

NAME

Pod::Checker, podchecker() - check pod documents for syntax errors

SYNOPSIS

```
use Pod::Checker;

$syntax_okay = podchecker($filepath, $outputpath, %options);

my $checker = new Pod::Checker %options;
$checker->parse_from_file($filepath, \*STDERR);
```

OPTIONS/ARGUMENTS

`$filepath` is the input POD to read and `$outputpath` is where to write POD syntax error messages. Either argument may be a scalar indicating a file-path, or else a reference to an open filehandle. If unspecified, the input-file defaults to `\*STDIN`, and the output-file defaults to `\*STDERR`.

podchecker()

This function can take a hash of options:

-warnings => *val*

Turn warnings on/off. *val* is usually 1 for on, but higher values trigger additional warnings. See *Warnings*.

DESCRIPTION

podchecker will perform syntax checking of Perl5 POD format documentation.

Curious/ambitious users are welcome to propose additional features they wish to see in **Pod::Checker** and **podchecker** and verify that the checks are consistent with *perlpod*.

The following checks are currently performed:

- Unknown '=xxxx' commands, unknown 'X<...>' interior-sequences, and unterminated interior sequences.
- Check for proper balancing of `=begin` and `=end`. The contents of such a block are generally ignored, i.e. no syntax checks are performed.
- Check for proper nesting and balancing of `=over`, `=item` and `=back`.
- Check for same nested interior-sequences (e.g. `L<...L<...>...>`).
- Check for malformed or non-existing entities `E<...>`.
- Check for correct syntax of hyperlinks `L<...>`. See *perlpod* for details.
- Check for unresolved document-internal links. This check may also reveal misspelled links that seem to be internal links but should be links to something else.

DIAGNOSTICS

Errors

* empty `=headn`

A heading (`=head1` or `=head2`) without any text? That ain't no heading!

* `=over` on line *N* without closing `=back`

The `=over` command does not have a corresponding `=back` before the next heading (`=head1` or `=head2`) or the end of the file.

* =item without previous =over

* =back without previous =over

An =item or =back command has been found outside a =over/=back block.

* No argument for =begin

A =begin command was found that is not followed by the formatter specification.

* =end without =begin

A standalone =end command was found.

* Nested =begin's

There were at least two consecutive =begin commands without the corresponding =end.

Only one =begin may be active at a time.

* =for without formatter specification

There is no specification of the formatter after the =for command.

* unresolved internal link *NAME*

The given link to *NAME* does not have a matching node in the current POD. This also happened when a single word node name is not enclosed in "".

* Unknown command "*CMD*"

An invalid POD command has been found. Valid are =head1, =head2, =head3, =head4, =over, =item, =back, =begin, =end, =for, =pod, =cut

* Unknown interior-sequence "*SEQ*"

An invalid markup command has been encountered. Valid are: B<>, C<>, E<>, F<>, I<>, L<>, S<>, X<>, Z<>

* nested commands *CMD*<...*CMD*<...>...>

Two nested identical markup commands have been found. Generally this does not make sense.

* garbled entity *STRING*

The *STRING* found cannot be interpreted as a character entity.

* Entity number out of range

An entity specified by number (dec, hex, oct) is out of range (1-255).

* malformed link L<>

The link found cannot be parsed because it does not conform to the syntax described in *perlpod*.

* nonempty Z<>

The Z<> sequence is supposed to be empty.

* empty X<>

The index entry specified contains nothing but whitespace.

* Spurious text after =pod / =cut

The commands =pod and =cut do not take any arguments.

* Spurious character(s) after =back

The =back command does not take any arguments.

Warnings

These may not necessarily cause trouble, but indicate mediocre style.

* multiple occurrence of link target *name*

The POD file has some `=item` and/or `=head` commands that have the same text. Potential hyperlinks to such a text cannot be unique then. This warning is printed only with warning level greater than one.

* line containing nothing but whitespace in paragraph

There is some whitespace on a seemingly empty line. POD is very sensitive to such things, so this is flagged. **vi** users switch on the **list** option to avoid this problem.

* file does not start with `=head`

The file starts with a different POD directive than `head`. This is most probably something you do not want.

* previous `=item` has no contents

There is a list `=item` right above the flagged line that has no text contents. You probably want to delete empty items.

* preceding non-item paragraph(s)

A list introduced by `=over` starts with a text or verbatim paragraph, but continues with `=item`s. Move the non-item paragraph out of the `=over/=back` block.

* `=item` type mismatch (*one* vs. *two*)

A list started with e.g. a bullet-like `=item` and continued with a numbered one. This is obviously inconsistent. For most translators the type of the *first* `=item` determines the type of the list.

* *N* unescaped `<>` in paragraph

Angle brackets not written as `<lt>` and `<gt>` can potentially cause errors as they could be misinterpreted as markup commands. This is only printed when the `-warnings` level is greater than 1.

* Unknown entity

A character entity was found that does not belong to the standard ISO set or the POD specials `verbar` and `sol`.

* No items in `=over`

The list opened with `=over` does not contain any items.

* No argument for `=item`

`=item` without any parameters is deprecated. It should either be followed by `*` to indicate an unordered list, by a number (optionally followed by a dot) to indicate an ordered (numbered) list or simple text for a definition list.

* empty section in previous paragraph

The previous section (introduced by a `=head` command) does not contain any text. This usually indicates that something is missing. Note: A `=head1` followed immediately by `=head2` does not trigger this warning.

* Verbatim paragraph in NAME section

The NAME section (`=head1 NAME`) should consist of a single paragraph with the script/module name, followed by a dash ``-'` and a very short description of what the thing is good for.

* =head*n* without preceding higher level

For example if there is a =head2 in the POD file prior to a =head1.

Hyperlinks

There are some warnings with respect to malformed hyperlinks:

* ignoring leading/trailing whitespace in link

There is whitespace at the beginning or the end of the contents of L<...>.

* (section) in '\$page' deprecated

There is a section detected in the page name of L<...>, e.g. L<passwd(2)>. POD hyperlinks may point to POD documents only. Please write C<passwd(2)> instead. Some formatters are able to expand this to appropriate code. For links to (builtin) functions, please say L<perlfunc/mkdir>, without ().

* alternative text/node '%s' contains non-escaped | or /

The characters | and / are special in the L<...> context. Although the hyperlink parser does its best to determine which "/" is text and which is a delimiter in case of doubt, one ought to escape these literal characters like this:

```
/      E<sol>
|      E<verbar>
```

RETURN VALUE

podchecker returns the number of POD syntax errors found or -1 if there were no POD commands at all found in the file.

EXAMPLES

See *SYNOPSIS*

INTERFACE

While checking, this module collects document properties, e.g. the nodes for hyperlinks (=head*X*, =item) and index entries (X<>). POD translators can use this feature to syntax-check and get the nodes in a first pass before actually starting to convert. This is expensive in terms of execution time, but allows for very robust conversions.

Since PodParser-1.24 the **Pod::Checker** module uses only the **poderror** method to print errors and warnings. The summary output (e.g. "Pod syntax OK") has been dropped from the module and has been included in **podchecker** (the script). This allows users of **Pod::Checker** to control completely the output behavior. Users of **podchecker** (the script) get the well-known behavior.

Pod::Checker->new(%options)

Return a reference to a new Pod::Checker object that inherits from Pod::Parser and is used for calling the required methods later. The following options are recognized:

-warnings => num Print warnings if num is true. The higher the value of num, the more warnings are printed. Currently there are only levels 1 and 2.

-quiet => num If num is true, do not print any errors/warnings. This is useful when Pod::Checker is used to munge POD code into plain text from within POD formatters.

\$checker->poderror(@args)

\$checker->poderror({%opts}, @args)

Internal method for printing errors and warnings. If no options are given, simply prints "@_". The following options are recognized and used to form the output:

-msg

A message to print prior to @args.

-line

The line number the error occurred in.

-file

The file (name) the error occurred in.

-severity

The error level, should be 'WARNING' or 'ERROR'.

`$checker->num_errors()`

Set (if argument specified) and retrieve the number of errors found.

`$checker->num_warnings()`

Set (if argument specified) and retrieve the number of warnings found.

`$checker->name()`

Set (if argument specified) and retrieve the canonical name of POD as found in the =head1 NAME section.

`$checker->node()`

Add (if argument specified) and retrieve the nodes (as defined by =headX and =item) of the current POD. The nodes are returned in the order of their occurrence. They consist of plain text, each piece of whitespace is collapsed to a single blank.

`$checker->idx()`

Add (if argument specified) and retrieve the index entries (as defined by X<>) of the current POD. They consist of plain text, each piece of whitespace is collapsed to a single blank.

`$checker->hyperlink()`

Add (if argument specified) and retrieve the hyperlinks (as defined by L<>) of the current POD. They consist of a 2-item array: line number and Pod::Hyperlink object.

AUTHOR

Please report bugs using <http://rt.cpan.org>.

Brad Appleton <bradapp@enteract.com> (initial version), Marek Rouchal <marekr@cpan.org>

Based on code for **Pod::Text::pod2text()** written by Tom Christiansen <tchrist@mox.perl.com>