NAME

Pod::Select, podselect() - extract selected sections of POD from input

SYNOPSIS

```
use Pod::Select;

## Select all the POD sections for each file in @filelist
## and print the result on standard output.
podselect(@filelist);

## Same as above, but write to tmp.out
podselect({-output => "tmp.out"}, @filelist):

## Select from the given filelist, only those POD sections that are
## within a 1st level section named any of: NAME, SYNOPSIS, OPTIONS.
podselect({-sections => ["NAME|SYNOPSIS", "OPTIONS"]}, @filelist):

## Select the "DESCRIPTION" section of the PODs from STDIN and write
## the result to STDERR.
podselect({-output => ">&STDERR", -sections => ["DESCRIPTION"]}, \*STDIN);
```

or

```
use Pod::Select;

## Create a parser object for selecting POD sections from the input
$parser = new Pod::Select();

## Select all the POD sections for each file in @filelist
## and print the result to tmp.out.
$parser->parse_from_file("<&STDIN", "tmp.out");

## Select from the given filelist, only those POD sections that are
## within a 1st level section named any of: NAME, SYNOPSIS, OPTIONS.
$parser->select("NAME|SYNOPSIS", "OPTIONS");
for (@filelist) { $parser->parse_from_file($_); }

## Select the "DESCRIPTION" and "SEE ALSO" sections of the PODs from
## STDIN and write the result to STDERR.
$parser->select("DESCRIPTION");
$parser->add_selection("SEE ALSO");
$parser->parse_from_filehandle(\*STDIN, \*STDERR);
```

REQUIRES

perl5.005, Pod::Parser, Exporter, Carp

EXPORTS

podselect()

DESCRIPTION

NOTE: This module is considered legacy; modern Perl releases (5.18 and higher) are going to remove Pod-Parser from core and use Pod-Simple for all things POD.

podselect() is a function which will extract specified sections of pod documentation from an input stream. This ability is provided by the **Pod::Select** module which is a subclass of **Pod::Parser** **Pod::Select** provides a method named ***select()*** to specify the set of POD sections to select for processing/printing. ***podselect()*** merely creates a **Pod::Select** object and then invokes the ***podselect()*** followed by ***parse_from_file()***.

SECTION SPECIFICATIONS

podselect() and ***Pod::Select::select()*** may be given one or more ‘‘section specifications’’ to restrict the text processed to only the desired set of sections and their corresponding subsections. A section specification is a string containing one or more Perl-style regular expressions separated by forward slashes (‘‘/’’). If you need to use a forward slash literally within a section title you can escape it with a backslash (‘‘\/’’).

The formal syntax of a section specification is:

- *head1-title-regex*/*head2-title-regex*/...

Any omitted or empty regular expressions will default to ‘‘.*’’. Please note that each regular expression given is implicitly anchored by adding ‘‘^’’ and ‘‘$’’ to the beginning and end. Also, if a given regular expression starts with a ‘‘!’’ character, then the expression is *negated* (so `!foo` would match anything *except* `foo`).

Some example section specifications follow.

- Match the `NAME` and `SYNOPSIS` sections and all of their subsections:

  ```
  NAME|SYNOPSIS
  ```

- Match only the `Question` and `Answer` subsections of the `DESCRIPTION` section:

  ```
  DESCRIPTION/Question|Answer
  ```

- Match the `Comments` subsection of *all* sections:

  ```
  /Comments
  ```

- Match all subsections of `DESCRIPTION` *except* for `Comments`:

  ```
  DESCRIPTION/!Comments
  ```

- Match the `DESCRIPTION` section but do *not* match any of its subsections:

  ```
  DESCRIPTION/!.+
  ```

- Match all top level sections but none of their subsections:

  ```
  /!.+
  ```

OBJECT METHODS

The following methods are provided in this module. Each one takes a reference to the object itself as an implicit first parameter.

curr_headings()

```
($head1, $head2, $head3, ...) = $parser->curr_headings();
$head1 = $parser->curr_headings(1);
```

This method returns a list of the currently active section headings and subheadings in the document being parsed. The list of headings returned corresponds to the most recently parsed paragraph of the input.

If an argument is given, it must correspond to the desired section heading number, in which case only the specified section heading is returned. If there is no current section heading at the specified level, then `undef` is returned.

select()

```
$parser->select($section_spec1,$section_spec2,...);
```

This method is used to select the particular sections and subsections of POD documentation that are to be printed and/or processed. The existing set of selected sections is *replaced* with the given set of sections. See ***add_selection()*** for adding to the current set of selected sections.

Each of the `$section_spec` arguments should be a section specification as described in ‘‘SECTION SPECIFICATIONS’’. The section specifications are parsed by this method and the resulting regular expressions are stored in the invoking object.

If no `$section_spec` arguments are given, then the existing set of selected sections is cleared out

(which means `all` sections will be processed).

This method should *not* normally be overridden by subclasses.

add_selection()

```
$parser->add_selection($section_spec1,$section_spec2,...);
```

This method is used to add to the currently selected sections and subsections of POD documentation that are to be printed and/or processed. See <*select()*> for replacing the currently selected sections.

Each of the `$section_spec` arguments should be a section specification as described in ''SECTION SPECIFICATIONS''. The section specifications are parsed by this method and the resulting regular expressions are stored in the invoking object.

This method should *not* normally be overridden by subclasses.

clear_selections()

```
$parser->clear_selections();
```

This method takes no arguments, it has the exact same effect as invoking <*select()*> with no arguments.

match_section()

```
$boolean = $parser->match_section($heading1,$heading2,...);
```

Returns a value of true if the given section and subsection heading titles match any of the currently selected section specifications in effect from prior calls to ***select()*** and ***add_selection()*** (or if there are no explicitly selected/deselected sections).

The arguments `$heading1`, `$heading2`, etc. are the heading titles of the corresponding sections, subsections, etc. to try and match. If `$headingN` is omitted then it defaults to the current corresponding section heading title in the input.

This method should *not* normally be overridden by subclasses.

is_selected()

```
$boolean = $parser->is_selected($paragraph);
```

This method is used to determine if the block of text given in `$paragraph` falls within the currently selected set of POD sections and subsections to be printed or processed. This method is also responsible for keeping track of the current input section and subsections. It is assumed that `$paragraph` is the most recently read (but not yet processed) input paragraph.

The value returned will be true if the `$paragraph` and the rest of the text in the same section as `$paragraph` should be selected (included) for processing; otherwise a false value is returned.

EXPORTED FUNCTIONS

The following functions are exported by this module. Please note that these are functions (not methods) and therefore `do not` take an implicit first argument.

podselect()

```
podselect(\%options,@filelist);
```

podselect will print the raw (untranslated) POD paragraphs of all POD sections in the given input files specified by `@filelist` according to the options given in `\%options`.

If any argument to **podselect** is a reference to a hash (associative array) then the values with the following keys are processed as follows:

-output

A string corresponding to the desired output file (or ''>&STDOUT'' or ''>&STDERR''), or a filehandle to write on. The default is to use standard output.

-sections

A reference to an array of sections specifications (as described in ''SECTION SPECIFICATIONS'') which indicate the desired set of POD sections and subsections to be selected from input. If no section specifications are given, then all sections of the PODs are used.

All other arguments are optional and should correspond to filehandles to read from or the names of input files containing POD sections. A file name of "“, ”-“ or ”<&STDIN" will be interpreted to mean standard input (which is the default if no arguments are given).

PRIVATE METHODS AND DATA

Pod::Select makes uses a number of internal methods and data fields which clients should not need to see or use. For the sake of avoiding name collisions with client data and methods, these methods and fields are briefly discussed here. Determined hackers may obtain further information about them by reading the **Pod::Select** source code.

Private data fields are stored in the hash-object whose reference is returned by the ***new()*** constructor for this class. The names of all private methods and data-fields used by **Pod::Select** begin with a prefix of ‘‘_’’ and match the regular expression `/^_\w+$/`.

SEE ALSO

Pod::Parser

AUTHOR

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

Brad Appleton <bradapp@enteract.com>

Based on code for **pod2text** written by Tom Christiansen <tchrist@mox.perl.com>

Pod::Select is part of the Pod::Parser distribution.