

NAME

perlpodextensions - optional extensions to the Pod markup language

DESCRIPTION

This document defines a number of optional extensions to the Pod (Plain Old Documentation) specification (see *perlpodspec*).

Pod formatters may implement any or none of these extensions, however any which are implemented must confirm to this specification.

If a Pod formatter encounters an extension which is not supported, it should warn the user in an appropriate manner.

Definitions

In this document, the terms "must" / "must not", "should" / "should not", and "may" have their standard meanings as described in "RFC 2119".

Pod Commands

"=ff"

This command (formfeed) indicates to a Pod formatter that a new page should be started.

Implementation of the =ff directive will be dependent on the output format's native support of a 'page' concept. For example, a print-based formatter such as LaTeX or PDF would actually move to a new physical page. An HTML formatter may use the =ff command to split a longer document into a number of .html files, possibly including hyperlinks or some other navigational structure to move between them.

If the output format does not feature a 'page' concept, the =ff directive must be silently ignored.

Note that the =ff command must only have an effect if the current page has already been written to, i.e. given the the Pod

```
Hello
```

```
=ff
```

```
=ff
```

```
World!
```

the formatter must ignore the second =ff command so as not to output a completely blank page.

Pod Formatting Codes

O<name> -- embedded object

The O<name> formatting code is used to embed an external object into the output of the Pod formatter. Typically this command will be used to include images or diagrams within a document.

Pod parsers must recognise two distinct attributes within the O< . . . > sequence:

First:

The caption or title to be given to the embedded object. If omitted, this must be `undef`.

Depending on the desired output format, the object title may be included as metadata (e.g. in HTML, the object title may appear as the `alt` attribute of an `` tag), however formatters may also choose to ignore object titles completely.

Second:

The filename or URL of the object. If omitted, this must be reported as an error.

If the object name matches the regular expression `m/\A\W+:[^\s]\S*\z/` it is defined as a URL - anything else is assumed to be the path to a file on the local system.

These attributes must be separated by a pipe (|) symbol. If the pipe symbol is not present, the parser must assume that the complete text in the formatting sequence represents the filename or URL.

All of the following must therefore be recognised as valid sequences:

```
O<image.png>
O</usr/local/share/images/logo.png>
O<Figure 1: High level flowchart|flowchart.png>
O<http://example.com/image.png>
O<Caption|http://example.com/image.png>
```

Implementation notes:

- The exact method of embedding objects will differ from formatter to formatter. For example, given the command

```
O<image.png>
```

an HTML formatter may simply convert this into an `` tag, whereas a PDF formatter would need to load the image, possibly scale it to the page dimensions, and include it as an inline object within the output PDF file.

- If the object name is a URL, Pod formatters may either link to the URL or download the object to a temporary location and process it as a local file. Any errors encountered in downloading the object should be reported. All temporary files must be removed when processing is complete.
- If the object name refers to a file, then any errors loading or parsing the file should be reported.
- If a relative path to a file is given, this is deemed to be relative to the the Pod file which is currently being processed, or to the current working directory if the Pod formatter is processing from a stream.
- If the file format of the object is not directly supported by the target output format (e.g. embedding an audio file in a plain text document), the Pod formatter may treat the `O<...>` code as a link (`L<...>`). Alternatively the formatter may provide some other way of marking embedded objects (such as a *References* section). Whatever method is chosen, it must be possible for the reader of the finished document to identify the names of any embedded objects and their intended position within the document.

SEE ALSO

perlpod, *perlpodspec*, <http://perl.jonallen.info/projects/pod2pdf>

AUTHOR

Jon Allen <jj@jonallen.info>