NAME

pod2man - Convert POD data to formatted *roff input

SYNOPSIS

pod2man [**--center**=*string*] [**--date**=*string*] [**--fixed**=*font*] [**--fixedbold**=*font*] [**--fixeditalic**=*font*] [**--fixedbolditalic**=*font*] [**--name**=*name*] [**--official**] [**--quotes**=*quotes*] [**--release**[=*version*]] [**--section**=*manext*] [**--stderr**] [**--utf8**] [**--verbose**] [*input* [*output*] ...]

pod2man **--help**

DESCRIPTION

pod2man is a front-end for Pod::Man, using it to generate *roff input from POD source. The resulting *roff code is suitable for display on a terminal using nroff(1), normally via man(1), or printing using troff(1).

input is the file to read for POD source (the POD can be embedded in code). If *input* isn't given, it defaults to `STDIN`. *output*, if given, is the file to which to write the formatted output. If *output* isn't given, the formatted output is written to `STDOUT`. Several POD files can be processed in the same **pod2man** invocation (saving module load and compile times) by providing multiple pairs of *input* and *output* files on the command line.

--section, **--release**, **--center**, **--date**, and **--official** can be used to set the headers and footers to use; if not given, Pod::Man will assume various defaults. See below or *Pod::Man* for details.

pod2man assumes that your *roff formatters have a fixed-width font named `CW`. If yours is called something else (like `CR`), use **--fixed** to specify it. This generally only matters for troff output for printing. Similarly, you can set the fonts used for bold, italic, and bold italic fixed-width output.

Besides the obvious pod conversions, Pod::Man, and therefore pod2man also takes care of formatting func(), func(n), and simple variable references like $foo or @bar so you don't have to use code escapes for them; complex expressions like `$fred{'stuff'}` will still need to be escaped, though. It also translates dashes that aren't used as hyphens into en dashes, makes long dashes--like this--into proper em dashes, fixes "paired quotes," and takes care of several other troff-specific tweaks. See *Pod::Man* for complete information.

OPTIONS

-c *string*, **--center**=*string*

Sets the centered page header to *string*. The default is "User Contributed Perl Documentation", but also see **--official** below.

-d *string*, **--date**=*string*

Set the left-hand footer string to this value. By default, the modification date of the input file will be used, or the current date if input comes from `STDIN`.

--fixed=*font*

The fixed-width font to use for verbatim text and code. Defaults to `CW`. Some systems may want `CR` instead. Only matters for troff(1) output.

--fixedbold=*font*

Bold version of the fixed-width font. Defaults to `CB`. Only matters for troff(1) output.

--fixeditalic=*font*

Italic version of the fixed-width font (actually, something of a misnomer, since most fixed-width fonts only have an oblique version, not an italic version). Defaults to `CI`. Only matters for troff(1) output.

--fixedbolditalic=*font*

Bold italic (probably actually oblique) version of the fixed-width font. Pod::Man doesn't assume you have this, and defaults to CB. Some systems (such as Solaris) have this font available as CX. Only matters for troff(1) output.

-h, **--help**

Print out usage information.

-l, **--lax**

No longer used. **pod2man** used to check its input for validity as a manual page, but this should now be done by *podchecker(1)* instead. Accepted for backward compatibility; this option no longer does anything.

-n *name*, **--name**=*name*

Set the name of the manual page to *name*. Without this option, the manual name is set to the uppercased base name of the file being converted unless the manual section is 3, in which case the path is parsed to see if it is a Perl module path. If it is, a path like `.../lib/Pod/Man.pm` is converted into a name like `Pod::Man`. This option, if given, overrides any automatic determination of the name.

Note that this option is probably not useful when converting multiple POD files at once. The convention for Unix man pages for commands is for the man page title to be in all-uppercase even if the command isn't.

-o, **--official**

Set the default header to indicate that this page is part of the standard Perl release, if **--center** is not also given.

-q *quotes*, **--quotes**=*quotes*

Sets the quote marks used to surround C<> text to *quotes*. If *quotes* is a single character, it is used as both the left and right quote; if *quotes* is two characters, the first character is used as the left quote and the second as the right quoted; and if *quotes* is four characters, the first two are used as the left quote and the second two as the right quote.

quotes may also be set to the special value `none`, in which case no quote marks are added around C<> text (but the font is still changed for troff output).

-r, **--release**

Set the centered footer. By default, this is the version of Perl you run **pod2man** under. Note that some system an macro sets assume that the centered footer will be a modification date and will prepend something like "Last modified: "; if this is the case, you may want to set **--release** to the last modified date and **--date** to the version number.

-s, **--section**

Set the section for the `.TH` macro. The standard section numbering convention is to use 1 for user commands, 2 for system calls, 3 for functions, 4 for devices, 5 for file formats, 6 for games, 7 for miscellaneous information, and 8 for administrator commands. There is a lot of variation here, however; some systems (like Solaris) use 4 for file formats, 5 for miscellaneous information, and 7 for devices. Still others use 1m instead of 8, or some mix of both. About the only section numbers that are reliably consistent are 1, 2, and 3.

By default, section 1 will be used unless the file ends in `.pm`, in which case section 3 will be selected.

--stderr

By default, **pod2man** puts any errors detected in the POD input in a POD ERRORS section in the output manual page. If **--stderr** is given, errors are sent to standard error instead and the POD ERRORS section is suppressed.

-u, **--utf8**

By default, **pod2man** produces the most conservative possible *roff output to try to ensure that it will work with as many different *roff implementations as possible. Many *roff implementations cannot handle non-ASCII characters, so this means all non-ASCII characters are converted either to a *roff escape sequence that tries to create a properly accented character (at least for troff output) or to X.

This option says to instead output literal UTF-8 characters. If your *roff implementation can handle it, this is the best output format to use and avoids corruption of documents containing non-ASCII characters. However, be warned that *roff source with literal UTF-8 characters is not supported by many implementations and may even result in segfaults and other bad behavior.

Be aware that, when using this option, the input encoding of your POD source must be properly declared unless it is US-ASCII or Latin-1. POD input without an `=encoding` command will be assumed to be in Latin-1, and if it's actually in UTF-8, the output will be double-encoded. See *perlpod(1)* for more information on the `=encoding` command.

-v, **--verbose**

Print out the name of each output file as it is being generated.

DIAGNOSTICS

If **pod2man** fails with errors, see *Pod::Man* and *Pod::Simple* for information about what those errors might mean.

EXAMPLES

```
pod2man program > program.1
pod2man SomeModule.pm /usr/perl/man/man3/SomeModule.3
pod2man --section=7 note.pod > note.7
```

If you would like to print out a lot of man page continuously, you probably want to set the C and D registers to set contiguous page numbering and even/odd paging, at least on some versions of man(7).

```
troff -man -rC1 -rD1 perl.1 perldata.1 perlsyn.1 ...
```

To get index entries on STDERR, turn on the F register, as in:

```
troff -man -rF1 perl.1
```

The indexing merely outputs messages via `.tm` for each major page, section, subsection, item, and any X<> directives. See *Pod::Man* for more details.

BUGS

Lots of this documentation is duplicated from *Pod::Man*.

NOTES

For those not sure of the proper layout of a man page, here are some notes on writing a proper man page.

The name of the program being documented is conventionally written in bold (using B<>) wherever it occurs, as are all program options. Arguments should be written in italics (I<>). Functions are traditionally written in italics; if you write a function as function(), Pod::Man will take care of this for you. Literal code or commands should be in C<>. References to other man pages should be in the form `manpage(section)`, and Pod::Man will automatically format those appropriately. As an exception, it's traditional not to use this form when referring to module documentation; use `L<Module::Name>` instead.

References to other programs or functions are normally in the form of man page references so that cross-referencing tools can provide the user with links and the like. It's possible to overdo this, though, so be careful not to clutter your documentation with too much markup.

The major headers should be set out using a `=head1` directive, and are historically written in the rather startling ALL UPPER CASE format, although this is not mandatory. Minor headers may be included using `=head2`, and are typically in mixed case.

The standard sections of a manual page are:

NAME

Mandatory section; should be a comma-separated list of programs or functions documented by this POD page, such as:

```
foo, bar - programs to do something
```

Manual page indexers are often extremely picky about the format of this section, so don't put anything in it except this line. A single dash, and only a single dash, should separate the list of programs or functions from the description. Do not use any markup such as C<> or B<>. Functions should not be qualified with `( )` or the like. The description should ideally fit on a single line, even if a man program replaces the dash with a few tabs.

SYNOPSIS

A short usage summary for programs and functions. This section is mandatory for section 3 pages.

DESCRIPTION

Extended description and discussion of the program or functions, or the body of the documentation for man pages that document something else. If particularly long, it's a good idea to break this up into subsections `=head2` directives like:

```
=head2 Normal Usage

=head2 Advanced Features

=head2 Writing Configuration Files
```

or whatever is appropriate for your documentation.

OPTIONS

Detailed description of each of the command-line options taken by the program. This should be separate from the description for the use of things like *Pod::Usage*. This is normally presented as a list, with each option as a separate `=item`. The specific option string should be enclosed in B<>. Any values that the option takes should be enclosed in I<>. For example, the section for the option **--section**=*manext* would be introduced with:

```
=item B<--section>=I<manext>
```

Synonymous options (like both the short and long forms) are separated by a comma and a space on the same `=item` line, or optionally listed as their own item with a reference to the canonical name. For example, since **--section** can also be written as **-s**, the above would be:

```
=item B<-s> I<manext>, B<--section>=I<manext>
```

(Writing the short option first is arguably easier to read, since the long option is long enough to draw the eye to it anyway and the short option can otherwise get lost in visual noise.)

RETURN VALUE

What the program or function returns, if successful. This section can be omitted for programs

whose precise exit codes aren't important, provided they return 0 on success as is standard. It should always be present for functions.

ERRORS

Exceptions, error return codes, exit statuses, and errno settings. Typically used for function documentation; program documentation uses DIAGNOSTICS instead. The general rule of thumb is that errors printed to STDOUT or STDERR and intended for the end user are documented in DIAGNOSTICS while errors passed internal to the calling program and intended for other programmers are documented in ERRORS. When documenting a function that sets errno, a full list of the possible errno values should be given here.

DIAGNOSTICS

All possible messages the program can print out--and what they mean. You may wish to follow the same documentation style as the Perl documentation; see perldiag(1) for more details (and look at the POD source as well).

If applicable, please include details on what the user should do to correct the error; documenting an error as indicating "the input buffer is too small" without telling the user how to increase the size of the input buffer (or at least telling them that it isn't possible) aren't very useful.

EXAMPLES

Give some example uses of the program or function. Don't skimp; users often find this the most useful part of the documentation. The examples are generally given as verbatim paragraphs.

Don't just present an example without explaining what it does. Adding a short paragraph saying what the example will do can increase the value of the example immensely.

ENVIRONMENT

Environment variables that the program cares about, normally presented as a list using `=over`, `=item`, and `=back`. For example:

```
=over 6

=item HOME

Used to determine the user's home directory.  F<.foorc> in this
directory is read for configuration details, if it exists.

=back
```

Since environment variables are normally in all uppercase, no additional special formatting is generally needed; they're glaring enough as it is.

FILES

All files used by the program or function, normally presented as a list, and what it uses them for. File names should be enclosed in F<>. It's particularly important to document files that will be potentially modified.

CAVEATS

Things to take special care with, sometimes called WARNINGS.

BUGS

Things that are broken or just don't work quite right.

RESTRICTIONS

Bugs you don't plan to fix. :-)

NOTES

Miscellaneous commentary.

AUTHOR

Who wrote it (use AUTHORS for multiple people). Including your current e-mail address (or some e-mail address to which bug reports should be sent) so that users have a way of contacting you is a good idea. Remember that program documentation tends to roam the wild for far longer than you expect and pick an e-mail address that's likely to last if possible.

HISTORY

Programs derived from other sources sometimes have this, or you might keep a modification log here. If the log gets overly long or detailed, consider maintaining it in a separate file, though.

COPYRIGHT AND LICENSE

For copyright

```
    Copyright YEAR(s) by YOUR NAME(s)
```

(No, (C) is not needed. No, "all rights reserved" is not needed.)

For licensing the easiest way is to use the same licensing as Perl itself:

```
    This library is free software; you may redistribute it and/or
modify
    it under the same terms as Perl itself.
```

This makes it easy for people to use your module with Perl. Note that this licensing is neither an endorsement or a requirement, you are of course free to choose any licensing.

SEE ALSO

Other man pages to check out, like man(1), man(7), makewhatis(8), or catman(8). Normally a simple list of man pages separated by commas, or a paragraph giving the name of a reference work. Man page references, if they use the standard `name(section)` form, don't have to be enclosed in L<> (although it's recommended), but other things in this section probably should be when appropriate.

If the package has a mailing list, include a URL or subscription instructions here.

If the package has a web site, include a URL here.

In addition, some systems use CONFORMING TO to note conformance to relevant standards and MT-LEVEL to note safeness for use in threaded programs or signal handlers. These headings are primarily useful when documenting parts of a C library. Documentation of object-oriented libraries or modules may use CONSTRUCTORS and METHODS sections for detailed documentation of the parts of the library and save the DESCRIPTION section for an overview; other large modules may use FUNCTIONS for similar reasons. Some people use OVERVIEW to summarize the description if it's quite long.

Section ordering varies, although NAME should *always* be the first section (you'll break some man page systems otherwise), and NAME, SYNOPSIS, DESCRIPTION, and OPTIONS generally always occur first and in that order if present. In general, SEE ALSO, AUTHOR, and similar material should be left for last. Some systems also move WARNINGS and NOTES to last. The order given above should be reasonable for most purposes.

Finally, as a general note, try not to use an excessive amount of markup. As documented here and in *Pod::Man*, you can safely leave Perl variables, function names, man page references, and the like unadorned by markup and the POD translators will figure it out for you. This makes it much easier to later edit the documentation. Note that many existing translators (including this one currently) will do the wrong thing with e-mail addresses when wrapped in L<>, so don't do that.

For additional information that may be more accurate for your specific system, see either *man(5)* or *man(7)* depending on your system manual section numbering conventions.

SEE ALSO

Pod::Man, *Pod::Simple*, *man(1)*, *nroff(1)*, *perlpod(1)*, *podchecker(1)*, *troff(1)*, *man(7)*

The man page documenting the an macro set may be *man(5)* instead of *man(7)* on your system.

The current version of this script is always available from its web site at *http://www.eyrie.org/~eagle/software/podlators/*. It is also part of the Perl core distribution as of 5.6.0.

AUTHOR

Russ Allbery <rra@stanford.edu>, based *very* heavily on the original **pod2man** by Larry Wall and Tom Christiansen. Large portions of this documentation, particularly the sections on the anatomy of a proper man page, are taken from the **pod2man** documentation by Tom.

COPYRIGHT AND LICENSE