

Re: Timeout for connect()

Source: <http://unix.derkeiler.com/Newsgroups/comp.unix.programmer/2004-03/0035.html>

From: Jo (JoJoTwilligo_at_hotmail.com)

Date: 03/01/04

Date: 1 Mar 2004 09:52:26 -0800

"David Schwartz" <davids@webmaster.com> wrote in message news:<c1u4go\$6vc\$1@nntp.webmaster.com>...
> "Jo" <JoJoTwilligo@hotmail.com> wrote in message
> news:72ce7f0c.0402291555.493fd62e@posting.google.com...
> >> You can set the handler to SIG_IGN. The signal will be ignored, but
> >> it
> >> will cause 'connect' to return before the connection completes or fails.
> >> The
> >> connection will still be in process when 'connect' returns EINTR. You can
> >> call 'connect' later to find if it succeeded or fail.
> >
> > Oh yeah? Don't I run the risk of some kind of leak? Say, a
> > connection or socket leak?
>
> I'm not sure what you're talking about. You only run the risk of a leak
> if you don't close the things you open.

That's what I was thinking of. Is there any danger in closing a socket that hasn't been opened? If not, then I would think the proper solution would be to close the socket once you know that the connect is to be canceled.

>
> > I'm also a bit surprised at how simple it
> > would be to write non-blocking code. I avoided the solution that
> > involves select(), because it was too complicated for my
> > implementation. From what I'm hearing here, I could do something like
> > this:
> >
> > do {
> > alarm(1);
> > } while (connect(who, &cares, question_mark) == -1 && errno == EINTR);
> > alarm(0);
>
>
> Just don't forget to ignore SIG_ALARM. I think you have this wrong
> though. The second call to 'connect' will fail with EINPROGRESS.

comp.unix.programmer: Re: Timeout for connect()

Alright, how 'bout this then:

```
signal(SIGALARM,SIG_IGN);
do {
    alarm(1);
} while (connect(who, &cares, question_mark) == -1 && (errno==EINTR ||
errno==EINPROGRESS));
```

- > > *Each call to connect() here will go back to the first call's*
- > > *connection context and not create a new one? If that's so, why does*
- > > *the solution to a non-blocking connect() in the faq have that*
- > > *complicated code involving a select()?*
- >
- > *You have your code wrong. Your second call to 'connect' will fail*
- > *immediately because a connection is already in progress.*
- >
- > *Can you state your actual problem? Odds are the solution has nothing*
- > *whatsoever to do with connection timeouts. What's the problem you have? What*
- > *are you trying to do?*

I'm writing a daemon, so I wanted to be able to cancel the connect on SIGTERM, or at least know for how long the connect() would block so the daemon-shutdown routine would know how long to wait. I also want to put a cap on the timeout of the connect(). In either situation, I need to leave the connect() to see if it is supposed to cancel. If it is supposed to cancel, I need to free any allocated resources. But if it's not supposed to cancel, I need to monitor the status of the connect(). Everywhere else I've read recommends using a complicated select() routine. Why is this routine so simple?