



Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'Pod::Escapes.3perl'

\$ man Pod::Escapes.3perl

Pod::Escapes(3perl) Perl Programmers Reference Guide Pod::Escapes(3perl)

NAME

Pod::Escapes - for resolving Pod E<...> sequences

SYNOPSIS

```
use Pod::Escapes qw(e2char);
...la la la, parsing POD, la la la...
$text = e2char($e_node->label);
unless(defined $text) {
    print "Unknown E sequence \"", $e_node->label, "\"!";
}
...else print/interpolate $text...
```

DESCRIPTION

This module provides things that are useful in decoding Pod E<...> sequences. Presumably, it should be used only by Pod parsers and/or formatters.

By default, Pod::Escapes exports none of its symbols. But you can request any of them to be exported. Either request them individually, as with "use Pod::Escapes qw(symbolname symbolname2...);", or you can do "use Pod::Escapes qw(:ALL);" to get all exportable symbols.

GOODIES

`e2char($e_content)`

Given a name or number that could appear in a "E<name_or_num>" sequence, this returns the string that it stands for. For example, `"e2char('sol')`", `"e2char('47')`", `"e2char('0x2F')`", and `"e2char('057')`" all return `"/"`, because `"E<sol>"`, `"E<47>"`, `"E<0x2f>"`, and `"E<057>"`, all mean `"/"`. If the name has no known value (as with a name of "qacute") or is syntactically invalid (as with a name of "1/4"), this returns `undef`.

`e2charnum($e_content)`

Given a name or number that could appear in a "E<name_or_num>" sequence, this returns the number of the Unicode character that this stands for. For example, `"e2char('sol')`", `"e2char('47')`", `"e2char('0x2F')`", and `"e2char('057')`" all return `47`, because `"E<sol>"`, `"E<47>"`, `"E<0x2f>"`, and `"E<057>"`, all mean `"/"`, whose Unicode number is `47`. If the name has no known value (as with a name of "qacute") or is syntactically invalid (as with a name of "1/4"), this returns `undef`.

`$Name2character{name}`

Maps from names (as in "E<name>") like "eacute" or "sol" to the string that each stands for. Note that this does not include numerics (like "64" or "x981c"). Under old Perl versions (before 5.7) you get a "?" in place of characters whose Unicode value is over 255.

`$Name2character_number{name}`

Maps from names (as in "E<name>") like "eacute" or "sol" to the Unicode value that each stands for. For example, `$Name2character_number{'eacute'}` is `201`, and `$Name2character_number{'eacute'}` is `8364`. You get the correct Unicode value, regardless of the version of Perl you're using -- which differs from `%Name2character`'s behavior under pre-5.7 Perls.

Note that this hash does not include numerics (like "64" or "x981c").

`$Latin1Code_to_fallback{integer}`

For numbers in the range 160 (0x00A0) to 255 (0x00FF), this maps from the character code for a Latin-1 character (like 233 for lowercase e-acute) to the US-ASCII character that best approximates it (like "e"). You may find this useful if you are rendering POD in a format that you think deals well only with US-ASCII characters.

`$Latin1Char_to_fallback{character}`

Just as above, but maps from characters (like "\xE9", lowercase e-acute) to characters (like "e").

`$Code2USASCII{integer}`

This maps from US-ASCII codes (like 32) to the corresponding character (like space, for 32). Only characters 32 to 126 are defined. This is meant for use by "e2char(\$x)" when it senses that it's running on a non-ASCII platform (where chr(32) doesn't get you a space -- but \$Code2USASCII{32} will). It's documented here just in case you might find it useful.

CAVEATS

On Perl versions before 5.7, Unicode characters with a value over 255 (like lambda or emdash) can't be conveyed. This module does work under such early Perl versions, but in the place of each such character, you get a "?". Latin-1 characters (characters 160-255) are unaffected.

Under EBCDIC platforms, "e2char(\$n)" may not always be the same as "chr(e2charnum(\$n))", and ditto for \$Name2character{\$name} and "chr(\$Name2character_number{\$name})", because the strings are returned as native, and the numbers are returned as Unicode. However, for Perls starting with v5.8, "e2char(\$n)" is the same as "chr(utf8::unicode_to_native(e2charnum(\$n)))", and ditto for \$Name2character{\$name} and "chr(utf8::unicode_to_native(\$Name2character_number{\$name}))".

SEE ALSO

Pod::Browser - a pod web server based on Catalyst.

Pod::Checker - check pod documents for syntax errors.

Pod::Coverage - check if the documentation for a module is comprehensive.

perlpod - description of pod format (for people documenting with pod).

perlpodspec - specification of pod format (for people processing it).

Text::Unidecode - ASCII transliteration of Unicode text.

REPOSITORY

<<https://github.com/neilbowers/Pod-Escapes>>

COPYRIGHT AND DISCLAIMERS

Copyright (c) 2001-2004 Sean M. Burke. All rights reserved.

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

This program is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

Portions of the data tables in this module are derived from the entity declarations in the W3C XHTML specification.

Currently (October 2001), that's these three:

<http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent>

<http://www.w3.org/TR/xhtml1/DTD/xhtml-special.ent>

<http://www.w3.org/TR/xhtml1/DTD/xhtml-symbol.ent>

AUTHOR

Sean M. Burke "sburke@cpan.org"

Now being maintained by Neil Bowers <neilb@cpan.org>

perl v5.34.0

2023-11-23

Pod::Escapes(3perl)