1 may be even better. (RedHat Linux/alpha with gcc 3.1 reported no problems, as did Linux 2.4.18 with gcc 2.95.4.) (In Tru64, it is preferable to use the bundled C compiler.)

### AmigaOS

Perl 5.8.0 doesn't build in AmigaOS. It broke at some point during the ithreads work and we could not find Amiga experts to unbreak the problems. Perl 5.6.1 still works for AmigaOS (as does the 5.7.2 development release).

### BeOS

The following tests fail on 5.8.0 Perl in BeOS Personal 5.03:

```
t/op/lfs.............................FAILED at test 17
t/op/magic...........................FAILED at test 24
ext/Fcntl/t/syslfs...................FAILED at test 17
ext/File/Glob/t/basic................FAILED at test 3
ext/POSIX/t/sigaction................FAILED at test 13
ext/POSIX/t/waitpid..................FAILED at test 1
```

(**Note:** more information was available in *README.beos* until support for BeOS was removed in Perl v5.18.0)

### Cygwin "unable to remap"

For example when building the Tk extension for Cygwin, you may get an error message saying ''unable to remap''. This is known problem with Cygwin, and a workaround is detailed in here: http://sources.redhat.com/ml/cygwin/2001-12/msg00894.html

### Cygwin ndbm tests fail on FAT

One can build but not install (or test the build of) the NDBM_File on FAT filesystems. Installation (or build) on NTFS works fine. If one attempts the test on a FAT install (or build) the following failures are expected:

```
../ext/NDBM_File/ndbm.t       13  3328    71   59  83.10%  1-2 4 16-71
../ext/ODBM_File/odbm.t      255 65280    ??   ??       %  ??
../lib/AnyDBM_File.t           2   512    12    2  16.67%  1 4
../lib/Memoize/t/errors.t      0   139    11    5  45.45%  7-11
../lib/Memoize/t/tie_ndbm.t   13  3328     4    4 100.00%  1-4
run/fresh_perl.t                        97    1   1.03%  91
```

NDBM_File fails and ODBM_File just coredumps.

If you intend to run only on FAT (or if using AnyDBM_File on FAT), run Configure with the -Ui_ndbm and -Ui_dbm options to prevent NDBM_File and ODBM_File being built.

### DJGPP Failures

```
t/op/stat............................FAILED at test 29
lib/File/Find/t/find.................FAILED at test 1
lib/File/Find/t/taint................FAILED at test 1
lib/h2xs.............................FAILED at test 15
lib/Pod/t/eol........................FAILED at test 1
lib/Test/Harness/t/strap-analyze.....FAILED at test 8
lib/Test/Harness/t/test-harness......FAILED at test 23
lib/Test/Simple/t/exit...............FAILED at test 1
```

The above failures are known as of 5.8.0 with native builds with long filenames, but there are a few more if running under dosemu because of limitations (and maybe bugs) of dosemu:

```
t/comp/cpp...........................FAILED at test 3
t/op/inccode.........................(crash)
```

and a few lib/ExtUtils tests, and several hundred Encode/t/Aliases.t failures that work fine with long filenames. So you really might prefer native builds and long filenames.

### FreeBSD built with ithreads coredumps reading large directories

This is a known bug in FreeBSD 4.5's **readdir_r()**, it has been fixed in FreeBSD 4.6 (see perlfreebsd (README.freebsd)).

### FreeBSD Failing locale Test 117 For ISO 8859-15 Locales

The ISO 8859-15 locales may fail the locale test 117 in FreeBSD. This is caused by the characters \xFF (y with diaeresis) and \xBE (Y with diaeresis) not behaving correctly when being matched case-insensitively. Apparently this problem has been fixed in the latest FreeBSD releases. ( http://www.freebsd.org/cgi/query-pr.cgi?pr=34308 )

### IRIX fails ext/List/Util/t/shuffle.t or Digest::MD5

IRIX with MIPSpro 7.3.1.2m or 7.3.1.3m compiler may fail the List::Util test ext/List/Util/t/shuffle.t by dumping core. This seems to be a compiler error since if compiled with gcc no core dump ensues, and no failures have been seen on the said test on any other platform.

Similarly, building the Digest::MD5 extension has been known to fail with ''*** Termination code 139

(bu21)''.

The cure is to drop optimization level (Configure −Doptimize=−O2).

### HP-UX lib/posix Subtest 9 Fails When LP64−Configured

If perl is configured with −Duse64bitall, the successful result of the subtest 10 of lib/posix may arrive before the successful result of the subtest 9, which confuses the test harness so much that it thinks the subtest 9 failed.

### Linux with glibc 2.2.5 fails t/op/int subtest #6 with −Duse64bitint

This is a known bug in the glibc 2.2.5 with long long integers. ( http://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=65612 )

### Linux With Sfio Fails op/misc Test 48

No known fix.

### Mac OS X

Please remember to set your environment variable LC_ALL to ''C'' (setenv LC_ALL C) before running ''make test'' to avoid a lot of warnings about the broken locales of Mac OS X.

The following tests are known to fail in Mac OS X 10.1.5 because of buggy (old) implementations of Berkeley DB included in Mac OS X:

```
 Failed Test                Stat Wstat Total Fail  Failed  List of Failed
 -------------------------------------------------------------------------
 ../ext/DB_File/t/db-btree.t    0    11    ??   ??       %  ??
 ../ext/DB_File/t/db-recno.t              149    3   2.01%  61 63 65
```

If you are building on a UFS partition, you will also probably see t/op/stat.t subtest #9 fail. This is caused by Darwin's UFS not supporting inode change time.

Also the ext/POSIX/t/posix.t subtest #10 fails but it is skipped for now because the failure is Apple's fault, not Perl's (blocked signals are lost).

If you Configure with ithreads, ext/threads/t/libc.t will fail. Again, this is not Perl's fault — the libc of Mac OS X is not threadsafe (in this particular test, the **localtime()** call is found to be threadunsafe.)

### Mac OS X dyld undefined symbols

If after installing Perl 5.8.0 you are getting warnings about missing symbols, for example

```
dyld: perl Undefined symbols
_perl_sv_2pv
_perl_get_sv
```

you probably have an old pre−Perl−5.8.0 installation (or parts of one) in /Library/Perl (the undefined symbols used to exist in pre−5.8.0 Perls). It seems that for some reason ''make install'' doesn't always completely overwrite the files in /Library/Perl. You can move the old Perl shared library out of the way like this:

```
cd /Library/Perl/darwin/CORE
mv libperl.dylib libperlold.dylib
```

and then reissue ''make install''. Note that the above of course is extremely disruptive for anything using the /usr/local/bin/perl. If that doesn't help, you may have to try removing all the .bundle files from beneath /Library/Perl, and again ''make install''−ing.

### OS/2 Test Failures

The following tests are known to fail on OS/2 (for clarity only the failures are shown, not the full error messages):

```
 ../lib/ExtUtils/t/Mkbootstrap.t    1   256    18    1   5.56%  8
 ../lib/ExtUtils/t/Packlist.t       1   256    34    1   2.94%  17
 ../lib/ExtUtils/t/basic.t          1   256    17    1   5.88%  14
 lib/os2_process.t                  2   512   227    2   0.88%  174 209
 lib/os2_process_kid.t                    227    2   0.88%  174 209
 lib/rx_cmprt.t                   255 65280    18    3  16.67%  16-18
```

### op/sprintf tests 91, 129, and 130

The op/sprintf tests 91, 129, and 130 are known to fail on some platforms. Examples include any platform using sfio, and Compaq/Tandem's NonStop-UX.

Test 91 is known to fail on QNX6 (nto), because `sprintf '%e',0` incorrectly produces `0.000000e+0` instead of `0.000000e+00`.

For tests 129 and 130, the failing platforms do not comply with the ANSI C Standard: lines 19ff on page 134 of ANSI X3.159 1989, to be exact. (They produce something other than "1" and "−1" when formatting 0.6 and −0.6 using the printf format "%.0f"; most often, they produce "0" and "−0".)

### SCO

The socketpair tests are known to be unhappy in SCO 3.2v5.0.4:

```
ext/Socket/socketpair.t..............FAILED tests 15-45
```

### Solaris 2.5

In case you are still using Solaris 2.5 (aka SunOS 5.5), you may experience failures (the test core dumping) in lib/locale.t. The suggested cure is to upgrade your Solaris.

### Solaris x86 Fails Tests With −Duse64bitint

The following tests are known to fail in Solaris x86 with Perl configured to use 64 bit integers:

```
ext/Data/Dumper/t/dumper.............FAILED at test 268
ext/Devel/Peek/Peek..................FAILED at test 7
```

### SUPER-UX (NEC SX)

The following tests are known to fail on SUPER-UX:

```
op/64bitint..........................FAILED tests 29-30, 32-33, 35-36
op/arith.............................FAILED tests 128-130
op/pack..............................FAILED tests 25-5625
op/pow...............................
op/taint.............................# msgsnd failed
../ext/IO/lib/IO/t/io_poll...........FAILED tests 3-4
../ext/IPC/SysV/ipcsysv..............FAILED tests 2, 5-6
../ext/IPC/SysV/t/msg................FAILED tests 2, 4-6
../ext/Socket/socketpair.............FAILED tests 12
../lib/IPC/SysV......................FAILED tests 2, 5-6
../lib/warnings......................FAILED tests 115-116, 118-119
```

The op/pack failure ("Cannot compress negative numbers at op/pack.t line 126") is serious but as of yet unsolved. It points at some problems with the signedness handling of the C compiler, as do the 64bitint, arith, and pow failures. Most of the rest point at problems with SysV IPC.

### Term::ReadKey not working on Win32

Use Term::ReadKey 2.20 or later.

### UNICOS/mk

- During Configure, the test

  ```
  Guessing which symbols your C compiler and preprocessor define...
  ```

  will probably fail with error messages like

  ```
  CC-20 cc: ERROR File = try.c, Line = 3
    The identifier "bad" is undefined.

    bad switch yylook 79bad switch yylook 79bad switch yylook 79bad switch y
    ^


  CC-65 cc: ERROR File = try.c, Line = 3
    A semicolon is expected at this point.
  ```

  This is caused by a bug in the awk utility of UNICOS/mk. You can ignore the error, but it does cause a slight problem: you cannot fully benefit from the h2ph utility (see h2ph) that can be used to convert C headers to Perl libraries, mainly used to be able to access from Perl the constants defined using C preprocessor, cpp. Because of the above error, parts of the converted headers will

be invisible. Luckily, these days the need for h2ph is rare.

- If building Perl with interpreter threads (ithreads), the **getgrent()**, **getgrnam()**, and **getgrgid()** functions cannot return the list of the group members due to a bug in the multithreaded support of UNICOS/mk. What this means is that in list context the functions will return only three values, not four.

### UTS

There are a few known test failures. (**Note:** the relevant information was available in *README.uts* until support for UTS was removed in Perl v5.18.0)

### VOS (Stratus)

When Perl is built using the native build process on VOS Release 14.5.0 and GNU C++/GNU Tools 2.0.1, all attempted tests either pass or result in TODO (ignored) failures.

### VMS

There should be no reported test failures with a default configuration, though there are a number of tests marked TODO that point to areas needing further debugging and/or porting work.

### Win32

In multi-CPU boxes, there are some problems with the I/O buffering: some output may appear twice.

### XML::Parser not working

Use XML::Parser 2.31 or later.

### z/OS (OS/390)

z/OS has rather many test failures but the situation is actually much better than it was in 5.6.0; it's just that so many new modules and tests have been added.

```
Failed Test                    Stat Wstat Total Fail  Failed  List of Failed
-------------------------------------------------------------------------------
../ext/Data/Dumper/t/dumper.t                357    8   2.24%  311 314 325 327
                                                               331 333 337 339
../ext/IO/lib/IO/t/io_unix.t                   5    4  80.00%  2-5
../ext/Storable/t/downgrade.t   12  3072   169   12   7.10%  14-15 46-47 78-79
                                                               110-111 150 161
../lib/ExtUtils/t/Constant.t   121 30976    48   48 100.00%  1-48
../lib/ExtUtils/t/Embed.t                    9    9 100.00%  1-9
op/pat.t                                   922    7   0.76%  665 776 785 832-
                                                               834 845
op/sprintf.t                               224    3   1.34%  98 100 136
op/tr.t                                     97    5   5.15%  63 71-74
uni/fold.t                                 780    6   0.77%  61 169 196 661
                                                               710-711
```

The failures in dumper.t and downgrade.t are problems in the tests, those in io_unix and sprintf are problems in the USS (UDP sockets and printf formats). The pat, tr, and fold failures are genuine Perl problems caused by EBCDIC (and in the pat and fold cases, combining that with Unicode). The Constant and Embed are probably problems in the tests (since they test Perl's ability to build extensions, and that seems to be working reasonably well.)

### Unicode Support on EBCDIC Still Spotty

Though mostly working, Unicode support still has problem spots on EBCDIC platforms. One such known spot are the `\p{}` and `\P{}` regular expression constructs for code points less than 256: the `pP` are testing for Unicode code points, not knowing about EBCDIC.

### Seen In Perl 5.7 But Gone Now

`Time::Piece` (previously known as `Time::Object`) was removed because it was felt that it didn't have enough value in it to be a core module. It is still a useful module, though, and is available from the CPAN.

Perl 5.8 unfortunately does not build anymore on AmigaOS; this broke accidentally at some point. Since there are not that many Amiga developers available, we could not get this fixed and tested in time for 5.8.0. Perl 5.6.1 still works for AmigaOS (as does the 5.7.2 development release).

The `PerlIO::Scalar` and `PerlIO::Via` (capitalised) were renamed as `PerlIO::scalar` and

`PerlIO::via` (all lowercase) just before 5.8.0. The main rationale was to have all core PerlIO layers to have all lowercase names. The "plugins" are named as usual, for example `PerlIO::via::QuotedPrint`.

The `threads::shared::queue` and `threads::shared::semaphore` were renamed as `Thread::Queue` and `Thread::Semaphore` just before 5.8.0. The main rationale was to have thread modules to obey normal naming, `Thread::` (the `threads` and `threads::shared` themselves are more pragma-like, they affect compile-time, so they stay lowercase).

## Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://bugs.perl.org/ . There may also be information at http://www.perl.com/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

## SEE ALSO

The *Changes* file for exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.

## HISTORY

Written by Jarkko Hietaniemi <*jhi@iki.fi*>.

## NAME

perldelta − what is new for perl v5.36.0

## DESCRIPTION

This document describes differences between the 5.34.0 release and the 5.36.0 release.

## Core Enhancements

### use v5.36

As always, `use v5.36` turns on the feature bundle for that version of Perl.

The 5.36 bundle enables the `signatures` feature. Introduced in Perl version 5.20.0, and modified several times since, the subroutine signatures feature is now no longer considered experimental. It is now considered a stable language feature and no longer prints a warning.

```
use v5.36;

sub add ($x, $y) {
  return $x + $y;
}
```

Despite this, certain elements of signatured subroutines remain experimental; see below.

The 5.36 bundle enables the `isa` feature. Introduced in Perl version 5.32.0, this operator has remained unchanged since then. The operator is now considered a stable language feature. For more detail see ‘‘Class Instance Operator’’ in perlop.

The 5.36 bundle also *disables* the features `indirect`, and `multidimensional`. These will forbid, respectively: the use of ‘‘indirect’’ method calls (like `$x = new Class;`); the use of a list expression as a hash key to simulate sparse multidimensional arrays. The specifics of these changes can be found in feature, but the short version is: this is a bit like having more `use strict` turned on, disabling features that cause more trouble than they’re worth.

Furthermore, `use v5.36` will also enable warnings as if you’d written `use warnings`.

Finally, with this release, the experimental `switch` feature, present in every feature bundle since they were introduced in v5.10, has been removed from the v5.36 bundle. If you want to use it (against our advice), you’ll have to enable it explicitly.

### −g command-line flag

A new command-line flag, −g, is available. It is a simpler alias for −0777.

For more information, see ‘‘−g’’ in perlrun.

### Unicode 14.0 is supported

See <https://www.unicode.org/versions/Unicode14.0.0/> for details.

### regex sets are no longer considered experimental

Prior to this release, the regex sets feature (officially named ‘‘Extended Bracketed Character Classes’’) was considered experimental. Introduced in Perl version 5.18.0, and modified several times since, this is now considered a stable language feature and its use no longer prints a warning. See ‘‘Extended Bracketed Character Classes’’ in perlrecharclass.

### Variable length lookbehind is mostly no longer considered experimental

Prior to this release, any form of variable length lookbehind was considered experimental. With this release the experimental status has been reduced to cover only lookbehind that contains capturing parenthesis. This is because it is not clear if

```
"aaz"=~/(?=z)(?<=(a|aa))/
```

should match and leave `$1` equaling ‘‘a’’ or ‘‘aa’’. Currently it will match the longest possible alternative, ‘‘aa’’. While we are confident that the overall construct will now match only when it should, we are not confident that we will keep the current ‘‘longest match’’ behavior.

### SIGFPE no longer deferred

Floating-point exceptions are now delivered immediately, in the same way as other ‘‘fault’’−like signals such as SIGSEGV. This means one has at least a chance to catch such a signal with a `$SIG{FPE}` handler, e.g. so that `die` can report the line in perl that triggered it.

### Stable boolean tracking

The ‘‘true’’ and ‘‘false’’ boolean values, often accessed by constructions like `!!0` and `!!1`, as well as being returned from many core functions and operators, now remember their boolean nature even through assignment into variables. The new function `is_bool()` in builtin can check whether a value has boolean nature.

This is likely to be useful when interoperating with other languages or data-type serialisation, among other places.

### iterating over multiple values at a time (experimental)

You can now iterate over multiple values at a time by specifying a list of lexicals within parentheses. For example,

```
for my ($key, $value) (%hash) { ... }
for my ($left, $right, $gripping) (@moties) { ... }
```

Prior to perl v5.36, attempting to specify a list after `for my` was a syntax error.

This feature is currently experimental and will cause a warning of category `experimental::for_list`. For more detail see ‘‘Compound Statements’’ in perlsyn. See also ‘‘builtin::indexed’’ in this document, which is a handy companion to n−at-a-time foreach.

### builtin functions (experimental)

A new core module builtin has been added, which provides documentation for new always-present functions that are built into the interpreter.

```
say "Reference type of arrays is ", builtin::reftype([]);
```

It also provides a lexical import mechanism for providing short name versions of these functions.

```
use builtin 'reftype';
say "Reference type of arrays is ", reftype([]);
```

This builtin function mechanism and the functions it provides are all currently **experimental**. We expect that `builtin` itself will cease to be experimental in the near future, but that individual functions in it may become stable on an ongoing basis. Other functions will be added to `builtin` over time.

For details, see builtin, but here’s a summary of builtin functions in v5.36:

builtin::trim
: This function treats its argument as a string, returning the result of removing all white space at its beginning and ending.

builtin::indexed
: This function returns a list twice as big as its argument list, where each item is preceded by its index within that list. This is primarily useful for using the new `foreach` syntax with multiple iterator variables to iterate over an array or list, while also tracking the index of each item:

```
use builtin 'indexed';

foreach my ($index, $val) (indexed @array) {
    ...
}
```

builtin::true, builtin::false, builtin::is_bool
: `true` and `false` return boolean true and false values. Perl is still perl, and doesn’t have strict typing of booleans, but these values will be known to have been created as booleans. `is_bool` will tell you whether a value was known to have been created as a boolean.

builtin::weaken, builtin::unweaken, builtin::is_weak
: These functions will, respectively: weaken a reference; strengthen a reference; and return whether a reference is weak. (A weak reference is not counted for garbage collection purposes. See perlref.) These can take the place of some similar routines in Scalar::Util.

builtin::blessed, builtin::refaddr, builtin::reftype
: These functions provide more data about references (or non-references, actually!) and can take the place of similar routines found in Scalar::Util.

builtin::ceil, builtin::floor

`ceil` returns the smallest integer greater than or equal to its argument. `floor` returns the largest integer less than or equal to its argument. These can take the place of similar routines found in POSIX.

### `defer` blocks (experimental)

This release adds support for `defer` blocks, which are blocks of code prefixed by the `defer` modifier. They provide a section of code which runs at a later time, during scope exit.

In brief, when a `defer` block is reached at runtime, its body is set aside to be run when the enclosing scope is exited. It is unlike a UNITCHECK (among other reasons) in that if the block *containing* the `defer` block is exited before the block is reached, it will not be run.

`defer` blocks can be used to take the place of ''scope guard'' objects where an object is passed a code block to be run by its destructor.

For more information, see ''defer blocks'' in perlsyn.

### try/catch can now have a `finally` block (experimental)

The experimental `try`/`catch` syntax has been extended to support an optional third block introduced by the `finally` keyword.

```
try {
    attempt();
    print "Success\n";
}
catch ($e) {
    print "Failure\n";
}
finally {
    print "This happens regardless\n";
}
```

This provides code which runs at the end of the `try`/`catch` construct, even if aborted by an exception or control-flow keyword. They are similar to `defer` blocks.

For more information, see ''Try Catch Exception Handling'' in perlsyn.

### non-ASCII delimiters for quote-like operators (experimental)

Perl traditionally has allowed just four pairs of string/pattern delimiters: `( )` `{ }` `[ ]` and `< >`, all in the ASCII range. Unicode has hundreds more possibilities, and using this feature enables many of them. When enabled, you can say `qrX X` for example, or `use utf8; qXstringX`. See ''The 'extra_paired_delimiters' feature'' in feature for details.

### `@_` is now experimental within signatured subs

Even though subroutine signatures are now stable, use of the legacy arguments array (`@_`) with a subroutine that has a signature *remains* experimental, with its own warning category. Silencing the `experimental::signatures` warning category is not sufficient to dismiss this. The new warning is emitted with the category name `experimental::args_array_with_signatures`.

Any subroutine that has a signature and tries to make use of the defaults argument array or an element thereof (`@_` or `$_[INDEX]`), either explicitly or implicitly (such as `shift` or `pop` with no argument) will provoke a warning at compile-time:

```
use v5.36;

sub f ($x, $y = 123) {
  say "The first argument is $_[0]";
}
```

```
Use of @_ in array element with signatured subroutine is experimental
at file.pl line 4.
```

The behaviour of code which attempts to do this is no longer specified, and may be subject to change in a future version.

## Incompatible Changes

### A physically empty sort is now a compile-time error

```
@a = sort @empty; # unaffected
@a = sort;        # now a compile-time error
@a = sort ();     # also a compile-time error
```

A bare sort used to be a weird way to create an empty list; now it croaks at compile time. This change is intended to free up some of the syntax space for possible future enhancements to `sort`.

## Deprecations

### `use VERSION` (where VERSION is below v5.11) after `use v5.11` is deprecated

When in the scope of `use v5.11` or later, a `use vX` line where *X* is lower than v5.11 will now issue a warning:

```
Downgrading a use VERSION declaration to below v5.11 is deprecated
```

For example:

```
use v5.14;
say "The say statement is permitted";
use v5.8;                               # This will print a warning
print "We must use print\n";
```

This is because the Perl team plans to change the behavior in this case. Since Perl v5.12 (and parts of v5.11), strict is enabled *unless it had previously been disabled*. In other words:

```
no strict;
use v5.12;  # will not enable strict, because "no strict" preceded it
$x = 1;     # permitted, despite no "my" declaration
```

In the future, this behavior will be eliminated and `use VERSION` will *always* enable strict for versions v5.12 and later.

Code which wishes to mix versions in this manner should use lexical scoping with block syntax to ensure that the differently versioned regions remain lexically isolated.

```
{
    use v5.14;
    say "The say statement is permitted";
}

{
    use v5.8;                           # No warning is emitted
    print "We must use print\n";
}
```

Of course, this is probably not something you ever need to do! If the first block compiles, it means you're using perl v5.14.0 or later.

## Performance Enhancements

- We now probe for compiler support for C11 thread local storage, and where available use this for ''implicit context'' for XS extensions making API calls for a threaded Perl build. This requires fewer function calls at the C level than POSIX thread specific storage. We continue to use the the pthreads approach if the C11 approach is not available.

  *Configure* run with the defaults will build an unthreaded Perl (which is slightly faster), but most operating systems ship a threaded Perl.

- Perl can now be configured to no longer allocate keys for large hashes from the shared string table.

  The same internal datatype (PVHV) is used for all of

  - Symbol tables
  - Objects (by default)
  - Associative arrays

  The shared string table was originally added to improve performance for blessed hashes used as objects, because every object instance has the same keys, so it is an optimisation to share memory

between them. It also makes sense for symbol tables, where derived classes will have the same keys (typically method names), and the OP trees built for method calls can also share memory. The shared string table behaves roughly like a cache for hash keys.

But for hashes actually used as associative arrays − mapping keys to values − typically the keys are not re-used in other hashes. For example, ‘‘seen’’ hashes are keyed by object IDs (or addresses), and logically these keys won’t repeat in other hashes.

Storing these ‘‘used just once’’ keys in the shared string table increases CPU and RAM use for no gain. For such keys the shared string table behaves as a cache with a 0% hit rate. Storing all the keys there increases the total size of the shared string table, as well as increasing the number of times it is resized as it grows. **Worse** − in any environment that has ‘‘copy on write’’ memory for child process (such as a pre-forking server), the memory pages used for the shared string table rapidly need to be copied as the child process manipulates hashes. Hence if most of the shared string table is such that keys are used only in one place, there is no benefit from re-use within the perl interpreter, but a high cost due to more pages for the OS to copy.

The perl interpreter can now be Configured to disable shared hash keys for ‘‘large’’ hashes (that are neither objects nor symbol tables). To do so, add `-Accflags='-DPERL_USE_UNSHARED_KEYS_IN_LARGE_HASHES'` to your *Configure* options. ‘‘Large’’ is a heuristic — currently the heuristic is that sharing is disabled when adding a key to a hash triggers allocation of more storage, and the hash has more than 42 keys.

This **might** cause slightly increased memory usage for programs that create (unblessed) data structures that contain multiple large hashes that share the same keys. But generally our testing suggests that for the specific cases described it is a win, and other code is unaffected.

- In certain scenarios, creation of new scalars is now noticeably faster.

  For example, the following code is now executing ~30% faster:

  ```
  $str = "A" x 64;
  for (0..1_000_000) {
      @svs = split //, $str
  }
  ```

  (You can read more about this one in [perl #19414] <https://github.com/Perl/perl5/pull/19414>.)

## Modules and Pragmata

### Updated Modules and Pragmata

- Archive::Tar has been upgraded from version 2.38 to 2.40.
- Attribute::Handlers has been upgraded from version 1.01 to 1.02.
- attributes has been upgraded from version 0.33 to 0.34.
- B has been upgraded from version 1.82 to 1.83.
- B::Concise has been upgraded from version 1.004 to 1.006.
- B::Deparse has been upgraded from version 1.56 to 1.64.
- bignum has been upgraded from version 0.51 to 0.65.
- charnames has been upgraded from version 1.48 to 1.50.
- Compress::Raw::Bzip2 has been upgraded from version 2.101 to 2.103.
- Compress::Raw::Zlib has been upgraded from version 2.101 to 2.105.
- CPAN has been upgraded from version 2.28 to 2.33.
- Data::Dumper has been upgraded from version 2.179 to 2.184.
- DB_File has been upgraded from version 1.855 to 1.857.
- Devel::Peek has been upgraded from version 1.30 to 1.32.
- Devel::PPPort has been upgraded from version 3.62 to 3.68.

- diagnostics has been upgraded from version 1.37 to 1.39.
- Digest has been upgraded from version 1.19 to 1.20.
- DynaLoader has been upgraded from version 1.50 to 1.52.
- Encode has been upgraded from version 3.08 to 3.17.
- Errno has been upgraded from version 1.33 to 1.36.
- experimental has been upgraded from version 0.024 to 0.028.
- Exporter has been upgraded from version 5.76 to 5.77.
- ExtUtils::MakeMaker has been upgraded from version 7.62 to 7.64.
- ExtUtils::Miniperl has been upgraded from version 1.10 to 1.11.
- ExtUtils::ParseXS has been upgraded from version 3.43 to 3.45.
- ExtUtils::Typemaps has been upgraded from version 3.43 to 3.45.
- Fcntl has been upgraded from version 1.14 to 1.15.
- feature has been upgraded from version 1.64 to 1.72.
- File::Compare has been upgraded from version 1.1006 to 1.1007.
- File::Copy has been upgraded from version 2.35 to 2.39.
- File::Fetch has been upgraded from version 1.00 to 1.04.
- File::Find has been upgraded from version 1.39 to 1.40.
- File::Glob has been upgraded from version 1.33 to 1.37.
- File::Spec has been upgraded from version 3.80 to 3.84.
- File::stat has been upgraded from version 1.09 to 1.12.
- FindBin has been upgraded from version 1.52 to 1.53.
- GDBM_File has been upgraded from version 1.19 to 1.23.
- Hash::Util has been upgraded from version 0.25 to 0.28.
- Hash::Util::FieldHash has been upgraded from version 1.21 to 1.26.
- HTTP::Tiny has been upgraded from version 0.076 to 0.080.
- I18N::Langinfo has been upgraded from version 0.19 to 0.21.
- if has been upgraded from version 0.0609 to 0.0610.
- IO has been upgraded from version 1.46 to 1.50.
- IO-Compress has been upgraded from version 2.102 to 2.106.
- IPC::Open3 has been upgraded from version 1.21 to 1.22.
- JSON::PP has been upgraded from version 4.06 to 4.07.
- libnet has been upgraded from version 3.13 to 3.14.
- Locale::Maketext has been upgraded from version 1.29 to 1.31.
- Math::BigInt has been upgraded from version 1.999818 to 1.999830.
- Math::BigInt::FastCalc has been upgraded from version 0.5009 to 0.5012.
- Math::BigRat has been upgraded from version 0.2614 to 0.2621.
- Module::CoreList has been upgraded from version 5.20210520 to 5.20220520.
- mro has been upgraded from version 1.25_001 to 1.26.
- NEXT has been upgraded from version 0.68 to 0.69.
- Opcode has been upgraded from version 1.50 to 1.57.
- open has been upgraded from version 1.12 to 1.13.

- overload has been upgraded from version 1.33 to 1.35.
- perlfaq has been upgraded from version 5.20210411 to 5.20210520.
- PerlIO has been upgraded from version 1.11 to 1.12.
- Pod::Functions has been upgraded from version 1.13 to 1.14.
- Pod::Html has been upgraded from version 1.27 to 1.33.
- Pod::Simple has been upgraded from version 3.42 to 3.43.
- POSIX has been upgraded from version 1.97 to 2.03.
- re has been upgraded from version 0.41 to 0.43.
- Scalar::Util has been upgraded from version 1.55 to 1.62.
- sigtrap has been upgraded from version 1.09 to 1.10.
- Socket has been upgraded from version 2.031 to 2.033.
- sort has been upgraded from version 2.04 to 2.05.
- Storable has been upgraded from version 3.23 to 3.26.
- Sys::Hostname has been upgraded from version 1.23 to 1.24.
- Test::Harness has been upgraded from version 3.43 to 3.44.
- Test::Simple has been upgraded from version 1.302183 to 1.302190.
- Text::ParseWords has been upgraded from version 3.30 to 3.31.
- Text::Tabs has been upgraded from version 2013.0523 to 2021.0814.
- Text::Wrap has been upgraded from version 2013.0523 to 2021.0814.
- threads has been upgraded from version 2.26 to 2.27.
- threads::shared has been upgraded from version 1.62 to 1.64.
- Tie::Handle has been upgraded from version 4.2 to 4.3.
- Tie::Hash has been upgraded from version 1.05 to 1.06.
- Tie::Scalar has been upgraded from version 1.05 to 1.06.
- Tie::SubstrHash has been upgraded from version 1.00 to 1.01.
- Time::HiRes has been upgraded from version 1.9767 to 1.9770.
- Unicode::Collate has been upgraded from version 1.29 to 1.31.
- Unicode::Normalize has been upgraded from version 1.28 to 1.31.
- Unicode::UCD has been upgraded from version 0.75 to 0.78.
- UNIVERSAL has been upgraded from version 1.13 to 1.14.
- version has been upgraded from version 0.9928 to 0.9929.
- VMS::Filespec has been upgraded from version 1.12 to 1.13.
- VMS::Stdio has been upgraded from version 2.45 to 2.46.
- warnings has been upgraded from version 1.51 to 1.58.
- Win32 has been upgraded from version 0.57 to 0.59.
- XS::APItest has been upgraded from version 1.16 to 1.22.
- XS::Typemap has been upgraded from version 0.18 to 0.19.
- XSLoader has been upgraded from version 0.30 to 0.31.

## Documentation

### New Documentation

*Porting/vote_admin_guide.pod*

This document provides the process for administering an election or vote within the Perl Core Team.

### Changes to Existing Documentation

We have attempted to update the documentation to reflect the changes listed in this document. If you find any we have missed, open an issue at <https://github.com/Perl/perl5/issues>.

Additionally, the following selected changes have been made:

*perlapi*

- This has been cleaned up some, and more than 80% of the (previously many) undocumented functions have now either been documented or deemed to have been inappropriately marked as API.

  As always, Patches Welcome!

*perldeprecation*

- notes the new location for functions moved from Pod::Html to Pod::Html::Util that are no longer intended to be used outside of core.

*perlexperiment*

- notes the `:win32` IO pseudolayer is removed (this happened in 5.35.2).

*perlgov*

- The election process has been finetuned to allow the vote to be skipped if there are no more candidates than open seats.
- A special election is now allowed to be postponed for up to twelve weeks, for example until a normal election.

*perlop*

- now notes that an invocant only needs to be an object or class name for method calls, not for subroutine references.

*perlre*

- Updated to discourage the use of the /d regexp modifier.

*perlrun*

- **−?** is now a synonym for **−h**
- **−g** is now a synonym for **−0777**

## Diagnostics

The following additions or changes have been made to diagnostic output, including warnings and fatal error messages. For the complete list of diagnostic messages, see perldiag.

### New Diagnostics

*New Errors*

- Can’t ‘‘%s’’ out of a ‘‘defer’’ block

  (F) An attempt was made to jump out of the scope of a defer block by using a control-flow statement such as `return`, `goto` or a loop control. This is not permitted.

- Can’t modify `%s` in `%s` (for scalar assignment to `undef`)

  Attempting to perform a scalar assignment to `undef`, for example via `undef = $foo;`, previously triggered a fatal runtime error with the message "Modification of a read-only value attempted.‘‘ It is more helpful to detect such attempted assignments prior to runtime, so they are now compile time errors, resulting in the message ’’Can’t modify undef operator in scalar assignment".

- panic: newFORLOOP, `%s`

  The parser failed an internal consistency check while trying to parse a `foreach` loop.

*New Warnings*

- Built-in function ’%s’ is experimental

  A call is being made to a function in the `builtin::` namespace, which is currently experimental.

- defer is experimental

  The `defer` block modifier is experimental. If you want to use the feature, disable the warning with `no warnings 'experimental::defer'`, but know that in doing so you are taking the risk that your code may break in a future Perl version.

- Downgrading a use VERSION declaration to below v5.11 is deprecated

  This warning is emitted on a `use VERSION` statement that requests a version below v5.11 (when the effects of `use strict` would be disabled), after a previous declaration of one having a larger number (which would have enabled these effects)

- for my (...) is experimental

  This warning is emitted if you use `for` to iterate multiple values at a time. This syntax is currently experimental and its behaviour may change in future releases of Perl.

- Implicit use of `@_` in `%s` with signatured subroutine is experimental

  An expression that implicitly involves the `@_` arguments array was found in a subroutine that uses a signature.

- Use of `@_` in `%s` with signatured subroutine is experimental

  An expression involving the `@_` arguments array was found in a subroutine that uses a signature.

- Wide character in `$0`

  Attempts to put wide characters into the program name (`$0`) now provoke this warning.

### Changes to Existing Diagnostics

- '/' does not take a repeat count in `%s`

  This warning used to not include the `in %s`.

- Subroutine `%s` redefined

  Localized subroutine redefinitions no longer trigger this warning.

- unexpected constant lvalue entersub entry via type/targ `%d:`%d" now has a panic prefix

  This makes it consistent with other checks of internal consistency when compiling a subroutine.

- Useless use of sort in scalar context is now in the new `scalar` category.

  When `sort` is used in scalar context, it provokes a warning that doing this is not useful. This warning used to be in the `void` category. A new category for warnings about scalar context has now been added, called `scalar`.

- Removed a number of diagnostics

  Many diagnostics that have been removed from the perl core across many years have now *also* been removed from the documentation.

## Configuration and Compilation

- The Perl C source code now uses some C99 features, which we have verified are supported by all compilers we target. This means that Perl's headers now contain some code that is legal in C99 but not C89.

  This may cause problems for some XS modules that unconditionally add `-Werror=declaration-after-statement` to their C compiler flags if compiling with gcc or clang. Earlier versions of Perl support long obsolete compilers that are strict in rejecting certain C99 features, particularly mixed declarations and code, and hence it makes sense for XS module authors to audit that their code does not violate this. However, doing this is now only possible on these earlier versions of Perl, hence these modules need to be changed to only add this flag for `<$] < 5.035005>`.

- The makedepend step is now run in parallel by using make

  When using MAKEFLAGS=-j8, this significantly reduces the time required for:

  ```
  sh ./makedepend MAKE=make cflags
  ```

- *Configure* now tests whether `#include <xlocale.h>` is required to use the POSIX 1003 thread-safe locale functions or some related extensions. This prevents problems where a non-public *xlocale.h* is removed in a library update, or *xlocale.h* isn't intended for public use. (github #18936 <https://github.com/Perl/perl5/pull/18936>)

## Testing

Tests were added and changed to reflect the other additions and changes in this release.

## Platform Support

### Windows

- Support for old MSVC++ (pre−VC12) has been removed

  These did not support C99 and hence can no longer be used to compile perl.

- Support for compiling perl on Windows using Microsoft Visual Studio 2022 (containing Visual C++ 14.3) has been added.

- The :win32 IO layer has been removed. This experimental replacement for the :unix layer never reached maturity in its nearly two decades of existence.

### VMS

`keys %ENV` on VMS returns consistent results

On VMS entries in the `%ENV` hash are loaded from the OS environment on first access, hence the first iteration of `%ENV` requires the entire environment to be scanned to find all possible keys. This initialisation had always been done correctly for full iteration, but previously was not happening for `%ENV` in scalar context, meaning that `scalar %ENV` would return 0 if called before any other `%ENV` access, or would only return the count of keys accessed if there had been no iteration.

These bugs are now fixed − `%ENV` and `keys %ENV` in scalar context now return the correct result − the count of all keys in the environment.

### Discontinued Platforms

AT&T UWIN

UWIN is a UNIX compatibility layer for Windows. It was last released in 2012 and has been superseded by Cygwin these days.

DOS/DJGPP

DJGPP is a port of the GNU toolchain to 32−bit x86 systems running DOS. The last known attempt to build Perl on it was on 5.20, which only got as far as building miniperl.

NetWare

Support code for Novell NetWare has been removed. NetWare was a server operating system by Novell. The port was last updated in July 2002, and the platform itself in May 2009.

Unrelated changes accidentally broke the build for the NetWare port in September 2009, and in 12 years no-one has reported this.

### Platform-Specific Notes

z/OS

This update enables us to build EBCDIC static/dynamic and 31−bit/64−bit addressing mode Perl. The number of tests that pass is consistent with the baseline before these updates.

These changes also provide the base support to be able to provide ASCII static/dynamic and 31−bit/64−bit addressing mode Perl.

The z/OS (previously called OS/390) README was updated to describe ASCII and EBCDIC builds.

## Internal Changes

- Since the removal of PERL_OBJECT in Perl 5.8, PERL_IMPLICIT_CONTEXT and MULTIPLICITY have been synonymous and they were being used interchangeably. To simplify the code, all instances of PERL_IMPLICIT_CONTEXT have been replaced with MULTIPLICITY.

  PERL_IMPLICIT_CONTEXT will remain defined for compatibility with XS modules.

- The API constant formerly named `G_ARRAY`, indicating list context, has now been renamed to a more accurate `G_LIST`. A compatibilty macro `G_ARRAY` has been added to allow existing code to work unaffected. New code should be written using the new constant instead. This is supported by `Devel::PPPort` version 3.63.

- Macros have been added to *perl.h* to facilitate version comparisons: `PERL_GCC_VERSION_GE`, `PERL_GCC_VERSION_GT`, `PERL_GCC_VERSION_LE` and `PERL_GCC_VERSION_LT`.

  Inline functions have been added to *embed.h* to determine the position of the least significant 1 bit in a word: `lsbit_pos32` and `lsbit_pos64`.

- `Perl_ptr_table_clear` has been deleted. This has been marked as deprecated since v5.14.0 (released in 2011), and is not used by any code on CPAN.

- Added new boolean macros and functions. See ‘‘Stable boolean tracking’’ for related information and perlapi for documentation.

  - sv_setbool
  - sv_setbool_mg
  - SvIsBOOL

- Added 4 missing functions for dealing with RVs:

  - sv_setrv_noinc
  - sv_setrv_noinc_mg
  - sv_setrv_inc
  - sv_setrv_inc_mg

- `xs_handshake()`'s two failure modes now provide distinct messages.

- Memory for hash iterator state (`struct xpvhv_aux`) is now allocated as part of the hash body, instead of as part of the block of memory allocated for the main hash array.

- A new **phase_name()** interface provides access to the name for each interpreter phase (i.e., PL_phase value).

- The `pack` behavior of `U` has changed for EBCDIC.

- New equality-test functions `sv_numeq` and `sv_streq` have been added, along with `..._flags`-suffixed variants. These expose a simple and consistent API to perform numerical or string comparison which is aware of operator overloading.

- Reading the string form of an integer value no longer sets the flag `SVf_POK`. The string form is still cached internally, and still re-read directly by the macros `SvPV(sv)` *etc* (inline, without calling a C function). XS code that already calls the APIs to get values will not be affected by this change. XS code that accesses flags directly instead of using API calls to express its intent *might* break, but such code likely is already buggy if passed some other values, such as floating point values or objects with string overloading.

  This small change permits code (such as JSON serializers) to reliably determine between

  - a value that was initially **written** as an integer, but then **read** as a string

    ```
    my $answer = 42;
    print "The answer is $answer\n";
    ```

  - that same value that was initially **written** as a string, but then **read** as an integer

    ```
    my $answer = "42";
    print "That doesn't look right\n"
        unless $answer == 6 * 9;
    ```

  For the first case (originally written as an integer), we now have:

  ```
  use Devel::Peek;
  my $answer = 42;
  Dump ($answer);
  my $void = "$answer";
  print STDERR "\n";
  Dump($answer)
  ```

```
SV = IV(0x562538925778) at 0x562538925788
  REFCNT = 1
  FLAGS = (IOK,pIOK)
  IV = 42

SV = PVIV(0x5625389263c0) at 0x562538925788
  REFCNT = 1
  FLAGS = (IOK,pIOK,pPOK)
  IV = 42
  PV = 0x562538919b50 "42"\0
  CUR = 2
  LEN = 10
```

For the second (originally written as a string), we now have:

```
use Devel::Peek;
my $answer = "42";
Dump ($answer);
my $void = $answer == 6 * 9;
print STDERR "\n";
Dump($answer)'


SV = PV(0x5586ffe9bfb0) at 0x5586ffec0788
  REFCNT = 1
  FLAGS = (POK,IsCOW,pPOK)
  PV = 0x5586ffee7fd0 "42"\0
  CUR = 2
  LEN = 10
  COW_REFCNT = 1

SV = PVIV(0x5586ffec13c0) at 0x5586ffec0788
  REFCNT = 1
  FLAGS = (IOK,POK,IsCOW,pIOK,pPOK)
  IV = 42
  PV = 0x5586ffee7fd0 "42"\0
  CUR = 2
  LEN = 10
  COW_REFCNT = 1
```

(One can't rely on the presence or absence of the flag `SVf_IsCOW` to determine the history of operations on a scalar.)

Previously both cases would be indistinguishable, with all 4 flags set:

```
SV = PVIV(0x55d4d62edaf0) at 0x55d4d62f0930
  REFCNT = 1
  FLAGS = (IOK,POK,pIOK,pPOK)
  IV = 42
  PV = 0x55d4d62e1740 "42"\0
  CUR = 2
  LEN = 10
```

(and possibly `SVf_IsCOW`, but not always)

This now means that if XS code *really* needs to determine which form a value was first written as, it should implement logic roughly

```
if (flags & SVf_IOK|SVf_NOK) && !(flags & SVf_POK)
    serialize as number
else if (flags & SVf_POK)
    serialize as string
else
    the existing guesswork ...
```

Note that this doesn't cover ''dualvars'' − scalars that report different values when asked for their string form or number form (such as `$!`). Most serialization formats cannot represent such duplicity.

*The existing guesswork* remains because as well as dualvars, values might be `undef`, references, overloaded references, typeglobs and other things that Perl itself can represent but do not map one-to-one into external formats, so need some amount of approximation or encapsulation.

- `sv_dump` (and Devel::PeekXs `Dump` function) now escapes high-bit octets in the PV as hex rather than octal. Since most folks understand hex more readily than octal, this should make these dumps a bit more legible. This does **not** affect any other diagnostic interfaces like `pv_display`.

## Selected Bug Fixes

- **utime()** now correctly sets errno/`$!` when called on a closed handle.
- The flags on the OPTVAL parameter to **setsockopt()** were previously checked before magic was called, possibly treating a numeric value as a packed buffer or vice versa. It also ignored the UTF−8 flag, potentially treating the internal representation of an upgraded SV as the bytes to supply to the **setsockopt()** system call. (github #18660 <https://github.com/Perl/perl5/issues/18660>)
- Only set IOKp, not IOK on $) and $(. This was issue #18955 <https://github.com/Perl/perl5/issues/18955>: This will prevent serializers from serializing these variables as numbers (which loses the additional groups). This restores behaviour from 5.16
- Use of the `mktables` debugging facility would cause perl to croak since v5.31.10; this problem has now been fixed.
- `makedepend` logic is now compatible with BSD make (fixes GH #19046 <https://github.com/Perl/perl5/issues/19046>).
- Calling `untie` on a tied hash that is partway through iteration now frees the iteration state immediately.

  Iterating a tied hash causes perl to store a copy of the current hash key to track the iteration state, with this stored copy passed as the second parameter to `NEXTKEY`. This internal state is freed immediately when tie hash iteration completes, or if the hash is destroyed, but due to an implementation oversight, it was not freed if the hash was untied. In that case, the internal copy of the key would persist until the earliest of

  1. `tie` was called again on the same hash
  2. The (now untied) hash was iterated (ie passed to any of `keys`, `values` or `each`)
  3. The hash was destroyed.

  This inconsistency is now fixed − the internal state is now freed immediately by `untie`.

  As the precise timing of this behaviour can be observed with pure Perl code (the timing of `DESTROY` on objects returned from `FIRSTKEY` and `NEXTKEY`) it's just possible that some code is sensitive to it.
- The `Internals::getcwd()` function added for bootstrapping miniperl in perl 5.30.0 is now only available in miniperl. [github #19122]
- Setting a breakpoint on a BEGIN or equivalently a `use` statement could cause a memory write to a freed `dbstate` op. [GH #19198 <https://github.com/Perl/perl5/issues/19198>]
- When bareword filehandles are disabled, the parser was interpreting any bareword as a filehandle, even when immediatey followed by parens.

## Errata From Previous Releases

- perl5300delta mistakenly identified a CVE whose correct identification is CVE-2015-1592.

## Obituaries

Raun ''Spider'' Boardman (SPIDB on CPAN), author of at least 66 commits to the Perl 5 core distribution between 1996 and 2002, passed away May 24, 2021 from complications of COVID. He will be missed.

David H. Adler (DHA) passed away on November 16, 2021. In 1997, David co-founded NY.pm, the first Perl user group, and in 1998 co-founded Perl Mongers to help establish other user groups across the globe. He was a frequent attendee at Perl conferences in both North America and Europe and well known for his role in organizing *Bad Movie Night* celebrations at those conferences. He also contributed to the work of the Perl Foundation, including administering the White Camel awards for community service. He will be missed.

## Acknowledgements

Perl 5.36.0 represents approximately a year of development since Perl 5.34.0 and contains approximately 250,000 lines of changes across 2,000 files from 82 authors.

Excluding auto-generated files, documentation and release tools, there were approximately 190,000 lines of changes to 1,300 .pm, .t, .c and .h files.

Perl continues to flourish into its fourth decade thanks to a vibrant community of users and developers. The following people are known to have contributed the improvements that became Perl 5.36.0:

Alyssa Ross, Andrew Fresh, Aristotle Pagaltzis, Asher Mancinelli, Atsushi Sugawara, Ben Cornett, Bernd, Biswapriyo Nath, Brad Barden, Bram, Branislav Zahradník, brian d foy, Chad Granum, Chris 'BinGOs' Williams, Christian Walde (Mithaldu), Christopher Yeleighton, Craig A. Berry, cuishuang, Curtis Poe, Dagfinn Ilmari Mannsåker, Dan Book, Daniel Laügt, Dan Jacobson, Dan Kogai, Dave Cross, Dave Lambley, David Cantrell, David Golden, David Marshall, David Mitchell, E. Choroba, Eugen Konkov, Felipe Gasper, François Perrad, Graham Knop, H.Merijn Brand, Hugo van der Sanden, Ilya Sashcheka, Ivan Panchenko, Jakub Wilk, James E Keenan, James Raspass, Karen Etheridge, Karl Williamson, Leam Hall, Leon Timmermans, Magnus Woldrich, Matthew Horsfall, Max Maischein, Michael G Schwern, Michiel Beijen, Mike Fulton, Neil Bowers, Nicholas Clark, Nicolas R, Niyas Sait, Olaf Alders, Paul Evans, Paul Marquess, Petar-Kaleychev, Pete Houston, Renee Baecker, Ricardo Signes, Richard Leach, Robert Rothenberg, Sawyer X, Scott Baker, Sergey Poznyakoff, Sergey Zhmylove, Sisyphus, Slaven Rezic, Steve Hay, Sven Kirmess, TAKAI Kousuke, Thibault Duponchelle, Todd Rinaldo, Tomasz Konojacki, Tomoyuki Sadahiro, Tony Cook, Unicode Consortium, Yves Orton, XXXXXX XXXXXXXX.

The list above is almost certainly incomplete as it is automatically generated from version control history. In particular, it does not include the names of the (very much appreciated) contributors who reported issues to the Perl bug tracker.

Many of the changes included in this version originated in the CPAN modules included in Perl's core. We're grateful to the entire CPAN community for helping Perl to flourish.

For a more complete list of all of Perl's historical contributors, please

## Reporting Bugs

If you find what you think is a bug, you might check the perl bug database at <https://github.com/Perl/perl5/issues>. There may also be information at <http://www.perl.org/>, the Perl Home Page.

If you believe you have an unreported bug, please open an issue at <https://github.com/Perl/perl5/issues>. Be sure to trim your bug down to a tiny but sufficient test case.

If the bug you are reporting has security implications which make it inappropriate to send to a public issue tracker, then see ''SECURITY VULNERABILITY CONTACT INFORMATION'' in perlsec for details of how to report the issue.

## Give Thanks

If you wish to thank the Perl 5 Porters for the work we had done in Perl 5, you can do so by running the `perlthanks` program:

```
perlthanks
```

This will send an email to the Perl 5 Porters list with your show of thanks.

## SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.