

Re: Killing a process that takes too long

Re: Killing a process that takes too long

Source: <http://coding.derkeiler.com/Archive/Perl/perl.beginners/2006-11/msg00691.html>

- *From:* info@xxxxxxxxxxxx (D. Bolliger)
 - *Date:* Wed, 22 Nov 2006 13:53:24 +0100
-

Jen Spinney am Dienstag, 21. November 2006 22:06:

On 11/21/06, Jen Spinney <jen.spinney@xxxxxxxx> wrote:

On 11/21/06, Tom Phoenix <tom@xxxxxxxxxxxxxxxx> wrote:

On 11/21/06, Jen Spinney <jen.spinney@xxxxxxxx> wrote:

I want to make a system call, and then kill
the process if it takes
too long.

So, if I do a `ps -af`, I can see that my perl
script is a goner, but
the process spawned from the system call is
still alive.

Yes; if you use `system()` to start a sub-process, you're letting
perl
manage it; so there's no way to get the process-ID.

You may instead use `fork` and `exec`; this lets you use the
process-ID to
manage the process directly. Be sure to use `wait` or `waitpid` to
reap
the completed child process, so as not to leave zombies.

Is that what you needed? Hope this helps!

--Tom Phoenix
Stonehenge Perl Training

Thanks Tom!

I replaced the system call with `fork` and `exec` and it works just the
way I want it to:

Re: Killing a process that takes too long

```
use warnings;
use strict;

my $pid;

eval {
local $SIG{ALRM} = sub {
print "Timed out\n";
kill 'INT', $pid;
die 'alarm';
};
alarm 5;
if ($pid = fork)
{
waitpid ($pid, 0);
}
else
{
exec ('sleep 45');
}
alarm 0;
};
die if $@ && $@ !~ /alarm/;
print "Exited normally.\n";

__END__
```

For my actual program, I had to do a bit more work because I have semicolons and other shell stuff in the command (which seems not to do so well with exec?), but I figured out a workaround.

So, thanks again!

– Jen

Sorry for top-posting my last post. I ran into a bit of snag when replacing system with exec. If you replace 'sleep 45' in my last post with "perl -e 'while (1) {print 1} | tee test.txt'", the pipe really messes things up. Can anyone give me guidance as to how I should set up a pipe when using fork () and exec () to replace system ()? Do I have to call pipe ()? I'm a beginner programmer, so this low-level stuff is somewhat scary and foreign to me.

Hello Jen, hello IPC gurus :-)

I'm not a specialist in this area, and took the opportunity to play around a bit – someone will correct me where I'm wrong, explain more – and give an example using modules ;:-)

Re: Killing a process that takes too long

Re: Killing a process that takes too long

You find a modified script at the bottom, which, as I can see, does what you want.

I think your original script has several problems:

The SIGINT can be blocked, and it does not guarantee that the (any) child ends. The same holds for SIGTERM. The only signal that guarantees the child end is SIGKILL – but it's not recommended to use SIGKILL by default because it may hinder the killed process to proper clean up.

I'm not sure, but the best way would probably be to try SIGINT, wait, try SIGTERM if process still alive, wait, then SIGKILL if still alive.

But in your script(s), SIGINT works.

The code in the SIGHANDLER should be as short as possible (if this still holds true nowadays). That's why I modified it to only set the \$finish variable to true; it's value is checked elsewhere in the code, and a timeout() sub is called that does the actual work of killing the child.

The two statements after the eval block are executed by the parent *and* the child. You can test it by placing \$\$ (process pid) in the output of these two statements. That's why I placed an exit after the exec in the child process.

Your check for \$@ containing 'alarm' uses !~ instead of =~ :-)

After replacing 'sleep 45' (which creates one single child process) with "perl -e 'while (1) {print 1}' | tee test.txt", the fork leads to following processes (ps ax output snippet):

```
8827 pts/11 S+ 0:00 /usr/bin/perl ./script.pl
8828 pts/11 S+ 0:00 sh -c perl -e 'while (1) {print 1}' | tee test.txt
8829 pts/11 R+ 0:01 perl -e while (1) {print 1}
8830 pts/11 S+ 0:00 tee test.txt
```

After the timeout, pid 8828 (forked from 8827) is killed, but 8829 and 8830 (the grandchild processes) are still running.

So we need a way to kill several processes of the process group of the parent, but not the parent itself.

The way I found after consulting perldoc -f kill, man kill and perldoc perlipc is to

- a) in the parent: ignore the INT and TERM signal
- b) sending the signal to the whole process group (see timeout()).

I hope this helps a bit and is corrected if necessary

=====

Re: Killing a process that takes too long

```
#!/usr/bin/perl

use strict;
use warnings;

my $pid;
my $finish=0;

# actions after timeout to keep SIGHANDLER short
#
sub timeout {
print "Timed out pid $pid\n";

# kill the process group, but not the parent process
local $SIG{INT}='IGNORE';
local $SIG{TERM}='IGNORE';
kill 'INT' => -$$;

# eventually try also with TERM and KILL if necessary
die 'alarm';
}

eval {
local $SIG{ALRM} = sub { $finish=1 };

alarm 5;

die "Can't fork!" unless defined ($pid=fork); # check also this!

if ($pid) { # parent
warn "child pid: $pid\n";

# Here's the code that checks for the timeout and do the work:
while (1) {
$finish and timeout() and last;
sleep 1;
}

waitpid ($pid, 0);
}
else { # child
exec (q[perl -e 'while (1) {print 1}' > tee test.txt]);
exit; # the child shouldn't execute code hereafter
}

alarm 0;

};

warn "\$@=$@\n";
die "Timeout Exit\n" if $@ and $@ =~ /alarm/;
```

Re: Killing a process that takes too long

Re: Killing a process that takes too long

```
print "Exited normally.\n";
```

```
__END__
```

```
.
```