

NAME

perlmodlib - constructing new Perl modules and finding existing ones

THE PERL MODULE LIBRARY

Many modules are included in the Perl distribution. These are described below, and all end in *.pm*. You may discover compiled library files (usually ending in *.so*) or small pieces of modules to be autoloading (ending in *.al*); these were automatically generated by the installation process. You may also discover files in the library directory that end in either *.pl* or *.ph*. These are old libraries supplied so that old programs that use them still run. The *.pl* files will all eventually be converted into standard modules, and the *.ph* files made by **h2ph** will probably end up as extension modules made by **h2xs**. (Some *.ph* values may already be available through the POSIX, Errno, or Fcntl modules.) The **pl2pm** file in the distribution may help in your conversion, but it's just a mechanical process and therefore far from bulletproof.

Pragmatic Modules

They work somewhat like compiler directives (pragmata) in that they tend to affect the compilation of your program, and thus will usually work well only when used within a `use`, or `no`. Most of these are lexically scoped, so an inner BLOCK may countermand them by saying:

```
no integer;
no strict 'refs';
no warnings;
```

which lasts until the end of that BLOCK.

Some pragmas are lexically scoped—typically those that affect the `%^H` hints variable. Others affect the current package instead, like `use vars` and `use subs`, which allow you to predeclare a variables or subroutines within a particular *file* rather than just a block. Such declarations are effective for the entire file for which they were declared. You cannot rescind them with `no vars` or `no subs`.

The following pragmas are defined (and have their own documentation).

<code>arybase</code>	Set indexing base via <code>\$_</code>
<code>attributes</code>	Get/set subroutine or variable attributes
<code>autodie</code>	Replace functions with ones that succeed or die with lexical scope
<code>autodie::exception</code>	Exceptions from autodying functions.
<code>autodie::exception::system</code>	Exceptions from autodying <i>system()</i> .
<code>autodie::hints</code>	Provide hints about user subroutines to autodie
<code>autodie::skip</code>	Skip a package when throwing autodie exceptions
<code>autouse</code>	Postpone load of modules until a function is used
<code>base</code>	Establish an ISA relationship with base classes at compile time
<code>bigint</code>	Transparent BigInteger support for Perl
<code>bignum</code>	Transparent BigNumber support for Perl
<code>bigrat</code>	Transparent BigNumber/BigRational support for Perl
<code>blib</code>	Use MakeMaker's uninstalled version of a package
<code>bytes</code>	Expose the individual bytes of characters
<code>charnings</code>	Access to Unicode character names and named character sequences; also define character names
<code>constant</code>	Declare constants
<code>deprecate</code>	Perl pragma for deprecating the core version of a module

diagnostics	Produce verbose warning diagnostics
encoding	Allows you to write your script in non-ASCII and non-UTF-8
encoding::warnings	Warn on implicit encoding conversions
experimental	Experimental features made easy
feature	Enable new features
fields	Compile-time class fields
filetest	Control the filetest permission operators
if	use a Perl module if a condition holds (also can no a module)
integer	Use integer arithmetic instead of floating point
less	Request less of something
lib	Manipulate @INC at compile time
locale	Use or avoid POSIX locales for built-in operations
mro	Method Resolution Order
ok	Alternative to Test::More::use_ok
open	Set default PerlIO layers for input and output
ops	Restrict unsafe operations when compiling
overload	Package for overloading Perl operations
overloading	Lexically control overloading
parent	Establish an ISA relationship with base classes at compile time
re	Alter regular expression behaviour
sigtrap	Enable simple signal handling
sort	Control <i>sort()</i> behaviour
strict	Restrict unsafe constructs
subs	Predeclare sub names
threads	Perl interpreter-based threads
threads::shared	Perl extension for sharing data structures between threads
utf8	Enable/disable UTF-8 (or UTF-EBCDIC) in source code
vars	Predeclare global variable names
version	Perl extension for Version Objects
vmsish	Control VMS-specific language features
warnings::register	Warnings import function

Standard Modules

Standard, bundled modules are all expected to behave in a well-defined manner with respect to namespace pollution because they use the Exporter module. See their own documentation for details.

It's possible that not all modules listed below are installed on your system. For example, the GDBM_File module will not be installed if you don't have the gdbm library.

Amiga::ARexx Perl extension for ARexx support
Amiga::Exec Perl extension for low level amiga support
AnyDBM_File Provide framework for multiple DBMs
App::Cpan Easily interact with CPAN from the command line
App::Prove Implements the `prove` command.
App::Prove::State State storage for the `prove` command.
App::Prove::State::Result Individual test suite results.
App::Prove::State::Result::Test Individual test results.
Archive::Tar Module for manipulations of tar archives
Archive::Tar::File A subclass for in-memory extracted file from [Archive::Tar](#)
Attribute::Handlers Simpler definition of attribute handlers
AutoLoader Load subroutines only on demand
AutoSplit Split a package for autoloading
B The Perl Compiler Backend
B::Concise Walk Perl syntax tree, printing concise info about ops
B::Debug Walk Perl syntax tree, printing debug info about ops
B::Deparse Perl compiler backend to produce perl code
B::Op_private OP `op_private` flag definitions
B::Showlex Show lexical variables used in functions or files
B::Terse Walk Perl syntax tree, printing terse info about ops
B::Xref Generates cross reference reports for Perl programs
Benchmark Benchmark running times of Perl code
IO::Socket::IP Family-neutral IP socket supporting both IPv4 and IPv6
Socket Networking constants and support functions
CORE Namespace for Perl's core routines
CPAN Query, download and build perl modules from CPAN sites
CPAN::API::HOWTO A recipe book for programming with CPAN.pm
CPAN::Debug Internal debugging for CPAN.pm
CPAN::Distroprefs Read and match distroprefs

- CPAN::FirstTime
Utility for CPAN::Config file Initialization
- CPAN::HandleConfig
Internal configuration handling for CPAN.pm
- CPAN::Kwalify
Interface between CPAN.pm and Kwalify.pm
- CPAN::Meta The distribution metadata for a CPAN dist
- CPAN::Meta::Converter
Convert CPAN distribution metadata structures
- CPAN::Meta::Feature
An optional feature provided by a CPAN distribution
- CPAN::Meta::History
History of CPAN Meta Spec changes
- CPAN::Meta::History::Meta_1_0
Version 1.0 metadata specification for META.yml
- CPAN::Meta::History::Meta_1_1
Version 1.1 metadata specification for META.yml
- CPAN::Meta::History::Meta_1_2
Version 1.2 metadata specification for META.yml
- CPAN::Meta::History::Meta_1_3
Version 1.3 metadata specification for META.yml
- CPAN::Meta::History::Meta_1_4
Version 1.4 metadata specification for META.yml
- CPAN::Meta::Merge
Merging CPAN Meta fragments
- CPAN::Meta::Prereqs
A set of distribution prerequisites by phase and type
- CPAN::Meta::Requirements
A set of version requirements for a CPAN dist
- CPAN::Meta::Spec
Specification for CPAN distribution metadata
- CPAN::Meta::Validator
Validate CPAN distribution metadata structures
- CPAN::Meta::YAML
Read and write a subset of YAML for CPAN Meta files
- CPAN::Nox Wrapper around CPAN.pm without using any XS module
- CPAN::Plugin
Base class for CPAN shell extensions
- CPAN::Plugin::Specfile
Proof of concept implementation of a trivial [CPAN::Plugin](#)
- CPAN::Queue
Internal queue support for CPAN.pm
- CPAN::Tarzip
Internal handling of tar archives for CPAN.pm

CPAN::Version
 Utility functions to compare CPAN versions

Carp
 Alternative warn and die for modules

Class::Struct
 Declare struct-like datatypes as Perl classes

Compress::Raw::Bzip2
 Low-Level Interface to bzip2 compression library

Compress::Raw::Zlib
 Low-Level Interface to zlib compression library

Compress::Zlib
 Interface to zlib compression library

Config
 Access Perl configuration information

Config::Perl::V
 Structured data retrieval of perl -V output

Cwd
 Get pathname of current working directory

DB
 Programmatic interface to the Perl debugging API

DBM_Filter
 Filter DBM keys/values

DBM_Filter::compress
 Filter for DBM_Filter

DBM_Filter::encode
 Filter for DBM_Filter

DBM_Filter::int32
 Filter for DBM_Filter

DBM_Filter::null
 Filter for DBM_Filter

DBM_Filter::utf8
 Filter for DBM_Filter

DB_File
 Perl5 access to Berkeley DB version 1.x

Data::Dumper
 Stringified perl data structures, suitable for both printing and eval

Devel::PPPort
 Perl/Pollution/Portability

Devel::Peek
 A data debugging tool for the XS programmer

Devel::SelfStubber
 Generate stubs for a SelfLoading module

Digest
 Modules that calculate message digests

Digest::MD5
 Perl interface to the MD5 Algorithm

Digest::SHA
 Perl extension for SHA-1/224/256/384/512

Digest::base
 Digest base class

Digest::file
 Calculate digests of files

DirHandle
 Supply object methods for directory handles

Dumpvalue
 Provides screen dump of Perl data.

DynaLoader
 Dynamically load C libraries into Perl code

Encode Character encodings in Perl

Encode::Alias
 Alias definitions to encodings

Encode::Byte Single Byte Encodings

Encode::CJKConstants
 Internally used by Encode::?:?:ISO_2022_*

Encode::CN China-based Chinese Encodings

Encode::CN::HZ
 Internally used by [Encode::CN](#)

Encode::Config
 Internally used by Encode

Encode::EBCDIC
 EBCDIC Encodings

Encode::Encoder
 Object Oriented Encoder

Encode::Encoding
 Encode Implementation Base Class

Encode::GSM0338
 ESTI GSM 03.38 Encoding

Encode::Guess
 Guesses encoding from data

Encode::JP Japanese Encodings

Encode::JP::H2Z
 Internally used by Encode::JP::2022_JP*

Encode::JP::JIS7
 Internally used by [Encode::JP](#)

Encode::KR Korean Encodings

Encode::KR::2022_KR
 Internally used by [Encode::KR](#)

Encode::MIME::Header
 MIME 'B' and 'Q' header encoding

Encode::MIME::Name
 Internally used by Encode

Encode::PerlIO
 A detailed document on Encode and PerlIO

Encode::Supported
 Encodings supported by Encode

Encode::Symbol
 Symbol Encodings

Encode::TW Taiwan-based Chinese Encodings

Encode::Unicode
 Various Unicode Transformation Formats

Encode::Unicode::UTF7
 UTF-7 encoding

English	Use nice English (or awk) names for ugly punctuation variables
Env	Perl module that imports environment variables as scalars or arrays
Errno	System errno constants
Exporter	Implements default import method for modules
Exporter::Heavy	Exporter guts
ExtUtils::CBuilder	Compile and link C code for Perl modules
ExtUtils::CBuilder::Platform::Windows	Builder class for Windows platforms
ExtUtils::Command	Utilities to replace common UNIX commands in Makefiles etc.
ExtUtils::Command::MM	Commands for the MM's to use in Makefiles
ExtUtils::Constant	Generate XS code to import C header constants
ExtUtils::Constant::Base	Base class for ExtUtils::Constant objects
ExtUtils::Constant::Utils	Helper functions for ExtUtils::Constant
ExtUtils::Constant::XS	Generate C code for XS modules' constants.
ExtUtils::Embed	Utilities for embedding Perl in C/C++ applications
ExtUtils::Install	Install files from here to there
ExtUtils::Installed	Inventory management of installed modules
ExtUtils::Liblist	Determine libraries to use and how to use them
ExtUtils::MM	OS adjusted ExtUtils::MakeMaker subclass
ExtUtils::MM_AIX	AIX specific subclass of ExtUtils::MM_Unix
ExtUtils::MM_Any	Platform-agnostic MM methods
ExtUtils::MM_BeOS	Methods to override UN*X behaviour in ExtUtils::MakeMaker
ExtUtils::MM_Cygwin	Methods to override UN*X behaviour in ExtUtils::MakeMaker
ExtUtils::MM_DOS	DOS specific subclass of ExtUtils::MM_Unix
ExtUtils::MM_Darwin	Special behaviors for OS X

- ExtUtils::MM_MacOS
Once produced Makefiles for MacOS Classic
- ExtUtils::MM_NW5
Methods to override UN*X behaviour in [ExtUtils::MakeMaker](#)
- ExtUtils::MM_OS2
Methods to override UN*X behaviour in [ExtUtils::MakeMaker](#)
- ExtUtils::MM_QNX
QNX specific subclass of [ExtUtils::MM_Unix](#)
- ExtUtils::MM_UWIN
U/WIN specific subclass of [ExtUtils::MM_Unix](#)
- ExtUtils::MM_Unix
Methods used by [ExtUtils::MakeMaker](#)
- ExtUtils::MM_VMS
Methods to override UN*X behaviour in [ExtUtils::MakeMaker](#)
- ExtUtils::MM_VOS
VOS specific subclass of [ExtUtils::MM_Unix](#)
- ExtUtils::MM_Win32
Methods to override UN*X behaviour in [ExtUtils::MakeMaker](#)
- ExtUtils::MM_Win95
Method to customize MakeMaker for Win9X
- ExtUtils::MY
[ExtUtils::MakeMaker](#) subclass for customization
- ExtUtils::MakeMaker
Create a module Makefile
- ExtUtils::MakeMaker::Config
Wrapper around Config.pm
- ExtUtils::MakeMaker::FAQ
Frequently Asked Questions About MakeMaker
- ExtUtils::MakeMaker::Locale
Bundled [Encode::Locale](#)
- ExtUtils::MakeMaker::Tutorial
Writing a module with MakeMaker
- ExtUtils::Manifest
Utilities to write and check a MANIFEST file
- ExtUtils::Miniperl
Write the C code for perlmain.c
- ExtUtils::Mkbootstrap
Make a bootstrap file for use by DynaLoader
- ExtUtils::Mksymlists
Write linker options files for dynamic extension
- ExtUtils::Packlist
Manage .packlist files
- ExtUtils::ParseXS
Converts Perl XS code into C code

ExtUtils::ParseXS::Constants
 Initialization values for some globals

ExtUtils::ParseXS::Eval
 Clean package to evaluate code in

ExtUtils::ParseXS::Utilities
 Subroutines used with [ExtUtils::ParseXS](#)

ExtUtils::Typemaps
 Read/Write/Modify Perl/XS typemap files

ExtUtils::Typemaps::Cmd
 Quick commands for handling typemaps

ExtUtils::Typemaps::InputMap
 Entry in the INPUT section of a typemap

ExtUtils::Typemaps::OutputMap
 Entry in the OUTPUT section of a typemap

ExtUtils::Typemaps::Type
 Entry in the TYPEMAP section of a typemap

ExtUtils::XSSymSet
 Keep sets of symbol names palatable to the VMS linker

ExtUtils::testlib
 Add blib/* directories to @INC

Fatal
 Replace functions with equivalents which succeed or die

Fcntl
 Load the C Fcntl.h defines

File::Basename
 Parse file paths into directory, filename and suffix.

File::Compare
 Compare files or filehandles

File::Copy
 Copy files or filehandles

File::DosGlob
 DOS like globbing and then some

File::Fetch
 A generic file fetching mechanism

File::Find
 Traverse a directory tree.

File::Glob
 Perl extension for BSD glob routine

File::GlobMapper
 Extend File Glob to Allow Input and Output Files

File::Path
 Create or remove directory trees

File::Spec
 Portably perform operations on file names

File::Spec::AmigaOS
[File::Spec](#) for AmigaOS

File::Spec::Cygwin
 Methods for Cygwin file specs

File::Spec::Epic
 Methods for Epic file specs

File::Spec::Functions
 Portably perform operations on file names

File::Spec::Mac
[File::Spec](#) for Mac OS (Classic)

File::Spec::OS2
 Methods for OS/2 file specs

File::Spec::Unix
[File::Spec](#) for Unix, base for other [File::Spec](#) modules

File::Spec::VMS
 Methods for VMS file specs

File::Spec::Win32
 Methods for Win32 file specs

File::Temp Return name and handle of a temporary file safely

File::stat By-name interface to Perl's built-in *stat()* functions

FileCache Keep more files open than the system permits

FileHandle Supply object methods for filehandles

Filter::Simple
 Simplified source filtering

Filter::Util::Call
 Perl Source Filter Utility Module

FindBin Locate directory of original perl script

GDBM_File Perl5 access to the gdbm library.

Getopt::Long Extended processing of command line options

Getopt::Std Process single-character switches with switch clustering

HTTP::Tiny A small, simple, correct HTTP/1.1 client

Hash::Util A selection of general-utility hash subroutines

Hash::Util::FieldHash
 Support for Inside-Out Classes

I18N::Collate Compare 8-bit scalar data according to the current locale

I18N::LangTags
 Functions for dealing with RFC3066-style language tags

I18N::LangTags::Detect
 Detect the user's language preferences

I18N::LangTags::List
 Tags and names for human languages

I18N::Langinfo
 Query locale information

IO Load various IO modules

IO::Compress::Base
 Base Class for IO::Compress modules

IO::Compress::Bzip2
 Write bzip2 files/buffers

IO::Compress::Deflate
Write RFC 1950 files/buffers

IO::Compress::FAQ
Frequently Asked Questions about IO::Compress

IO::Compress::Gzip
Write RFC 1952 files/buffers

IO::Compress::RawDeflate
Write RFC 1951 files/buffers

IO::Compress::Zip
Write zip files/buffers

IO::Dir Supply object methods for directory handles

IO::File Supply object methods for filehandles

IO::Handle Supply object methods for I/O handles

IO::Pipe Supply object methods for pipes

IO::Poll Object interface to system poll call

IO::Seekable Supply seek based methods for I/O objects

IO::Select OO interface to the select system call

IO::Socket Object interface to socket communications

IO::Socket::INET
Object interface for AF_INET domain sockets

IO::Socket::UNIX
Object interface for AF_UNIX domain sockets

IO::Uncompress::AnyInflate
Uncompress zlib-based (zip, gzip) file/buffer

IO::Uncompress::AnyUncompress
Uncompress gzip, zip, bzip2 or lzop file/buffer

IO::Uncompress::Base
Base Class for IO::Uncompress modules

IO::Uncompress::Bunzip2
Read bzip2 files/buffers

IO::Uncompress::Gunzip
Read RFC 1952 files/buffers

IO::Uncompress::Inflate
Read RFC 1950 files/buffers

IO::Uncompress::RawInflate
Read RFC 1951 files/buffers

IO::Uncompress::Unzip
Read zip files/buffers

IO::Zlib IO:: style interface to [Compress::Zlib](#)

IPC::Cmd Finding and running system commands made easy

IPC::Msg SysV Msg IPC object class

IPC::Open2 Open a process for both reading and writing using *open2()*

IPC::Open3 Open a process for reading, writing, and error handling using *open3()*

- IPC::Semaphore
SysV Semaphore IPC object class
- IPC::SharedMem
SysV Shared Memory IPC object class
- IPC::SysV
System V IPC constants and system calls
- JSON::PP
JSON::XS compatible pure-Perl module.
- JSON::PP::Boolean
Dummy module providing [JSON::PP::Boolean](#)
- List::Util
A selection of general-utility list subroutines
- List::Util::XS
Indicate if [List::Util](#) was compiled with a C compiler
- Locale::Codes
A distribution of modules to handle locale codes
- Locale::Codes::API
A description of the callable function in each module
- Locale::Codes::Changes
Details changes to [Locale::Codes](#)
- Locale::Codes::Country
Standard codes for country identification
- Locale::Codes::Currency
Standard codes for currency identification
- Locale::Codes::LangExt
Standard codes for language extension identification
- Locale::Codes::LangFam
Standard codes for language extension identification
- Locale::Codes::LangVar
Standard codes for language variation identification
- Locale::Codes::Language
Standard codes for language identification
- Locale::Codes::Script
Standard codes for script identification
- Locale::Country
Standard codes for country identification
- Locale::Currency
Standard codes for currency identification
- Locale::Language
Standard codes for language identification
- Locale::Maketext
Framework for localization
- Locale::Maketext::Cookbook
Recipes for using [Locale::Maketext](#)
- Locale::Maketext::Guts
Deprecated module to load [Locale::Maketext](#) utf8 code

`Locale::Maketext::GutsLoader`
Deprecated module to load `Locale::Maketext` utf8 code

`Locale::Maketext::Simple`
Simple interface to `Locale::Maketext::Lexicon`

`Locale::Maketext::TPJ13`
Article about software localization

`Locale::Script`
Standard codes for script identification

`MIME::Base64`
Encoding and decoding of base64 strings

`MIME::QuotedPrint`
Encoding and decoding of quoted-printable strings

`Math::BigFloat`
Arbitrary size floating point math package

`Math::BigInt` Arbitrary size integer/float math package

`Math::BigInt::Calc`
Pure Perl module to support `Math::BigInt`

`Math::BigInt::CalcEmu`
Emulate low-level math with `BigInt` code

`Math::BigInt::FastCalc`
`Math::BigInt::Calc` with some XS for more speed

`Math::BigRat`
Arbitrary big rational numbers

`Math::Complex`
Complex numbers and associated mathematical functions

`Math::Trig` Trigonometric functions

`Memoize` Make functions faster by trading space for time

`Memoize::AnyDBM_File`
Glue to provide EXISTS for `AnyDBM_File` for `Storable` use

`Memoize::Expire`
Plug-in module for automatic expiration of memoized values

`Memoize::ExpireFile`
Test for `Memoize` expiration semantics

`Memoize::ExpireTest`
Test for `Memoize` expiration semantics

`Memoize::NDBM_File`
Glue to provide EXISTS for `NDBM_File` for `Storable` use

`Memoize::SDBM_File`
Glue to provide EXISTS for `SDBM_File` for `Storable` use

`Memoize::Storable`
Store Memoized data in `Storable` database

`Module::CoreList`
What modules shipped with versions of perl

Module::CoreList::Utils
 What utilities shipped with versions of perl

Module::Load
 Runtime require of both modules and files

Module::Load::Conditional
 Looking up module information / loading at runtime

Module::Loaded
 Mark modules as loaded or unloaded

Module::Metadata
 Gather package and POD information from perl module files

NDBM_File Tied access to ndbm files

NEXT Provide a pseudo-class NEXT (et al) that allows method redispatch

Net::Cmd Network Command class (as used by FTP, SMTP etc)

Net::Config Local configuration data for libnet

Net::Domain Attempt to evaluate the current host's internet name and domain

Net::FTP FTP Client class

Net::FTP::dataconn
 FTP Client data connection class

Net::NNTP NNTP Client class

Net::Netrc OO interface to users netrc file

Net::POP3 Post Office Protocol 3 Client class (RFC1939)

Net::Ping Check a remote host for reachability

Net::SMTP Simple Mail Transfer Protocol Client

Net::Time Time and daytime network client interface

Net::hostent By-name interface to Perl's built-in gethost*() functions

Net::libnetFAQ
 Libnet Frequently Asked Questions

Net::netent By-name interface to Perl's built-in getnet*() functions

Net::protoent By-name interface to Perl's built-in getproto*() functions

Net::servent By-name interface to Perl's built-in getserv*() functions

O Generic interface to Perl Compiler backends

ODBM_File Tied access to odbm files

Opcodes Disable named opcodes when compiling perl code

POSIX Perl interface to IEEE Std 1003.1

Params::Check
 A generic input parsing/checking mechanism.

Parse::CPAN::Meta
 Parse META.yml and META.json CPAN metadata files

Perl::OSType Map Perl operating system names to generic types

PerlIO On demand loader for PerlIO layers and root of PerlIO::* name space

PerlIO::encoding
 Encoding layer

PerlIO::mmap
 Memory mapped IO

PerlIO::scalar
 In-memory IO, scalar IO

PerlIO::via
 Helper class for PerlIO layers implemented in perl

PerlIO::via::QuotedPrint
 PerlIO layer for quoted-printable strings

Pod::Escapes
 For resolving Pod E<...> sequences

Pod::Find
 Find POD documents in directory trees

Pod::Functions
 Group Perl's functions a la perlfunc.pod

Pod::Html
 Module to convert pod files to HTML

Pod::InputObjects
 Objects representing POD input paragraphs, commands, etc.

Pod::Man
 Convert POD data to formatted *roff input

Pod::ParseLink
 Parse an L<> formatting code in POD text

Pod::ParseUtils
 Helpers for POD parsing and conversion

Pod::Parser
 Base class for creating POD filters and translators

Pod::Perldoc
 Look up Perl documentation in Pod format.

Pod::Perldoc::BaseTo
 Base for [Pod::Perldoc](#) formatters

Pod::Perldoc::GetOptsOO
 Customized option parser for [Pod::Perldoc](#)

Pod::Perldoc::ToANSI
 Render Pod with ANSI color escapes

Pod::Perldoc::ToChecker
 Let Perldoc check Pod for errors

Pod::Perldoc::ToMan
 Let Perldoc render Pod as man pages

Pod::Perldoc::ToNroff
 Let Perldoc convert Pod to nroff

Pod::Perldoc::ToPod
 Let Perldoc render Pod as ... Pod!

Pod::Perldoc::ToRtf
 Let Perldoc render Pod as RTF

Pod::Perldoc::ToTerm
 Render Pod with terminal escapes

Pod::Perldoc::ToText
 Let Perldoc render Pod as plaintext

`Pod::Perldoc::ToTk`
Let Perldoc use `Tk::Pod` to render Pod

`Pod::Perldoc::ToXml`
Let Perldoc render Pod as XML

`Pod::PlainText`
Convert POD data to formatted ASCII text

`Pod::Select` Extract selected sections of POD from input

`Pod::Simple` Framework for parsing Pod

`Pod::Simple::Checker`
Check the Pod syntax of a document

`Pod::Simple::Debug`
Put [Pod::Simple](#) into trace/debug mode

`Pod::Simple::DumpAsText`
Dump Pod-parsing events as text

`Pod::Simple::DumpAsXML`
Turn Pod into XML

`Pod::Simple::HTML`
Convert Pod to HTML

`Pod::Simple::HTMLBatch`
Convert several Pod files to several HTML files

`Pod::Simple::LinkSection`
Represent “section” attributes of L codes

`Pod::Simple::Methody`
Turn [Pod::Simple](#) events into method calls

`Pod::Simple::PullParser`
A pull-parser interface to parsing Pod

`Pod::Simple::PullParserEndToken`
End-tokens from [Pod::Simple::PullParser](#)

`Pod::Simple::PullParserStartToken`
Start-tokens from [Pod::Simple::PullParser](#)

`Pod::Simple::PullParserTextToken`
Text-tokens from [Pod::Simple::PullParser](#)

`Pod::Simple::PullParserToken`
Tokens from [Pod::Simple::PullParser](#)

`Pod::Simple::RTF`
Format Pod as RTF

`Pod::Simple::Search`
Find POD documents in directory trees

`Pod::Simple::SimpleTree`
Parse Pod into a simple parse tree

`Pod::Simple::Subclassing`
Write a formatter as a [Pod::Simple](#) subclass

`Pod::Simple::Text`
Format Pod as plaintext

Pod::Simple::TextContent
Get the text content of Pod

Pod::Simple::XHTML
Format Pod as validating XHTML

Pod::Simple::XMLOutputStream
Turn Pod into XML

Pod::Text
Convert POD data to formatted text

Pod::Text::Color
Convert POD data to formatted color ASCII text

Pod::Text::Termcap
Convert POD data to ASCII text with format escapes

Pod::Usage
Print a usage message from embedded pod documentation

SDBM_File
Tied access to sdbm files

Safe
Compile and execute code in restricted compartments

Scalar::Util
A selection of general-utility scalar subroutines

Search::Dict
Look - search for key in dictionary file

SelectSaver
Save and restore selected file handle

SelfLoader
Load functions only on demand

Storable
Persistence for Perl data structures

Sub::Util
A selection of utility subroutines for subs and CODE references

Symbol
Manipulate Perl symbols and their names

Sys::Hostname
Try every conceivable way to get hostname

Sys::Syslog
Perl interface to the UNIX *syslog(3)* calls

Sys::Syslog::Win32
Win32 support for [Sys::Syslog](#)

TAP::Base
Base class that provides common functionality to [TAP::Parser](#)

TAP::Formatter::Base
Base class for harness output delegates

TAP::Formatter::Color
Run Perl test scripts with color

TAP::Formatter::Console
Harness output delegate for default console output

TAP::Formatter::Console::ParallelSession
Harness output delegate for parallel console output

TAP::Formatter::Console::Session
Harness output delegate for default console output

TAP::Formatter::File
Harness output delegate for file output

TAP::Formatter::File::Session
Harness output delegate for file output

TAP::Formatter::Session
Abstract base class for harness output delegate

TAP::Harness
Run test scripts with statistics

TAP::Harness::Env
Parsing harness related environmental variables where appropriate

TAP::Object
Base class that provides common functionality to all TAP::* modules

TAP::Parser
Parse TAP output

TAP::Parser::Aggregator
Aggregate [TAP::Parser](#) results

TAP::Parser::Grammar
A grammar for the Test Anything Protocol.

TAP::Parser::Iterator
Base class for TAP source iterators

TAP::Parser::Iterator::Array
Iterator for array-based TAP sources

TAP::Parser::Iterator::Process
Iterator for process-based TAP sources

TAP::Parser::Iterator::Stream
Iterator for filehandle-based TAP sources

TAP::Parser::IteratorFactory
Figures out which SourceHandler objects to use for a given Source

TAP::Parser::Multiplexer
Multiplex multiple TAP::Parsers

TAP::Parser::Result
Base class for [TAP::Parser](#) output objects

TAP::Parser::Result::Bailout
Bailout result token.

TAP::Parser::Result::Comment
Comment result token.

TAP::Parser::Result::Plan
Plan result token.

TAP::Parser::Result::Pragma
TAP pragma token.

TAP::Parser::Result::Test
Test result token.

TAP::Parser::Result::Unknown
Unknown result token.

TAP::Parser::Result::Version
TAP syntax version token.

TAP::Parser::Result::YAML
YAML result token.

TAP::Parser::ResultFactory
Factory for creating [TAP::Parser](#) output objects

TAP::Parser::Scheduler
Schedule tests during parallel testing

- TAP::Parser::Scheduler::Job
A single testing job.
- TAP::Parser::Scheduler::Spinner
A no-op job.
- TAP::Parser::Source
A TAP source & meta data about it
- TAP::Parser::SourceHandler
Base class for different TAP source handlers
- TAP::Parser::SourceHandler::Executable
Stream output from an executable TAP source
- TAP::Parser::SourceHandler::File
Stream TAP from a text file.
- TAP::Parser::SourceHandler::Handle
Stream TAP from an [IO::Handle](#) or a GLOB.
- TAP::Parser::SourceHandler::Perl
Stream TAP from a Perl executable
- TAP::Parser::SourceHandler::RawTAP
Stream output from raw TAP in a scalar/array ref.
- TAP::Parser::YAMLish::Reader
Read YAMLish data from iterator
- TAP::Parser::YAMLish::Writer
Write YAMLish data
- Term::ANSIColor
Color screen output using ANSI escape sequences
- Term::Cap Perl termcap interface
- Term::Complete
Perl word completion module
- Term::ReadLine
Perl interface to various `readline` packages.
- Test Provides a simple framework for writing test scripts
- Test::Builder Backend for building test libraries
- Test::Builder::IO::Scalar
A copy of [IO::Scalar](#) for [Test::Builder](#)
- Test::Builder::Module
Base class for test modules
- Test::Builder::Tester
Test testsuites that have been built with
- Test::Builder::Tester::Color
Turn on colour in [Test::Builder::Tester](#)
- Test::Harness Run Perl standard test scripts with statistics
- Test::Harness::Beyond
Beyond make test
- Test::More Yet another framework for writing test scripts
- Test::Simple Basic utilities for writing tests.

Test::Tester Ease testing test modules built with [Test::Builder](#)
Test::Tester::Capture
 Help testing test modules built with [Test::Builder](#)
Test::Tester::CaptureRunner
 Help testing test modules built with [Test::Builder](#)
Test::Tutorial A tutorial about writing really basic tests
Test::use::ok Alternative to [Test::More::use_ok](#)
Text::Abbrev Abbrev - create an abbreviation table from a list
Text::Balanced
 Extract delimited text sequences from strings.
Text::ParseWords
 Parse text into an array of tokens or array of arrays
Text::Tabs Expand and unexpand tabs like unix [expand\(1\)](#) and [unexpand\(1\)](#)
Text::Wrap Line wrapping to form simple paragraphs
Thread Manipulate threads in Perl (for old code only)
Thread::Queue
 Thread-safe queues
Thread::Semaphore
 Thread-safe semaphores
Tie::Array Base class for tied arrays
Tie::File Access the lines of a disk file via a Perl array
Tie::Handle Base class definitions for tied handles
Tie::Hash Base class definitions for tied hashes
Tie::Hash::NamedCapture
 Named regexp capture buffers
Tie::Memoize
 Add data to hash when needed
Tie::RefHash Use references as hash keys
Tie::Scalar Base class definitions for tied scalars
Tie::StdHandle
 Base class definitions for tied handles
Tie::SubstrHash
 Fixed-table-size, fixed-key-length hashing
Time::HiRes High resolution alarm, sleep, gettimeofday, interval timers
Time::Local Efficiently compute time from local and GMT time
Time::Piece Object Oriented time objects
Time::Seconds
 A simple API to convert seconds to other date values
Time::gmtime
 By-name interface to Perl's built-in [gmtime\(\)](#) function
Time::localtime
 By-name interface to Perl's built-in [localtime\(\)](#) function

Time::tm Internal object used by [Time::gmtime](#) and [Time::localtime](#)

UNIVERSAL Base class for ALL classes (blessed references)

Unicode::Collate
Unicode Collation Algorithm

Unicode::Collate::CJK::Big5
Weighting CJK Unified Ideographs

Unicode::Collate::CJK::GB2312
Weighting CJK Unified Ideographs

Unicode::Collate::CJK::JISX0208
Weighting JIS KANJI for [Unicode::Collate](#)

Unicode::Collate::CJK::Korean
Weighting CJK Unified Ideographs

Unicode::Collate::CJK::Pinyin
Weighting CJK Unified Ideographs

Unicode::Collate::CJK::Stroke
Weighting CJK Unified Ideographs

Unicode::Collate::CJK::Zhuyin
Weighting CJK Unified Ideographs

Unicode::Collate::Locale
Linguistic tailoring for DUCET via [Unicode::Collate](#)

Unicode::Normalize
Unicode Normalization Forms

Unicode::UCD
Unicode character database

User::grent By-name interface to Perl's built-in `getgr*`() functions

User::pwent By-name interface to Perl's built-in `getpw*`() functions

VMS::DCLsym
Perl extension to manipulate DCL symbols

VMS::Filespec
Convert between VMS and Unix file specification syntax

VMS::Stdio Standard I/O functions via VMS extensions

Win32 Interfaces to some Win32 API Functions

Win32API::File
Low-level access to Win32 system API calls for files/dirs.

Win32CORE Win32 CORE function stubs

XS::APItest Test the perl C API

XS::Typemap
Module to test the XS typemaps distributed with perl

XSLoader Dynamically load C libraries into Perl code

autodie::Scope::Guard
Wrapper class for calling subs at end of scope

autodie::Scope::GuardStack
Hook stack for managing scopes via `%^H`

autodie::Util Internal Utility subroutines for autodie and Fatal

version::Internals

Perl extension for Version Objects

To find out *all* modules installed on your system, including those without documentation or outside the standard release, just use the following command (under the default win32 shell, double quotes should be used instead of single quotes).

```
% perl -MFile::Find=find -MFile::Spec::Functions -Tlwe \
'find { wanted => sub { print canonpath $_ if /\.pm\z/ } ,
no_chdir => 1 } , @INC'
```

(The `-T` is here to prevent `'.` from being listed in `@INC`.) They should all have their own documentation installed and accessible via your system *man(1)* command. If you do not have a **find** program, you can use the Perl **find2perl** program instead, which generates Perl code as output you can run through perl. If you have a **man** program but it doesn't find your modules, you'll have to fix your manpath. See perl for details. If you have no system **man** command, you might try the **perldoc(1)** program.

Note also that the command **perldoc(1)** `perllocalgi` ves you a (possibly incomplete) list of the modules that have been further installed on your system. (The `perllocal.pod` file is updated by the standard MakeMaker install process.)

Extension Modules

Extension modules are written in C (or a mix of Perl and C). They are usually dynamically loaded into Perl if and when you need them, but may also be linked in statically. Supported extension modules include Socket, Fcntl, and POSIX.

Many popular C extension modules do not come bundled (at least, not completely) due to their sizes, volatility, or simply lack of time for adequate testing and configuration across the multitude of platforms on which Perl was beta-tested. You are encouraged to look for them on CPAN (described below), or using web search engines like Alta Vista or Google.

CPAN

CPAN stands for Comprehensive Perl Archive Network; it's a globally replicated trove of Perl materials, including documentation, style guides, tricks and traps, alternate ports to non-Unix systems and occasional binary distributions for these. Search engines for CPAN can be found at <http://www.cpan.org/>

Most importantly, CPAN includes around a thousand unbundled modules, some of which require a C compiler to build. Major categories of modules are:

- Language Extensions and Documentation Tools
- Development Support
- Operating System Interfaces
- Networking, Device Control (modems) and InterProcess Communication
- Data Types and Data Type Utilities
- Database Interfaces
- User Interfaces
- Interfaces to / Emulations of Other Programming Languages
- File Names, File Systems and File Locking (see also File Handles)
- String Processing, Language Text Processing, Parsing, and Searching
- Option, Argument, Parameter, and Configuration File Processing
- Internationalization and Locale
- Authentication, Security, and Encryption

- World Wide Web, HTML, HTTP, CGI, MIME
- Server and Daemon Utilities
- Archiving and Compression
- Images, Pixmap and Bitmap Manipulation, Drawing, and Graphing
- Mail and Usenet News
- Control Flow Utilities (callbacks and exceptions etc)
- File Handle and Input/Output Stream Utilities
- Miscellaneous Modules

The list of the registered CPAN sites follows. Please note that the sorting order is alphabetical on fields:

Continent | |-->Country | |-->[state/province] | |-->ftp | |-->[http]

and thus the North American servers happen to be listed between the European and the South American sites.

Registered CPAN sites

Africa

South Africa

<http://cpan.mirror.ac.za/>
<ftp://cpan.mirror.ac.za/>
<http://mirror.is.co.za/pub/cpan/>
<ftp://ftp.is.co.za/pub/cpan/>
<ftp://ftp.saix.net/pub/CPAN/>

Asia

China

<http://cpan.wenzk.com/>

Hong Kong

<http://ftp.cuhk.edu.hk/pub/packages/perl/CPAN/>
<ftp://ftp.cuhk.edu.hk/pub/packages/perl/CPAN/>
<http://mirrors.geoexpat.com/cpan/>

India

<http://perlmirror.indialinks.com/>

Indonesia

<http://cpan.biz.net.id/>
<http://komo.vlsm.org/CPAN/>
<ftp://komo.vlsm.org/CPAN/>
<http://cpan.cermin.lipi.go.id/>
<ftp://cermin.lipi.go.id/pub/CPAN/>
<http://cpan.pesat.net.id/>

Japan

<ftp://ftp.u-aizu.ac.jp/pub/CPAN>
<ftp://ftp.kddilabs.jp/CPAN/>
<http://ftp.nara.wide.ad.jp/pub/CPAN/>
<ftp://ftp.nara.wide.ad.jp/pub/CPAN/>
<http://ftp.jaist.ac.jp/pub/CPAN/>
<ftp://ftp.jaist.ac.jp/pub/CPAN/>
<ftp://ftp.dti.ad.jp/pub/lang/CPAN/>
<ftp://ftp.ring.gr.jp/pub/lang/perl/CPAN/>
<http://ftp.riken.jp/lang/CPAN/>
<ftp://ftp.riken.jp/lang/CPAN/>
<http://ftp.yz.yamagata-u.ac.jp/pub/lang/cpan/>
<ftp://ftp.yz.yamagata-u.ac.jp/pub/lang/cpan/>

Republic of Korea

<http://ftp.kaist.ac.kr/pub/CPAN>
<ftp://ftp.kaist.ac.kr/pub/CPAN>
<http://cpan.mirror.cdnetworks.com/>
<ftp://cpan.mirror.cdnetworks.com/CPAN/>
<http://cpan.sarang.net/>
<ftp://cpan.sarang.net/CPAN/>

Russia

<http://cpan.tomsk.ru/>
<ftp://cpan.tomsk.ru/>

Singapore

<http://mirror.averse.net/pub/CPAN>
<ftp://mirror.averse.net/pub/CPAN>
<http://cpan.mirror.choon.net/>
<http://cpan.oss.eznetsols.org>
<ftp://ftp.oss.eznetsols.org/cpan>

Taiwan

<http://ftp.cse.yzu.edu.tw/pub/CPAN/>
<ftp://ftp.cse.yzu.edu.tw/pub/CPAN/>
<http://cpan.nctu.edu.tw/>
<ftp://cpan.nctu.edu.tw/>
<ftp://ftp.ncu.edu.tw/CPAN/>
<http://cpan.cdpa.nsysu.edu.tw/>
<ftp://cpan.cdpa.nsysu.edu.tw/Unix/Lang/CPAN/>
<http://cpan.stu.edu.tw>
<ftp://ftp.stu.edu.tw/CPAN>
<http://ftp.stu.edu.tw/CPAN>
<ftp://ftp.stu.edu.tw/pub/CPAN>
<http://cpan.cs.pu.edu.tw/>
<ftp://cpan.cs.pu.edu.tw/pub/CPAN>

Thailand

<http://mirrors.issp.co.th/cpan/>
<ftp://mirrors.issp.co.th/cpan/>
<http://mirror.yourconnect.com/CPAN/>
<ftp://mirror.yourconnect.com/CPAN/>

Turkey

<http://cpan.gazi.edu.tr/>

Central America

Costa Rica

<http://mirrors.ucr.ac.cr/CPAN/>
<ftp://mirrors.ucr.ac.cr/CPAN/>

Europe**Austria**

<http://cpan.inode.at/>
<ftp://cpan.inode.at>
<http://gd.tuwien.ac.at/languages/perl/CPAN/>
<ftp://gd.tuwien.ac.at/pub/CPAN/>

Belgium

<http://ftp.belnet.be/mirror/ftp.cpan.org/>
<ftp://ftp.belnet.be/mirror/ftp.cpan.org/>
<http://ftp.easynet.be/pub/CPAN/>
<http://cpan.weepee.org/>

Bosnia and Herzegovina

<http://cpan.blic.net/>

Bulgaria

<http://cpan.cbox.biz/>
<ftp://cpan.cbox.biz/cpan/>
<http://cpan.digsys.bg/>
<ftp://ftp.digsys.bg/pub/CPAN>

Croatia

<http://ftp.carnet.hr/pub/CPAN/>
<ftp://ftp.carnet.hr/pub/CPAN/>

Czech Republic

<ftp://ftp.fi.muni.cz/pub/CPAN/>
<http://archive.cpan.cz/>

Denmark

<http://mirrors.dotsrc.org/cpan>
<ftp://mirrors.dotsrc.org/cpan/>
<http://www.cpan.dk/>
<http://mirror.uni-c.dk/pub/CPAN/>

Finland

<ftp://ftp.funet.fi/pub/languages/perl/CPAN/>
<http://mirror.eunet.fi/CPAN>

France

<http://cpan.enstimac.fr/>
<ftp://ftp.inria.fr/pub/CPAN/>
<http://distrib-coffee.ipsl.jussieu.fr/pub/mirrors/cpan/>
<ftp://distrib-coffee.ipsl.jussieu.fr/pub/mirrors/cpan/>
<ftp://ftp.lip6.fr/pub/perl/CPAN/>
<http://mir2.ovh.net/ftp.cpan.org>
<ftp://mir1.ovh.net/ftp.cpan.org>
<ftp://ftp.oleane.net/pub/CPAN/>
<http://ftp.crihan.fr/mirrors/ftp.cpan.org/>
<ftp://ftp.crihan.fr/mirrors/ftp.cpan.org/>
<http://ftp.u-strasbg.fr/CPAN>
<ftp://ftp.u-strasbg.fr/CPAN>
<http://cpan.cict.fr/>
<ftp://cpan.cict.fr/pub/CPAN/>

Germany

<ftp://ftp.fu-berlin.de/unix/languages/perl/>
<http://mirrors.softliste.de/cpan/>
<ftp://ftp.rub.de/pub/CPAN/>
<http://www.planet-elektronik.de/CPAN/>
<http://ftp.hosteurope.de/pub/CPAN/>
<ftp://ftp.hosteurope.de/pub/CPAN/>
<http://www.mirror-space.org/cpan/>
<http://mirror.netcologne.de/cpan/>
<ftp://mirror.netcologne.de/cpan/>
<ftp://ftp.freenet.de/pub/ftp.cpan.org/pub/CPAN/>
<http://ftp-stud.hs-esslingen.de/pub/Mirrors/CPAN/>
<ftp://ftp-stud.hs-esslingen.de/pub/Mirrors/CPAN/>
<http://mirrors.zerg.biz/cpan/>
<http://ftp.gwdg.de/pub/languages/perl/CPAN/>
<ftp://ftp.gwdg.de/pub/languages/perl/CPAN/>
<http://dl.ambiweb.de/mirrors/ftp.cpan.org/>
<http://cpan.mirror.clusters.kg/>
<http://cpan.mirror.iphh.net/>
<ftp://cpan.mirror.iphh.net/pub/CPAN/>
<http://cpan.mirroring.de/>
<http://mirror.informatik.uni-mannheim.de/pub/mirrors/CPAN/>
<ftp://mirror.informatik.uni-mannheim.de/pub/mirrors/CPAN/>
<http://www.chemmedia.de/mirrors/CPAN/>
<http://ftp.cw.net/pub/CPAN/>
<ftp://ftp.cw.net/pub/CPAN/>
<http://cpan.cpantesters.org/>
<ftp://cpan.cpantesters.org/CPAN/>
<http://cpan.mirrored.de/>
<ftp://mirror.petamem.com/CPAN/>
<http://cpan.noris.de/>
<ftp://cpan.noris.de/pub/CPAN/>
<ftp://ftp.mpi-sb.mpg.de/pub/perl/CPAN/>
<ftp://ftp.gmd.de/mirrors/CPAN/>

Greece

<ftp://ftp.forthnet.gr/pub/languages/perl/CPAN>
<ftp://ftp.ntua.gr/pub/lang/perl/>
<http://cpan.cc.uoc.gr/>
<ftp://ftp.cc.uoc.gr/mirrors/CPAN/>

Hungary

<http://cpan.mirrors.enexis.hu/>
<ftp://cpan.mirrors.enexis.hu/mirrors/cpan/>
<http://cpan.hu/>

Iceland

<http://ftp.rhnet.is/pub/CPAN/>
<ftp://ftp.rhnet.is/pub/CPAN/>

Ireland

<http://ftp.esat.net/pub/languages/perl/CPAN/>
<ftp://ftp.esat.net/pub/languages/perl/CPAN/>
<http://ftp.heanet.ie/mirrors/ftp.perl.org/pub/CPAN>
<ftp://ftp.heanet.ie/mirrors/ftp.perl.org/pub/CPAN>

Italy

<http://bo.mirror.garr.it/mirrors/CPAN/>
<http://cpan.panu.it/>
<ftp://ftp.panu.it/pub/mirrors/perl/CPAN/>

Latvia

<http://kvin.lv/pub/CPAN/>

Lithuania

<http://ftp.litnet.lt/pub/CPAN/>
<ftp://ftp.litnet.lt/pub/CPAN/>

Malta

<http://cpan.waldonet.net.mt/>

Netherlands

<ftp://ftp.quicknet.nl/pub/CPAN/>
<http://mirror.hostfuss.com/CPAN/>
<ftp://mirror.hostfuss.com/CPAN/>
<http://mirrors3.kernel.org/cpan/>
<ftp://mirrors3.kernel.org/pub/CPAN/>
<http://cpan.mirror.versatel.nl/>
<ftp://ftp.mirror.versatel.nl/cpan/>
<ftp://download.xs4all.nl/pub/mirror/CPAN/>
<http://mirror.leaseweb.com/CPAN/>
<ftp://mirror.leaseweb.com/CPAN/>
<ftp://ftp.cpan.nl/pub/CPAN/>
<http://archive.cs.uu.nl/mirror/CPAN/>
<ftp://ftp.cs.uu.nl/mirror/CPAN/>
<http://luxitude.net/cpan/>

Norway

<ftp://ftp.uninett.no/pub/languages/perl/CPAN/>
<ftp://ftp.uit.no/pub/languages/perl/cpan/>

Poland

<http://piotrkosoft.net/pub/mirrors/CPAN/>
<ftp://ftp.piotrkosoft.net/pub/mirrors/CPAN/>
<http://ftp.man.poznan.pl/pub/CPAN/>
<ftp://ftp.man.poznan.pl/pub/CPAN/>
<ftp://ftp.ps.pl/pub/CPAN/>
<ftp://sunsite.icm.edu.pl/pub/CPAN/>
<ftp://ftp.tpnet.pl/d4/CPAN/>

Portugal

<http://cpan.dei.uc.pt/>
<ftp://ftp.dei.uc.pt/pub/CPAN/>
<ftp://ftp.ist.utl.pt/pub/CPAN/>
<http://cpan.perl.pt/>
<http://cpan.ip.pt/>
<ftp://cpan.ip.pt/pub/cpan/>
<http://mirrors.nfsi.pt/CPAN/>
<ftp://mirrors.nfsi.pt/pub/CPAN/>
<http://cpan.dcc.fc.up.pt/>

Romania

<http://ftp.astral.ro/pub/CPAN/>
ftp://ftp.astral.ro/pub/CPAN/
ftp://ftp.lug.ro/CPAN
<http://mirrors.xservers.ro/CPAN/>
<http://mirrors.hostingromania.ro/ftp.cpan.org/>
ftp://ftp.hostingromania.ro/mirrors/ftp.cpan.org/
ftp://ftp.iasi.roedu.net/pub/mirrors/ftp.cpan.org/

Russia

ftp://ftp.aha.ru/CPAN/
<http://cpan.rinet.ru/>
ftp://cpan.rinet.ru/pub/mirror/CPAN/
ftp://ftp.SpringDaemons.com/pub/CPAN/
<http://mirror.rol.ru/CPAN/>
<http://ftp.silvernet.ru/CPAN/>
<http://ftp.spbu.ru/CPAN/>
ftp://ftp.spbu.ru/CPAN/

Slovakia

<http://cpan.fyxm.net/>

Slovenia

<http://www.klevze.si/cpan>

Spain

<http://osl.ugr.es/CPAN/>
ftp://ftp.rediris.es/mirror/CPAN/
<http://ftp.gui.uva.es/sites/cpan.org/>
ftp://ftp.gui.uva.es/sites/cpan.org/

Sweden

<http://mirrors4.kernel.org/cpan/>
ftp://mirrors4.kernel.org/pub/CPAN/

Switzerland

<http://cpan.mirror.solnet.ch/>
ftp://ftp.solnet.ch/mirror/CPAN/
ftp://ftp.adwired.ch/CPAN/
<http://mirror.switch.ch/ftp/mirror/CPAN/>
ftp://mirror.switch.ch/mirror/CPAN/

Ukraine

<http://cpan.makeperl.org/>
ftp://cpan.makeperl.org/pub/CPAN
<http://cpan.org.ua/>
<http://cpan.gafol.net/>
ftp://ftp.gafol.net/pub/cpan/

United Kingdom

<http://www.mirror-service.org/sites/ftp.funet.fi/pub/languages/perl/CPAN/>
<ftp://ftp.mirror-service.org/sites/ftp.funet.fi/pub/languages/perl/CPAN/>
<http://mirror.tje.me.uk/pub/mirrors/ftp.cpan.org/>
<ftp://mirror.tje.me.uk/pub/mirrors/ftp.cpan.org/>
<http://www.mirror.8086.net/sites/CPAN/>
<ftp://ftp.mirror.8086.net/sites/CPAN/>
<http://cpan.mirror.anlx.net/>
<ftp://ftp.mirror.anlx.net/CPAN/>
<http://mirror.bytemark.co.uk/CPAN/>
<ftp://mirror.bytemark.co.uk/CPAN/>
<http://cpan.etla.org/>
<ftp://cpan.etla.org/pub/CPAN>
<ftp://ftp.demon.co.uk/pub/CPAN/>
<http://mirror.sov.uk.goscomb.net/CPAN/>
<ftp://mirror.sov.uk.goscomb.net/pub/CPAN/>
<http://ftp.plig.net/pub/CPAN/>
<ftp://ftp.plig.net/pub/CPAN/>
<http://ftp.ticklers.org/pub/CPAN/>
<ftp://ftp.ticklers.org/pub/CPAN/>
<http://cpan.mirrors.uk2.net/>
<ftp://mirrors.uk2.net/pub/CPAN/>
<http://mirror.ox.ac.uk/sites/www.cpan.org/>
<ftp://mirror.ox.ac.uk/sites/www.cpan.org/>

North America

Bahamas

<http://www.securehost.com/mirror/CPAN/>

Canada

<http://cpan.arcticnetwork.ca>
<ftp://mirror.arcticnetwork.ca/pub/CPAN>
<http://cpan.sunsite.ualberta.ca/>
<ftp://cpan.sunsite.ualberta.ca/pub/CPAN/>
<http://theoryx5.uwinnipeg.ca/pub/CPAN/>
<ftp://theoryx5.uwinnipeg.ca/pub/CPAN/>
<http://arwen.cs.dal.ca/mirror/CPAN/>
<ftp://arwen.cs.dal.ca/pub/mirror/CPAN/>
<http://CPAN.mirror.rafal.ca/>
<ftp://CPAN.mirror.rafal.ca/pub/CPAN/>
<ftp://ftp.nrc.ca/pub/CPAN/>
<http://mirror.csclub.uwaterloo.ca/pub/CPAN/>
<ftp://mirror.csclub.uwaterloo.ca/pub/CPAN/>

Mexico

<http://www.msg.com.mx/CPAN/>
<ftp://ftp.msg.com.mx/pub/CPAN/>

United States

Alabama

<http://mirror.hiwaay.net/CPAN/>
<ftp://mirror.hiwaay.net/CPAN/>

Arizona

<http://cpan.ezarticleinformation.com/>

California

<http://cpan.knowledgematters.net/>
<http://cpan.binkerton.com/>
<http://cpan.develooper.com/>
<http://mirrors.gossamer-threads.com/CPAN>
<http://cpan.schatt.com/>
<http://mirrors.kernel.org/cpan/>
<ftp://mirrors.kernel.org/pub/CPAN>
<http://mirrors2.kernel.org/cpan/>
<ftp://mirrors2.kernel.org/pub/CPAN/>
<http://cpan.mirror.facebook.net/>
<http://mirrors1.kernel.org/cpan/>
<ftp://mirrors1.kernel.org/pub/CPAN/>
<http://cpan-sj.viaverio.com/>
<ftp://cpan-sj.viaverio.com/pub/CPAN/>
<http://www.perl.com/CPAN/>

Florida

<ftp://ftp.cise.ufl.edu/pub/mirrors/CPAN/>
<http://mirror.atlantic.net/pub/CPAN/>
<ftp://mirror.atlantic.net/pub/CPAN/>

Idaho

<http://mirror.its.uidaho.edu/pub/cpan/>
<ftp://mirror.its.uidaho.edu/cpan/>

Illinois

<http://cpan.mirrors.hoobly.com/>
<http://cpan.uchicago.edu/pub/CPAN/>
<ftp://cpan.uchicago.edu/pub/CPAN/>
<http://mirrors.servercentral.net/CPAN/>
<http://www.stathy.com/CPAN/>
<ftp://www.stathy.com/CPAN/>

Indiana

<ftp://ftp.uwsg.iu.edu/pub/perl/CPAN/>
<http://cpan.netnitco.net/>
<ftp://cpan.netnitco.net/pub/mirrors/CPAN/>
<http://ftp.ndlug.nd.edu/pub/perl/>
<ftp://ftp.ndlug.nd.edu/pub/perl/>

Massachusetts

<http://mirrors.ccs.neu.edu/CPAN/>

Michigan

<http://ftp.wayne.edu/cpan/>
<ftp://ftp.wayne.edu/cpan/>

Minnesota

<http://cpan.msi.umn.edu/>

New Jersey

<http://mirror.datapipe.net/CPAN/>
<ftp://mirror.datapipe.net/pub/CPAN/>

New York

<http://mirrors.24-7-solutions.net/pub/CPAN/>
ftp://mirrors.24-7-solutions.net/pub/CPAN/
<http://mirror.cc.columbia.edu/pub/software/cpan/>
ftp://mirror.cc.columbia.edu/pub/software/cpan/
<http://cpan.belfry.net/>
<http://cpan.erlbaum.net/>
ftp://cpan.erlbaum.net/CPAN/
<http://cpan.hexten.net/>
ftp://cpan.hexten.net/
ftp://mirror.nyi.net/CPAN/
<http://mirror.rit.edu/CPAN/>
ftp://mirror.rit.edu/CPAN/

North Carolina

<http://www.ibiblio.org/pub/mirrors/CPAN>
ftp://ftp.ncsu.edu/pub/mirror/CPAN/

Oregon

<http://ftp.osuosl.org/pub/CPAN/>
ftp://ftp.osuosl.org/pub/CPAN/

Pennsylvania

<http://ftp.epix.net/CPAN/>
ftp://ftp.epix.net/pub/languages/perl/
<http://cpan.pair.com/>
ftp://cpan.pair.com/pub/CPAN/

South Carolina

<http://cpan.mirror.clemson.edu/>

Tennessee

<http://mira.sunsite.utk.edu/CPAN/>

Texas

<http://mirror.uta.edu/CPAN>

Utah

ftp://mirror.xmission.com/CPAN/

Virginia

<http://cpan-du.viaverio.com/>
ftp://cpan-du.viaverio.com/pub/CPAN/
<http://perl.secsup.org/>
ftp://perl.secsup.org/pub/perl/
ftp://mirror.cogentco.com/pub/CPAN/

Washington

<http://cpan.llarian.net/>
ftp://cpan.llarian.net/pub/CPAN/
ftp://ftp-mirror.internap.com/pub/CPAN/

Wisconsin

<http://cpan.mirrors.tds.net>
ftp://cpan.mirrors.tds.net/pub/CPAN
<http://mirror.sit.wisc.edu/pub/CPAN/>
ftp://mirror.sit.wisc.edu/pub/CPAN/

Oceania

Australia

<http://mirror.internode.on.net/pub/cpan/>
<ftp://mirror.internode.on.net/pub/cpan/>
<http://cpan.mirror.aussiehq.net.au/>
<http://mirror.as24220.net/cpan/>
<ftp://mirror.as24220.net/cpan/>

New Zealand

<ftp://ftp.auckland.ac.nz/pub/perl/CPAN/>
<http://cpan.inspire.net.nz>
<ftp://cpan.inspire.net.nz/cpan>
<http://cpan.catalyst.net.nz/CPAN/>
<ftp://cpan.catalyst.net.nz/pub/CPAN/>

South America

Argentina

<http://cpan.patan.com.ar/>
<http://cpan.localhost.net.ar>
<ftp://mirrors.localhost.net.ar/pub/mirrors/CPAN>

Brazil

<ftp://cpan.pop-mg.com.br/pub/CPAN/>
<http://ftp.pucpr.br/CPAN>
<ftp://ftp.pucpr.br/CPAN>
<http://cpan.kinghost.net/>

Chile

<http://cpan.dcc.uchile.cl/>
<ftp://cpan.dcc.uchile.cl/pub/lang/cpan/>

Colombia

<http://www.laqee.unal.edu.co/CPAN/>

RSYNC Mirrors

mirror.as24220.net::cpan
cpan.inode.at::CPAN
gd.tuwien.ac.at::CPAN
ftp.belnet.be::packages/cpan
rsync.linorg.usp.br::CPAN
rsync.arcticnetwork.ca::CPAN
CPAN.mirror.rafal.ca::CPAN
mirror.csclub.uwaterloo.ca::CPAN
theoryx5.uwinnipeg.ca::CPAN
www.laqee.unal.edu.co::CPAN
mirror.uni-c.dk::CPAN
rsync.nic.funet.fi::CPAN
rsync://distrib-coffee.ipsl.jussieu.fr/pub/mirrors/cpan/
mir1.ovh.net::CPAN
miroir-francais.fr::cpan
ftp.crihan.fr::CPAN
rsync://mirror.cict.fr/cpan/
rsync://mirror.netcologne.de/cpan/
ftp-stud.hs-esslingen.de::CPAN/
ftp.gwdg.de::FTP/languages/perl/CPAN/
cpan.mirror.iphh.net::CPAN
cpan.cpantesters.org::cpan
cpan.hu::CPAN
komo.vlsm.org::CPAN
mirror.unej.ac.id::cpan


```

ftp.esat.net::/pub/languages/perl/CPAN
ftp.heanet.ie::mirrors/ftp.perl.org/pub/CPAN
rsync.panu.it::CPAN
cpan.fastbull.org::CPAN
ftp.kddilabs.jp::cpan
ftp.nara.wide.ad.jp::cpan/
rsync://ftp.jaist.ac.jp/pub/CPAN/
rsync://ftp.riken.jp/cpan/
mirror.linuxiso.kz::CPAN
rsync://mirrors3.kernel.org/mirrors/CPAN/
rsync://rsync.osmirror.nl/cpan/
mirror.leaseweb.com::CPAN
cpan.nautile.nc::CPAN
mirror.icis.pcz.pl::CPAN
piotrkosoft.net::mirrors/CPAN
rsync://cpan.perl.pt/
ftp.kaist.ac.kr::cpan
cpan.sarang.net::CPAN
mirror.averse.net::cpan
rsync.oss.eznetsols.org
mirror.ac.za::cpan
ftp.is.co.za::IS-Mirror/ftp.cpan.org/
rsync://ftp.gui.uva.es/cpan/
rsync://mirrors4.kernel.org/mirrors/CPAN/
ftp.solnet.ch::CPAN
ftp.ulak.net.tr::CPAN
gafol.net::cpan
rsync.mirrorservice.org::ftp.funet.fi/pub/
rsync://rsync.mirror.8086.net/CPAN/
rsync.mirror.anlx.net::CPAN
mirror.bytemark.co.uk::CPAN
ftp.plig.net::CPAN
rsync://ftp.ticklers.org:CPAN/
mirrors.ibiblio.org::CPAN
cpan-du.viaverio.com::CPAN
mirror.hiwaay.net::CPAN
rsync://mira.sunsite.utk.edu/CPAN/
cpan.mirrors.tds.net::CPAN
mirror.its.uidaho.edu::cpan
rsync://mirror.cc.columbia.edu::cpan/
ftp.fxcorporate.com::CPAN
rsync.atlantic.net::CPAN
mirrors.kernel.org::mirrors/CPAN
rsync://mirrors2.kernel.org/mirrors/CPAN/
cpan.pair.com::CPAN
rsync://mirror.rit.edu/CPAN/
rsync://mirror.facebook.net/cpan/
rsync://mirrors1.kernel.org/mirrors/CPAN/
cpan-sj.viaverio.com::CPAN

```

For an up-to-date listing of CPAN sites, see <http://www.cpan.org/SITES> or <ftp://www.cpan.org/SITES> .

Modules: Creation, Use, and Abuse

(The following section is borrowed directly from Tim Bunce's modules file, available at your nearest CPAN site.)

Perl implements a class using a package, but the presence of a package doesn't imply the presence of a class. A package is just a namespace. A class is a package that provides subroutines that can be used as methods. A method is just a subroutine that expects, as its first argument, either the name of a package (for "static" methods), or a reference to something (for "virtual" methods).

A module is a file that (by convention) provides a class of the same name (sans the .pm), plus an import method in that class that can be called to fetch exported symbols. This module may implement some of its methods by loading dynamic C or C++ objects, but that should be totally transparent to the user of the module. Likewise, the module might set up an AUTOLOAD function to slurp in subroutine definitions on demand, but this is also transparent. Only the .pm file is required to exist. See [perlsub\(1\)](#), [perlobj\(1\)](#), and [AutoLoader](#) for details about the AUTOLOAD mechanism.

Guidelines for Module Creation

- Do similar modules already exist in some form?

If so, please try to reuse the existing modules either in whole or by inheriting useful features into a new class. If this is not practical try to get together with the module authors to work on extending or enhancing the functionality of the existing modules. A perfect example is the plethora of packages in perl4 for dealing with command line options.

If you are writing a module to expand an already existing set of modules, please coordinate with the author of the package. It helps if you follow the same naming scheme and module interaction scheme as the original author.

- Try to design the new module to be easy to extend and reuse.

Try to use `warnings;` (or use `warnings qw(...);`). Remember that you can add `no warnings qw(...);` to individual blocks of code that need less warnings.

Use blessed references. Use the two argument form of `bless` to bless into the class name given as the first parameter of the constructor, e.g.,:

```
sub new {
    my $class = shift;
    return bless {}, $class;
}
```

or even this if you'd like it to be used as either a static or a virtual method.

```
sub new {
    my $self = shift;
    my $class = ref($self) || $self;
    return bless {}, $class;
}
```

Pass arrays as references so more parameters can be added later (it's also faster). Convert functions into methods where appropriate. Split large methods into smaller more flexible ones. Inherit methods from other modules if appropriate.

Avoid class name tests like: `die "Invalid" unless ref $ref eq 'FOO'`. Generally you can delete the `eq 'FOO'` part with no harm at all. Let the objects look after themselves! Generally, avoid hard-wired class names as far as possible.

Avoid `$r->Class::func()` where using `@ISA=qw(... Class ...)` and `$r->func()` would work.

Use `autosplit` so little used or newly added functions won't be a burden to programs that don't use them. Add test functions to the module after `__END__` either using `AutoSplit` or by saying:

```
eval join(' ',<main::DATA>) || die $@ unless caller();
```

Does your module pass the 'empty subclass' test? If you say `@SUBCLASS::ISA = qw(YOURCLASS);` your applications should be able to use `SUBCLASS` in exactly the same way as

YOURCLASS. For example, does your application still work if you change: `$obj = YOURCLASS->new();` into: `$obj = SUBCLASS->new();` ?

Avoid keeping any state information in your packages. It makes it difficult for multiple other packages to use yours. Keep state information in objects.

Always use **-w**.

Try to use `strict;` (or use `strict qw(...);`). Remember that you can add `no strict qw(...);` to individual blocks of code that need less strictness.

Always use **-w**.

Follow the guidelines in `perlstyle`.

Always use **-w**.

- Some simple style guidelines

The [perlstyle\(1\)](#) manual supplied with Perl has many helpful points.

Coding style is a matter of personal taste. Many people evolve their style over several years as they learn what helps them write and maintain good code. Here's one set of assorted suggestions that seem to be widely used by experienced developers:

Use underscores to separate words. It is generally easier to read `$var_names_like_this` than `$VarNamesLikeThis`, especially for non-native speakers of English. It's also a simple rule that works consistently with `VAR_NAMES_LIKE_THIS`.

Package/Module names are an exception to this rule. Perl informally reserves lowercase module names for 'pragma' modules like `integer` and `strict`. Other modules normally begin with a capital letter and use mixed case with no underscores (need to be short and portable).

You may find it helpful to use letter case to indicate the scope or nature of a variable. For example:

```
$ALL_CAPS_HERE constants only (beware clashes with Perl vars)
$Some_Caps_Here package-wide global/static
$no_caps_here function scope my() or local() variables
```

Function and method names seem to work best as all lowercase. e.g., `$obj->as_string()`.

You can use a leading underscore to indicate that a variable or function should not be used outside the package that defined it.

- Select what to export.

Do NOT export method names!

Do NOT export anything else by default without a good reason!

Exports pollute the namespace of the module user. If you must export try to use `@EXPORT_OK` in preference to `@EXPORT` and avoid short or common names to reduce the risk of name clashes.

Generally anything not exported is still accessible from outside the module using the `ModuleName::item_name` (or `$blessed_ref->method`) syntax. By convention you can use a leading underscore on names to indicate informally that they are 'internal' and not for public use.

(It is actually possible to get private functions by saying: `my $subref = sub { ... }; &$subref;`. But there's no way to call that directly as a method, because a method must have a name in the symbol table.)

As a general rule, if the module is trying to be object oriented then export nothing. If it's just a collection of functions then `@EXPORT_OK` anything but use `@EXPORT` with caution.

- Select a name for the module.

This name should be as descriptive, accurate, and complete as possible. Avoid any risk of ambiguity.

Always try to use two or more whole words. Generally the name should reflect what is special about what the module does rather than how it does it. Please use nested module names to group informally or categorize a module. There should be a very good reason for a module not to have a nested name. Module names should begin with a capital letter.

Having 57 modules all called Sort will not make life easy for anyone (though having 23 called Sort::Quick is only marginally better :-). Imagine someone trying to install your module alongside many others.

If you are developing a suite of related modules/classes it's good practice to use nested classes with a common prefix as this will avoid namespace clashes. For example: Xyz::Control, Xyz::View, Xyz::Model etc. Use the modules in this list as a naming guide.

If adding a new module to a set, follow the original author's standards for naming modules and the interface to methods in those modules.

If developing modules for private internal or project specific use, that will never be released to the public, then you should ensure that their names will not clash with any future public module. You can do this either by using the reserved Local::* category or by using a category name that includes an underscore like Foo_Corp::*.

To be portable each component of a module name should be limited to 11 characters. If it might be used on MS-DOS then try to ensure each is unique in the first 8 characters. Nested modules make this easier.

For additional guidance on the naming of modules, please consult:

http://pause.perl.org/pause/query?ACTION=pause_namingmodules

or send mail to the <module-authors@perl.org> mailing list.

- Have you got it right?

How do you know that you've made the right decisions? Have you picked an interface design that will cause problems later? Have you picked the most appropriate name? Do you have any questions?

The best way to know for sure, and pick up many helpful suggestions, is to ask someone who knows. The <module-authors@perl.org> mailing list is useful for this purpose; it's also accessible via news interface as perl.module-authors at nntp.perl.org.

All you need to do is post a short summary of the module, its purpose and interfaces. A few lines on each of the main methods is probably enough. (If you post the whole module it might be ignored by busy people - generally the very people you want to read it!)

Don't worry about posting if you can't say when the module will be ready - just say so in the message. It might be worth inviting others to help you, they may be able to complete it for you!

- README and other Additional Files.

It's well known that software developers usually fully document the software they write. If, however, the world is in urgent need of your software and there is not enough time to write the full documentation please at least provide a README file containing:

- A description of the module/package/extension etc.
- A copyright notice - see below.
- Prerequisites - what else you may need to have.
- How to build it - possible changes to Makefile.PL etc.
- How to install it.
- Recent changes in this release, especially incompatibilities

- Changes / enhancements you plan to make in the future.

If the README file seems to be getting too large you may wish to split out some of the sections into separate files: INSTALL, Copying, ToDo etc.

- Adding a Copyright Notice.

How you choose to license your work is a personal decision. The general mechanism is to assert your Copyright and then make a declaration of how others may copy/use/modify your work.

Perl, for example, is supplied with two types of licence: The GNU GPL and The Artistic Licence (see the files README, Copying, and Artistic, or [perlgpl\(1\)](#) and [perlartistic](#)). Larry has good reasons for NOT just using the GNU GPL.

My personal recommendation, out of respect for Larry, Perl, and the Perl community at large is to state something simply like:

```
Copyright (c) 1995 Your Name. All rights reserved.
This program is free software; you can redistribute it and/or
modify it under the same terms as Perl itself.
```

This statement should at least appear in the README file. You may also wish to include it in a Copying file and your source files. Remember to include the other words in addition to the Copyright.

- Give the module a version/issue/release number.

To be fully compatible with the Exporter and MakeMaker modules you should store your module's version number in a non-my package variable called \$VERSION. This should be a positive floating point number with at least two digits after the decimal (i.e., hundredths, e.g. \$VERSION = "0.01"). Don't use a "1.3.2" style version. See Exporter for details.

It may be handy to add a function or method to retrieve the number. Use the number in announcements and archive file names when releasing the module (ModuleName-1.02.tar.Z). See [perldoc\(1\)](#) ExtUtils::MakeMaker.pm for details.

- How to release and distribute a module.

It's good idea to post an announcement of the availability of your module (or the module itself if small) to the comp.lang.perl.announce Usenet newsgroup. This will at least ensure very wide once-off distribution.

If possible, register the module with CPAN. You should include details of its location in your announcement.

Some notes about ftp archives: Please use a long descriptive file name that includes the version number. Most incoming directories will not be readable/listable, i.e., you won't be able to see your file after uploading it. Remember to send your email notification message as soon as possible after uploading else your file may get deleted automatically. Allow time for the file to be processed and/or check the file has been processed before announcing its location.

FTP Archives for Perl Modules:

Follow the instructions and links on:

```
http://www.cpan.org/modules/00modlist.long.html
http://www.cpan.org/modules/04pause.html
```

or upload to one of these sites:

```
https://pause.kbx.de/pause/
http://pause.perl.org/
```

and notify <modules@perl.org>.

By using the WWW interface you can ask the Upload Server to mirror your modules from your ftp or WWW site into your own directory on CPAN!

Please remember to send me an updated entry for the Module list!

- Take care when changing a released module.

Always strive to remain compatible with previous released versions. Otherwise try to add a mechanism to revert to the old behavior if people rely on it. Document incompatible changes.

Guidelines for Converting Perl 4 Library Scripts into Modules

- There is no requirement to convert anything.

If it ain't broke, don't fix it! Perl 4 library scripts should continue to work with no problems. You may need to make some minor changes (like escaping non-array @'s in double quoted strings) but there is no need to convert a .pl file into a Module for just that.

- Consider the implications.

All Perl applications that make use of the script will need to be changed (slightly) if the script is converted into a module. Is it worth it unless you plan to make other changes at the same time?

- Make the most of the opportunity.

If you are going to convert the script to a module you can use the opportunity to redesign the interface. The guidelines for module creation above include many of the issues you should consider.

- The pl2pm utility will get you started.

This utility will read *.pl files (given as parameters) and write corresponding *.pm files. The pl2pm utilities does the following:

- Adds the standard Module prologue lines
- Converts package specifiers from ' to ::
- Converts die(...) to croak(...)
- Several other minor changes

Being a mechanical process pl2pm is not bullet proof. The converted code will need careful checking, especially any package statements. Don't delete the original .pl file till the new .pm one works!

Guidelines for Reusing Application Code

- Complete applications rarely belong in the Perl Module Library.
- Many applications contain some Perl code that could be reused.

Help save the world! Share your code in a form that makes it easy to reuse.

- Break-out the reusable code into one or more separate module files.
- Take the opportunity to reconsider and redesign the interfaces.
- In some cases the 'application' can then be reduced to a small

fragment of code built on top of the reusable modules. In these cases the application could invoked as:

```
% perl -e 'use Module::Name; method(@ARGV)' ...
or
% perl -mModule::Name ... (in perl5.002 or higher)
```

NOTE

Perl does not enforce private and public parts of its modules as you may have been used to in other languages like C++, Ada, or Modula-17. Perl doesn't have an infatuation with enforced privacy. It would prefer that you stayed out of its living room because you weren't invited, not because it has a shotgun.

The module and its user have a contract, part of which is common law, and part of which is "written". Part of the common law contract is that a module doesn't pollute any namespace it wasn't asked to. The written

contract for the module (A.K.A. documentation) may make other provisions. But then you know when you use `RedefineTheWorld` that you're redefining the world and willing to take the consequences.