

Re: How to capture pid

Source: <http://coding.derkeiler.com/Archive/Perl/perl.beginners/2004-03/0251.html>

From: Zentara (zentara_at_highstream.net)

Date: 03/04/04

To: beginners@perl.org

Date: Thu, 04 Mar 2004 09:23:43 -0500

On Wed, 03 Mar 2004 10:36:06 -0600, reader@newsguy.com (Harry Putnam)

wrote:

> zentara <zentara@highstream.net> writes:

>> On Tue, 02 Mar 2004 18:07:58 -0600, reader@newsguy.com (Harry Putnam)

>> wrote:

>>> For some reason the pid printed does not match the output of ps:

>>>

>>> actual sample script"

>>>

>>> Note the script outputs 15173 and ps shows 30335

>>>

>>> Do you have an idea what is happening here?

>>

>> Yeah, you are getting the parent pid in the perl script.

>> Use Proc::ProcessTable to get the pid of your program name.

>> Here is an example:

>

> ---8<snip script

>

> Not sure I understand what I'm supposed to be seeing.

> None of this output resembles a pid.

>

> NOt sure I get what this has to do with two pids involved.

Well it was just an example to demonstrate that the pid returned to your perl script is sometimes not the pid of the actual programname which you are running. I was expecting you to watch top and ps as it was running, to see the relation between the parent pid and the pid you are looking for. The command "ps --forest" is a handy way to get an ascii graphical output to look at.

But here is a more direct example, and in your case, I don't see a problem. The desired pid is returned on my system.

Back to John Krahn's answer:

Now when you run this code, also have a second shell opened and

perl.beginners: Re: How to capture pid

in it run "ps --forest". The pid of tcpdump is accurate.

```
#####  
#!/usr/bin/perl  
$|=1;  
open FILE, ">$0.log" or  
    die "Cannot open file: $!";  
my $pid = open DUMP, 'tcpdump -v -ieth0 |' or  
    die "Cannot open pipe from tcpdump: $!";  
  
print "Pid was $pid\n";  
print FILE while <DUMP>;  
close DUMP or die "Cannot close pipe from tcpdump: $!";  
close FILE;  
__END__  
#####
```

But sometimes this won't work, as when you are calling a perl script, or doing an "exec". In your original code, you are doing an exec.

Do a "ps --forest" as you run the following, which is using exec:

```
#####  
#!/usr/bin/perl  
use Proc::ProcessTable;  
$|=1;  
my $cmd = 'top';  
  
if(fork() == 0){exec( "$cmd >> ppid.log" )}  
  
print "Pid is $pid\n";  
  
my $t1 = new Proc::ProcessTable;  
my $pid1;  
my $commandline = 'top';  
foreach my $p (@{$t1->table}){  
    if($p->cmdline =~ /\Q$commandline\E/){  
        $pid1 = $p->pid; #correct pid to use  
        print "pid1->$pid1\n";  
    }  
}  
<>;  
__END__  
#####
```

If you don't want to mess around with these details, you probably would like Proc::Background

```
#####3  
#!/usr/bin/perl  
  
use Proc::Background;  
my $proc;  
my @commands=($cmd1,$cmd2);  
foreach my $foo (@commands) {
```

perl.beginners: Re: How to capture pid

```
$proc = Proc::Background->new($foo);  
}
```

```
#####
```

```
--  
I'm not really a human, but I play one on earth.  
http://zentara.net/japh.html
```