

Re: on the semantics of connect(): EINTR, EALREADY, EINPROGRESS

Source: <http://unix.derkeiler.com/Newsgroups/comp.unix.programmer/2003-04/0964.html>

From: joe@invalid.address

Date: 04/22/03

From: joe@invalid.address

Date: Tue, 22 Apr 2003 19:36:39 GMT

david.madore@ens.fr (David Madore) writes:

- > *I have a question regarding the semantics of the Unix connect()*
- > *system call for blocking (stream) sockets and specifically what*
- > *happens when it is interrupted by a signal.*
- >
- > *Question: assume fd is a file descriptor referring to a blocking,*
- > *SOCK_STREAM socket, say in the PF_INET protocol family. A process*
- > *attempts a first call to connect(fd,&name,namelen) and this call is*
- > *interrupted by a signal, returning EINTR. In what state is fd then*
- > *left? In particular, what happens if connect(fd,&name,namelen) is*
- > *immediately re-attempted? I can imagine several possible behaviors:*

The standard behavior is defined at

<http://www.opengroup.org/onlinepubs/007904975/functions/connect.html>

"If the initiating socket is connection-mode, then connect() shall attempt to establish a connection to the address specified by the address argument. If the connection cannot be established immediately and O_NONBLOCK is not set for the file descriptor for the socket, connect() shall block for up to an unspecified timeout interval until the connection is established. If the timeout interval expires before the connection is established, connect() shall fail and the connection attempt shall be aborted. If connect() is interrupted by a signal that is caught while blocked waiting to establish a connection, connect() shall fail and set errno to [EINTR], but the connection request shall not be aborted, and the connection shall be established asynchronously."

[...]

"When the connection has been established asynchronously, select() and poll() shall indicate that the file descriptor for the socket is ready for writing."

comp.unix.programmer: Re: on the semantics of connect(): EINTR, EALREADY, EINPROGRESS

- > 1. the second call returns immediately with EALREADY, as if the
- > socket were non-blocking,
- >
- > 2. the second call blocks until connection is achieved (in which
- > case success is reported) or another signal is received (in which
- > case EINTR is returned again), or, of course, an error is produced,
- >
- > 3. the second call retries the connection from start, as if the fd had
- > been closed and another socket had been produced,
- >
- > 4. the second call returns with EISCONN (sort of stupid, but you never
- > know), or
- >
- > 5. demons fly through your nose (i.e., undefined behavior), or again
- >
- > 6. this simply can't happen because connect() is never interrupted by
- > a signal and never returns EINTR.
- >
- > What does the Norm say (by "the Norm", I mean The Open Group's Single
- > Unix Specification, version 3, aka SUSv3)? If I read it to the
- > letter, it prescribes behavior 1 above: indeed, it says "If connect()
- > is interrupted by a signal that is caught while blocked waiting to
- > establish a connection, connect() shall fail and set errno to [EINTR],
- > but the connection request shall not be aborted, and the connection
- > shall be established asynchronously.

I think you just missed the second quote above. Put the descriptor in the write mask for select() and then test it when it comes up ready for writing. Since it's then essentially a non-blocking connect, see also

<http://www.muq.org/~cynbe/ref/nonblocking-connects.html>

<http://cr.yp.to/docs/connect.html>

- > [...] The connect() function
- > shall fail if: [...] [EALREADY] A connection request is already in
- > progress for the specified socket." There is nothing saying that
- > EALREADY is used only for non-blocking sockets, so the second call
- > should fail with EALREADY if the first one returned EINTR, starting an
- > asynchronous connection attempt: this is behavior 1 in my list above.

As I read it, the situation described turns this into a non-blocking connection attempt, even though the socket isn't in non-blocking mode ("the connection shall be established asynchronously").

Of course, the system man pages are probably a better source of what any particular implementation actually does.