

NAME

perlutil - utilities packaged with the Perl distribution

DESCRIPTION

Along with the Perl interpreter itself, the Perl distribution installs a range of utilities on your system. There are also several utilities which are used by the Perl distribution itself as part of the install process. This document exists to list all of these utilities, explain what they are for and provide pointers to each module's documentation, if appropriate.

DOCUMENTATION

perldoc

The main interface to Perl's documentation is `perldoc`, although if you're reading this, it's more than likely that you've already found it. `perldoc` will extract and format the documentation from any file in the current directory, any Perl module installed on the system, or any of the standard documentation pages, such as this one. Use `perldoc <name>` to get information on any of the utilities described in this document.

pod2man and *pod2text*

If it's run from a terminal, `perldoc` will usually call `pod2man` to translate POD (Plain Old Documentation - see *perlpod* for an explanation) into a manpage, and then run `man` to display it; if `man` isn't available, `pod2text` will be used instead and the output piped through your favourite pager.

pod2html and *pod2latex*

As well as these two, there are two other converters: `pod2html` will produce HTML pages from POD, and `pod2latex`, which produces LaTeX files.

pod2usage

If you just want to know how to use the utilities described here, `pod2usage` will just extract the "USAGE" section; some of the utilities will automatically call `pod2usage` on themselves when you call them with `-help`.

podselect

`pod2usage` is a special case of `podselect`, a utility to extract named sections from documents written in POD. For instance, while utilities have "USAGE" sections, Perl modules usually have "SYNOPSIS" sections: `podselect -s "SYNOPSIS" ...` will extract this section for a given file.

podchecker

If you're writing your own documentation in POD, the `podchecker` utility will look for errors in your markup.

splain

`splain` is an interface to `perldiag` - paste in your error message to it, and it'll explain it for you.

roffitall

The `roffitall` utility is not installed on your system but lives in the `pod/` directory of your Perl source kit; it converts all the documentation from the distribution to **roff* format, and produces a typeset PostScript or text file of the whole lot.

CONVERTORS

To help you convert legacy programs to Perl, we've included three conversion filters:

a2p

`a2p` converts `awk` scripts to Perl programs; for example, `a2p -F`: on the simple `awk` script

`{print $2}` will produce a Perl program based around this code:

```
while (<>) {
    ($Fld1,$Fld2) = split(/[:\n]/, $_, 9999);
    print $Fld2;
}
```

s2p

Similarly, *s2p* converts *sed* scripts to Perl programs. *s2p* run on `s/foo/bar` will produce a Perl program based around this:

```
while (<>) {
    chomp;
    s/foo/bar/g;
    print if $printit;
}
```

find2perl

Finally, *find2perl* translates *find* commands to Perl equivalents which use the *File::Find* module. As an example, `find2perl . -user root -perm 4000 -print` produces the following callback subroutine for `File::Find`:

```
sub wanted {
    my ($dev,$ino,$mode,$nlink,$uid,$gid);
    (($dev,$ino,$mode,$nlink,$uid,$gid) = lstat($_)) &&
    $uid == $uid{'root'} &&
    (($mode & 0777) == 04000);
    print("$name\n");
}
```

As well as these filters for converting other languages, the *p2pm* utility will help you convert old-style Perl 4 libraries to new-style Perl5 modules.

Administration

libnetcfg

To display and change the libnet configuration run the *libnetcfg* command.

Development

There are a set of utilities which help you in developing Perl programs, and in particular, extending Perl with C.

perlbug

perlbug is the recommended way to report bugs in the perl interpreter itself or any of the standard library modules back to the developers; please read through the documentation for *perlbug* thoroughly before using it to submit a bug report.

h2ph

Back before Perl had the XS system for connecting with C libraries, programmers used to get library constants by reading through the C header files. You may still see `require 'syscall.ph'` or similar around - the *.ph* file should be created by running *h2ph* on the corresponding *.h* file. See the *h2ph* documentation for more on how to convert a whole bunch of header files at once.

c2ph and *pstruct*

c2ph and *pstruct*, which are actually the same program but behave differently depending on how they are called, provide another way of getting at C with Perl - they'll convert C structures and

union declarations to Perl code. This is deprecated in favour of *h2xs* these days.

h2xs

h2xs converts C header files into XS modules, and will try and write as much glue between C libraries and Perl modules as it can. It's also very useful for creating skeletons of pure Perl modules.

dprofpp

Perl comes with a profiler, the *Devel::DProf* module. The *dprofpp* utility analyzes the output of this profiler and tells you which subroutines are taking up the most run time. See *Devel::DProf* for more information.

perlcc

perlcc is the interface to the experimental Perl compiler suite.

SEE ALSO

perldoc, *pod2man*, *perlpod*, *pod2html*, *pod2usage*, *podselect*, *podchecker*, *splain*, *perldiag*, *roffitall*, *a2p*, *s2p*, *find2perl*, *File::Find*, *pl2pm*, *perlbug*, *h2ph*, *c2ph*, *h2xs*, *dprofpp*, *Devel::DProf*, *perlcc*