NAME

Pod::LaTeX - Convert Pod data to formatted Latex

SYNOPSIS

```
use Pod::LaTeX;
my $parser = Pod::LaTeX->new ( );

$parser->parse_from_filehandle;

$parser->parse_from_file ('file.pod', 'file.tex');
```

DESCRIPTION

`Pod::LaTeX` is a module to convert documentation in the Pod format into Latex. The ***pod2latex*** command uses this module for translation.

`Pod::LaTeX` is a derived class from *Pod::Select*.

OBJECT METHODS

The following methods are provided in this module. Methods inherited from `Pod::Select` are not described in the public interface.

`initialize`

Initialise the object. This method is subclassed from `Pod::Parser`. The base class method is invoked. This method defines the default behaviour of the object unless overridden by supplying arguments to the constructor.

Internal settings are defaulted as well as the public instance data. Internal hash values are accessed directly (rather than through a method) and start with an underscore.

This method should not be invoked by the user directly.

Data Accessors

The following methods are provided for accessing instance data. These methods should be used for accessing configuration parameters rather than assuming the object is a hash.

Default values can be supplied by using these names as keys to a hash of arguments when using the `new()` constructor.

AddPreamble

Logical to control whether a `latex` preamble is to be written. If true, a valid `latex` preamble is written before the pod data is written. This is similar to:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{textcomp}
\begin{document}
```

but will be more complicated if table of contents and indexing are required. Can be used to set or retrieve the current value.

```
$add = $parser->AddPreamble();
$parser->AddPreamble(1);
```

If used in conjunction with `AddPostamble` a full latex document will be written that could be immediately processed by `latex`.

For some pod escapes it may be necessary to include the amsmath package. This is not yet added to the preamble automatically.

AddPostamble

Logical to control whether a standard `latex` ending is written to the output file after the document has been processed. In its simplest form this is simply:

```
\end{document}
```

but can be more complicated if a index is required. Can be used to set or retrieve the current value.

```
$add = $parser->AddPostamble();
$parser->AddPostamble(1);
```

If used in conjunction with `AddPreaamble` a full latex document will be written that could be immediately processed by `latex`.

Head1Level

The `latex` sectioning level that should be used to correspond to a pod `=head1` directive. This can be used, for example, to turn a `=head1` into a `latex subsection`. This should hold a number corresponding to the required position in an array containing the following elements:

```
[0] chapter
[1] section
[2] subsection
[3] subsubsection
[4] paragraph
[5] subparagraph
```

Can be used to set or retrieve the current value:

```
$parser->Head1Level(2);
$sect = $parser->Head1Level;
```

Setting this number too high can result in sections that may not be reproducible in the expected way. For example, setting this to 4 would imply that `=head3` do not have a corresponding `latex` section (`=head1` would correspond to a `paragraph`).

A check is made to ensure that the supplied value is an integer in the range 0 to 5.

Default is for a value of 1 (i.e. a `section`).

Label

This is the label that is prefixed to all `latex` label and index entries to make them unique. In general, pods have similarly titled sections (NAME, DESCRIPTION etc) and a `latex` label will be multiply defined if more than one pod document is to be included in a single `latex` file. To overcome this, this label is prefixed to a label whenever a label is required (joined with an underscore) or to an index entry (joined by an exclamation mark which is the normal index separator). For example, `\label{text}` becomes `\label{Label_text}`.

Can be used to set or retrieve the current value:

```
$label = $parser->Label;
$parser->Label($label);
```

This label is only used if `UniqueLabels` is true. Its value is set automatically from the `NAME` field if `ReplaceNAMEwithSection` is true. If this is not the case it must be set manually before starting the parse.

Default value is `undef`.

LevelNoNum

Control the point at which `latex` section numbering is turned off. For example, this can be used to make sure that `latex` sections are numbered but subsections are not.

Can be used to set or retrieve the current value:

```
$lev = $parser->LevelNoNum;
$parser->LevelNoNum(2);
```

The argument must be an integer between 0 and 5 and is the same as the number described in `Head1Level` method description. The number has nothing to do with the pod heading number, only the `latex` sectioning.

Default is 2. (i.e. `latex` subsections are written as `subsection*` but sections are numbered).

MakeIndex

Controls whether `latex` commands for creating an index are to be inserted into the preamble and postamble

```
$makeindex = $parser->MakeIndex;
$parser->MakeIndex(0);
```

Irrelevant if both `AddPreamble` and `AddPostamble` are false (or equivalently, `UserPreamble` and `UserPostamble` are set).

Default is for an index to be created.

ReplaceNAMEwithSection

This controls whether the `NAME` section in the pod is to be translated literally or converted to a slightly modified output where the section name is the pod name rather than "NAME".

If true, the pod segment

```
=head1 NAME

pod::name - purpose

=head1 SYNOPSIS
```

is converted to the `latex`

```
\section{pod::name\label{pod_name}\index{pod::name}}

Purpose

\subsection*{SYNOPSIS\label{pod_name_SYNOPSIS}%
             \index{pod::name!SYNOPSIS}}
```

(dependent on the value of `Head1Level` and `LevelNoNum`). Note that subsequent `head1` directives translate to subsections rather than sections and that the labels and index now include the pod name (dependent on the value of `UniqueLabels`).

The `Label` is set from the pod name regardless of any current value of `Label`.

```
$mod = $parser->ReplaceNAMEwithSection;
$parser->ReplaceNAMEwithSection(0);
```

Default is to translate the pod literally.

StartWithNewPage

If true, each pod translation will begin with a `latex` `\clearpage`.

```
$parser->StartWithNewPage(1);
$newpage = $parser->StartWithNewPage;
```

Default is false.

TableOfContents

If true, a table of contents will be created. Irrelevant if `AddPreamble` is false or `UserPreamble` is set.

```
$toc = $parser->TableOfContents;
$parser->TableOfContents(1);
```

Default is false.

UniqueLabels

If true, the translator will attempt to make sure that each `latex` label or index entry will be uniquely identified by prefixing the contents of `Label`. This allows multiple documents to be combined without clashing common labels such as `DESCRIPTION` and `SYNOPSIS`

```
$parser->UniqueLabels(1);
$unq = $parser->UniqueLabels;
```

Default is true.

UserPreamble

User supplied `latex` preamble. Added before the pod translation data.

If set, the contents will be prepended to the output file before the translated data regardless of the value of `AddPreamble`. `MakeIndex` and `TableOfContents` will also be ignored.

UserPostamble

User supplied `latex` postamble. Added after the pod translation data.

If set, the contents will be prepended to the output file after the translated data regardless of the value of `AddPostamble`. `MakeIndex` will also be ignored.

Lists

Contains details of the currently active lists. The array contains `Pod::List` objects. A new `Pod::List` object is created each time a list is encountered and it is pushed onto this stack. When the list context ends, it is popped from the stack. The array will be empty if no lists are active.

Returns array of list information in list context Returns array ref in scalar context

Subclassed methods

The following methods override methods provided in the `Pod::Select` base class. See `Pod::Parser` and `Pod::Select` for more information on what these methods require.

begin_pod

Writes the `latex` preamble if requested. Only writes something if AddPreamble is true. Writes a standard header unless a UserPreamble is defined.

end_pod

Write the closing `latex` code. Only writes something if AddPostamble is true. Writes a standard header unless a UserPostamble is defined.

command

Process basic pod commands.

verbatim

Verbatim text

textblock

Plain text paragraph.

interior_sequence

Interior sequence expansion

List Methods

Methods used to handle lists.

begin_list

Called when a new list is found (via the `over` directive). Creates a new `Pod::List` object and stores it on the list stack.

```
$parser->begin_list($indent, $line_num);
```

end_list

Called when the end of a list is found (the `back` directive). Pops the `Pod::List` object off the stack of lists and writes the `latex` code required to close a list.

```
$parser->end_list($line_num);
```

add_item

Add items to the list. The first time an item is encountered (determined from the state of the current `Pod::List` object) the type of list is determined (ordered, unnumbered or description) and the relevant latex code issued.

```
$parser->add_item($paragraph, $line_num);
```

Methods for headings

head

Print a heading of the required level.

```
$parser->head($level, $paragraph, $parobj);
```

The first argument is the pod heading level. The second argument is the contents of the heading. The 3rd argument is a Pod::Paragraph object so that the line number can be extracted.

Internal methods

Internal routines are described in this section. They do not form part of the public interface. All private methods start with an underscore.

_output

Output text to the output filehandle. This method must be always be called to output parsed text.

```
$parser->_output($text);
```

Does not write anything if a =begin is active that should be ignored.

_replace_special_chars

Subroutine to replace characters that are special in `latex` with the escaped forms

```
$escaped = $parser->_replace_special_chars($paragraph);
```

Need to call this routine before interior_sequences are munged but not if verbatim. It must be called before interpolation of interior sequences so that curly brackets and special latex characters inserted during interpolation are not themselves escaped. This means that < and > can not be modified here since the text still contains interior sequences.

Special characters and the `latex` equivalents are:

```
}      \}
{      \{
_      \_
$      \$
%      \%
&      \&
\      $\backslash$
^      \^{}
~      \~{}
#      \#
```

_replace_special_chars_late

Replace special characters that can not be replaced before interior sequence interpolation. See `_replace_special_chars` for a routine to replace special characters prior to interpolation of interior sequences.

Does the following transformation:

```
<   $<$
>   $>$
|   $|$
```

_create_label

Return a string that can be used as an internal reference in a `latex` document (i.e. accepted by the `\label` command)

```
$label = $parser->_create_label($string)
```

If UniqueLabels is true returns a label prefixed by Label() This can be suppressed with an optional second argument.

```
$label = $parser->_create_label($string, $suppress);
```

If a second argument is supplied (of any value including undef) the Label() is never prefixed. This means that this routine can be called to create a Label() without prefixing a previous setting.

_create_index

Similar to `_create_label` except an index entry is created. If `UniqueLabels` is true, the index entry is prefixed by the current `Label` and an exclamation mark.

```
$ind = $parser->_create_index($paragraph);
```

An exclamation mark is used by `makeindex` to generate sub-entries in an index.

_clean_latex_commands

Removes latex commands from text. The latex command is assumed to be of the form `\command{ text }`. "`text`" is retained

```
$clean = $parser->_clean_latex_commands($text);
```

_split_delimited

Split the supplied string into two parts at approximately the specified word boundary. Special care is made to make sure that it does not split in the middle of some curly brackets.

e.g. "this text is \textbf{very bold}" would not be split into "this text is \textbf{very" and " bold".

```
($hunk1, $hunk2) = $self->_split_delimited( $para, $length);
```

The length indicates the maximum length of hunk1.

NOTES

Compatible with `latex2e` only. Can not be used with `latex` v2.09 or earlier.

A subclass of `Pod::Select` so that specific pod sections can be converted to `latex` by using the `select` method.

Some HTML escapes are missing and many have not been tested.

SEE ALSO

Pod::Parser, *Pod::Select*, *pod2latex*, *Pod::Simple*.

AUTHORS

Tim Jenness <tjenness@cpan.org>

Bug fixes and improvements have been received from: Simon Cozens <simon@cozens.net>, Mark A. Hershberger <mah@everybody.org>, Marcel Grunauer <marcel@codewerk.com>, Hugh S Myers <hsmyers@sdragons.com>, Peter J Acklam <jacklam@math.uio.no>, Sudhi Herle <sudhi@herle.net>, Ariel Scolnicov <ariels@compugen.co.il>, Adriano Rodrigues Ferreira <ferreira@triang.com.br>, R. de Vries <r.de.vries@dutchspace.nl> and Dave Mitchell <davem@iabyn.com>.

COPYRIGHT

REVISION

Id