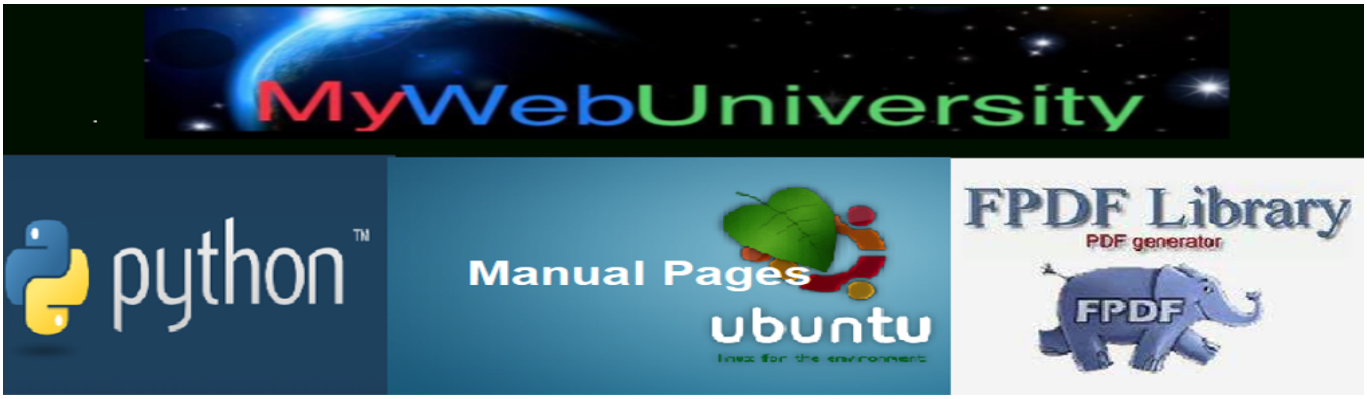

Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'Proc::ProcessTable.3pm'

$ man Proc::ProcessTable.3pm

Proc::ProcessTable(3pm) User Contributed Perl Documentation Proc::ProcessTable(3pm)

NAME

Proc::ProcessTable - Perl extension to access the unix process table

SYNOPSIS

```
use Proc::ProcessTable;

my $p = Proc::ProcessTable->new( 'cache_ttys' => 1 );
my @fields = $p->fields;
my $ref = $p->table;
```

DESCRIPTION

Perl interface to the unix process table.

METHODS

new Creates a new ProcessTable object. The constructor can take the following flags:

enable_ttys -- causes the constructor to use the tty determination code, which is the default behavior. Setting this to 0 disables this code, thus preventing the module from traversing the device tree, which on some systems, can be quite large and/or contain invalid device paths (for example, Solaris does not clean up invalid device entries when disks are swapped). If this is specified with cache_ttys, a warning is

generated and the cache_ttys is overridden to be false.

cache_ttys -- causes the constructor to look for and use a file that caches a mapping of tty names to device numbers, and to create the file if it doesn't exist. This feature requires the Storable module. By default, the cache file name consists of a prefix /tmp/TTYDEVS_ and a byte order tag. The file name can be accessed (and changed) via $Proc::ProcessTable::TTYDEVSFILE.

fields

Returns a list of the field names supported by the module on the current architecture.

table

Reads the process table and returns a reference to an array of Proc::ProcessTable::Process objects. Attributes of a process object are returned by accessors named for the attribute; for example, to get the uid of a process just do:

$process->uid

The priority and pgrp methods also allow values to be set, since these are supported directly by internal perl functions.

EXAMPLES

```
# A cheap and sleazy version of ps
use Proc::ProcessTable;

my $FORMAT = "%-6s %-10s %-8s %-24s %s\n";
my $t = Proc::ProcessTable->new;
printf($FORMAT, "PID", "TTY", "STAT", "START", "COMMAND");
foreach my $p ( @{$t->table} ){
  printf($FORMAT,
        $p->pid,
        $p->ttydev,
        $p->state,
```

```
        scalar(localtime($p->start)),
        $p->cmndline);
}

# Dump all the information in the current process table
use Proc::ProcessTable;

my $t = Proc::ProcessTable->new;

foreach my $p (@{$t->table}) {
 print "--------------------------------\n";
 foreach my $f ($t->fields){
   print $f, ":  ", $p->{$f}, "\n";
 }
}
```

CAVEATS

Please see the file README in the distribution for a list of supported operating systems.

Please see the file PORTING for information on how to help make this work on your OS.

AUTHOR

J. Bargsten, D. Urist

SEE ALSO

Proc::ProcessTable::Process, perl(1).