NAME

Pod::Text - Convert POD data to formatted ASCII text

SYNOPSIS

```
use Pod::Text;
my $parser = Pod::Text->new (sentence => 0, width => 78);

# Read POD from STDIN and write to STDOUT.
$parser->parse_from_filehandle;

# Read POD from file.pod and write to file.txt.
$parser->parse_from_file ('file.pod', 'file.txt');
```

DESCRIPTION

Pod::Text is a module that can convert documentation in the POD format (the preferred language for documenting Perl) into formatted ASCII. It uses no special formatting controls or codes whatsoever, and its output is therefore suitable for nearly any device.

As a derived class from Pod::Simple, Pod::Text supports the same methods and interfaces. See *Pod::Simple* for all the details; briefly, one creates a new parser with `Pod::Text->new()` and then normally calls parse_file().

new() can take options, in the form of key/value pairs, that control the behavior of the parser. The currently recognized options are:

alt

If set to a true value, selects an alternate output format that, among other things, uses a different heading style and marks `=item` entries with a colon in the left margin. Defaults to false.

code

If set to a true value, the non-POD parts of the input file will be included in the output. Useful for viewing code documented with POD blocks with the POD rendered and the code left intact.

errors

How to report errors. `die` says to throw an exception on any POD formatting error. `stderr` says to report errors on standard error, but not to throw an exception. `pod` says to include a POD ERRORS section in the resulting documentation summarizing the errors. `none` ignores POD errors entirely, as much as possible.

The default is `output`.

indent

The number of spaces to indent regular text, and the default indentation for `=over` blocks. Defaults to 4.

loose

If set to a true value, a blank line is printed after a `=head1` heading. If set to false (the default), no blank line is printed after `=head1`, although one is still printed after `=head2`. This is the default because it's the expected formatting for manual pages; if you're formatting arbitrary text documents, setting this to true may result in more pleasing output.

margin

The width of the left margin in spaces. Defaults to 0. This is the margin for all text, including headings, not the amount by which regular text is indented; for the latter, see the *indent* option. To set the right margin, see the *width* option.

nourls

Normally, L<> formatting codes with a URL but anchor text are formatted to show both the anchor text and the URL. In other words:

```
L<foo|http://example.com/>
```

is formatted as:

```
foo <http://example.com/>
```

This option, if set to a true value, suppresses the URL when anchor text is given, so this example would be formatted as just `foo`. This can produce less cluttered output in cases where the URLs are not particularly important.

quotes

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

This may also be set to the special value `none`, in which case no quote marks are added around C<> text.

sentence

If set to a true value, Pod::Text will assume that each sentence ends in two spaces, and will try to preserve that spacing. If set to false, all consecutive whitespace in non-verbatim paragraphs is compressed into a single space. Defaults to true.

stderr

Send error messages about invalid POD to standard error instead of appending a POD ERRORS section to the generated output. This is equivalent to setting `errors` to `stderr` if `errors` is not already set. It is supported for backward compatibility.

utf8

By default, Pod::Text uses the same output encoding as the input encoding of the POD source (provided that Perl was built with PerlIO; otherwise, it doesn't encode its output). If this option is given, the output encoding is forced to UTF-8.

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.

width

The column at which to wrap text on the right-hand side. Defaults to 76.

The standard Pod::Simple method parse_file() takes one argument, the file or file handle to read from, and writes output to standard output unless that has been changed with the output_fh() method. See *Pod::Simple* for the specific details and for other alternative interfaces.

DIAGNOSTICS

Bizarre space in item

Item called without tag

(W) Something has gone wrong in internal `=item` processing. These messages indicate a bug in Pod::Text; you should never see them.

Can't open %s for reading: %s

(F) Pod::Text was invoked via the compatibility mode pod2text() interface and the input file it

was given could not be opened.

Invalid errors setting "%s"

(F) The `errors` parameter to the constructor was set to an unknown value.

Invalid quote specification "%s"

(F) The quote specification given (the `quotes` option to the constructor) was invalid. A quote specification must be one, two, or four characters long.

POD document had syntax errors

(F) The POD document being formatted had syntax errors and the `errors` option was set to `die`.

BUGS

Encoding handling assumes that PerlIO is available and does not work properly if it isn't. The `utf8` option is therefore not supported unless Perl is built with PerlIO support.

CAVEATS

If Pod::Text is given the `utf8` option, the encoding of its output file handle will be forced to UTF-8 if possible, overriding any existing encoding. This will be done even if the file handle is not created by Pod::Text and was passed in from outside. This maintains consistency regardless of PERL_UNICODE and other settings.

If the `utf8` option is not given, the encoding of its output file handle will be forced to the detected encoding of the input POD, which preserves whatever the input text is. This ensures backward compatibility with earlier, pre-Unicode versions of this module, without large numbers of Perl warnings.

This is not ideal, but it seems to be the best compromise. If it doesn't work for you, please let me know the details of how it broke.

NOTES

This is a replacement for an earlier Pod::Text module written by Tom Christiansen. It has a revamped interface, since it now uses Pod::Simple, but an interface roughly compatible with the old Pod::Text::pod2text() function is still available. Please change to the new calling convention, though.

The original Pod::Text contained code to do formatting via termcap sequences, although it wasn't turned on by default and it was problematic to get it to work at all. This rewrite doesn't even try to do that, but a subclass of it does. Look for *Pod::Text::Termcap*.

SEE ALSO

Pod::Simple, *Pod::Text::Termcap*, *perlpod(1)*, *pod2text(1)*

The current version of this module 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 Pod::Text by Tom Christiansen <tchrist@mox.perl.com> and its conversion to Pod::Parser by Brad Appleton <bradapp@enteract.com>. Sean Burke's initial conversion of Pod::Man to use Pod::Simple provided much-needed guidance on how to use Pod::Simple.

COPYRIGHT AND LICENSE