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` 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.

`uselocale` **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 `usefeature'unicode_strings'`, except taint rules are followed. Sorting remains by code point order in this release. [perl #56820].

`uselocale` **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 `uselocale` 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 `uselocale`. 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 `uselocale`.

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(1) 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(1) 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(1)

This document was removed (actually, renamed perlgit(1) 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(1)

- New sections have been added to document the new index/value array slice and key/value hash slice syntax.

perldebguts(1)

- The `DB::goto` and `DB::lsub` debugger subroutines are now documented. [perl #77680]

perlexperiment(1)

- `\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(1) and perlsub.)

perlfunc(1)

- `chop` and `chomp` now note that they can reset the hash iterator.
- `exec`'s handling of arguments is now more clearly documented.
- `eval EXPR`no w has caveats about expanding floating point numbers in some locales.
- `goto EXPR`is no w 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(1)

- 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(1)

- The Super Quick Patch Guide section has been updated.

perlhacktips(1)

- The documentation has been updated to include some more examples of `gdb` usage.

perllexwarn(1)

- The perllexwarn(1) 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(1) as a stub that points to it. This change consolidates all documentation for lexical warnings in a single place.

perllocale(1)

- The documentation now mentions *fc()* and `\F`, and includes many clarifications and corrections in general.

perlop(1)

- The language design of Perl has always called for monomorphic operators. This is now mentioned explicitly.

perlopentut(1)

- 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(1)

- 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(1)

- The documentation has been updated in the light of recent changes to *regcomp.c*.

perlsub(1)

- 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(1)

- There is now a JavaScript section.

perlunicode(1)

- The documentation has been updated to reflect `Bidi_Class` changes in Unicode 6.3.

perlvar(1)

- 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(1)

- 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(1) 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(1)

- *perlbug(1)* now has a `-p` option for attaching patches with a bug report.
- perlbug(1) has been modified to supply the report template with CRLF line endings on Windows. [perl #121277 <https://rt.perl.org/Public/Bug/Display.html?id=121277>]
- perlbug(1) 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. [perl #121505 <https://rt.perl.org/Public/Bug/Display.html?id=121505>]

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(1) 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). [perl #121119 <https://rt.perl.org/Public/Bug/Display.html?id=121119>]
- 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. [perl #121395 <https://rt.perl.org/Public/Bug/Display.html?id=121395>]
- 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. [perl #121230 <https://rt.perl.org/Public/Bug/Display.html?id=121230>]

- 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.) [perl #121471 <https://rt.perl.org/Public/Bug/Display.html?id=121471>]
- 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. [perl #121643 <https://rt.perl.org/Public/Bug/Display.html?id=121643>]
- The test harness now has no failures when perl is built on a FAT drive with the Windows OS on an NTFS drive. [perl #21442 <https://rt.perl.org/Public/Bug/Display.html?id=21442>]
- 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;`. [perl #121721 <https://rt.perl.org/Public/Bug/Display.html?id=121721>]
- Introduced by perl #113536 <https://rt.perl.org/Public/Bug/Display.html?id=113536>, 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 psuedo-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. [perl #121676 <https://rt.perl.org/Public/Bug/Display.html?id=121676>]

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`, b ut 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(1), ''sv_2iv_flags'' in perlapi(1), ''sv_2uv_flags'' in perlapi(1), and ''sv_2nv_flags'' in perlapi(1) 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 litteral 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 `$1` etc. from any matches inside the block) in its FETCH method. This was not the case if a reference to an overloaded object was the last thing assigned to the tied variable. Instead, the match variables referred to the outer pattern during the FETCH call.
- Fix unexpected tainting via regexp using locale. Previously, under certain conditions, the use of character classes could cause tainting when it shouldn’t. Some character classes are locale-dependent, but before this patch, sometimes tainting was happening even for character classes that don’t depend on the locale. [perl #120675]
- Under certain conditions, Perl would throw an error if in an lookbehind assertion in a regexp, the assertion referred to a named subpattern, complaining the lookbehind was variable when it wasn’t. This has been fixed. [perl #120600], [perl #120618]. The current fix may be improved on in the future.
- `$^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(1)` - 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 precedeence. 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(1) 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 w arning 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 w arn. 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 DESTR OY method on the sub that is being redefined, and that method assigns a subroutine to the same slot (`*foo = sub {}`), `$_[0]` is no longer left pointing to a freed scalar. Now DESTROY is delayed until the new subroutine has been installed.
- On Windows, perl no longer calls *CloseHandle()* on a socket handle. This makes debugging easier on Windows by removing certain irrelevant bad handle exceptions. It also fixes a race condition that made socket functions randomly fail in a Perl process with multiple OS threads, and possible test failures in *dist/IO/t/cachepropagate-tcp.t*. [perl #120091/118059]
- Formats involving UTF-8 encoded strings, or strange vars like ties, overloads, or stringified refs (and in recent perls, pure NOK vars) would generally do the wrong thing in formats when the var is treated as a string and repeatedly chopped, as in `^<<<~~` and similar. This has now been resolved. [perl #33832/45325/113868/119847/119849/119851]
- `semctl(..., SETVAL, ...)`w ould set the semaphore to the top 32-bits of the supplied integer instead of the bottom 32-bits on 64-bit big-endian systems. [perl #120635]
- `readdir()` now only sets `$!` 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 behavious 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 b ut points to a directory a temporary file cannot be created in. [perl #120951]
- `for ( $h{k} || '' )` no longer auto-vi vifies `$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://rt.perl.org/Ticket/Display.html?id=120835> 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 Mannsåker, 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 Písa, 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, 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 http://rt.perl.org/perlbug/ http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the perlbug(1) 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.