

advices on sockets

Source: <http://unix.derkeiler.com/Newsgroups/comp.unix.programmer/2008-03/msg00191.html>

- *From:* mast4as <mast4as@xxxxxxxxxx>
 - *Date:* Thu, 13 Mar 2008 08:32:59 -0700 (PDT)
-

Hi everyone,

Yes I spend quite some time (2 days) reading tutorials, man pages on sockets and related topics but still can't seem to find the proper way of doing what I need. It's not so much that it's impossible, it's just that I don't have the knowledge and seek over people's experiences on this topic ;-)

The little app I need to write is a x-window to which a graphic program sends an image. The graphic program renders a frame and send it to the x-window in tiles (blocks of pixels). I ended up deciding to implement that with a server (x-window) -client (graphic app) type of model. The reason is that eventually several "instances" of the graphic program can run on the computer at the same time and the images that each "instance" works on, needs to be send to the same window.

The other condition is that the x-window stays active even though the graphic app has finished processing the frame. The user can therefore continue manipulating the frame in the x-window (zooming, panning, etc...)

Now I must say I am trying to prototype things here, some I just want to find a simple, elegant and robust way to do that. I am not a socket expert (obviously), have very modest programming skills in comparison to you all... So I am just really asking for a bit of supervision here, with something I can't seem to find any answers for in posts tutorials on the web. So please be kind ;-)

I have many question regarding this problem...

1/ on the client side (the graphic app), I'd like to start to the server (the x-window) if it's not running yet. This yields 2 sub-questions:

1.1 what's the best way to find out that the server is not running ?

Is that fact

that a call to connect() fails is enough ?

1.2 is there a way I can start my server/x-window code by another mean that

advices on sockets

calling the function `exec()` ? I am asking that question because it means the code for the server has to be compiled as a separate app. Can I avoid that?
Can I write the code for the x-window/server with the code of the client and start the x-window from within the graphics app as a *separate process* that won't die when the graphics app has finished processing the image ?

2/ my other problem (thing I am not clear about) is the way I am passing the data from the client (graphics app) and the server (x-window). I have also 2 questions for this.

2.1 ok i read in the docs that by default, `accept()` is blocking. Remember that I wrote at the beginning of the post that I want the x-window app to run as an independent process from the graphics app and that the user can manipulate the frame in the x-window using shortcuts for example (z = zoom in, Z = zoom out, or 'q' to quit the x-window program). So the x-display/server program needs to check for both new data coming through the socket but also X events. If I use a loop to check for new connections from clients, because `accept()` is blocking, it stops me from checking X-events.

```
// this for example doesn't work because accept is blocking
while(1) {
checkXEvent(&c);
if (c=='q')
quit = 1;
if (accept(sockfd, ...) == -1)
continue;
// we have an incoming connection with a client, proceed..., fork
to create a
// a process child and treat packed of incoming data
...
}
```

so my question here is the following. Is the only way to do this is by making the socket non-blocking ? But in that case, isn't the fact that having a non-blocking socket uses a lot of unnecessary CPU resources (because it never stops checking if there's an incoming connection).

I played with `select()` (commented part of my code) but it didn't

advices on sockets

work. In particular my understanding is that if I use select() is can't use the fork technique anymore. In other words if 2 instances of the graphics app processing 2 images are running at the same time, the file descriptor set will contain 2 file descriptors (one for each connection it has had from the 2 running clients). Meaning if the 2 clients send data to the sever at the same time, I will have to read the data from each client and do i need to do with that data sent. Is that a better way than fork ?

2.2 to pass the data i didn't find anything better that coming up with a stupid type of protocol. I send a request to the server first that tells it what it should expect to read next (some info about the size of the incoming tile, or the RGB data for the tile). In the little prototype I worked on it seems to work but I am not sure it's reliable ? Is it how you would do it ? Is there a better way ?

I put the code online with you have the patience and the kindness to have a look

<http://www.scratchapixel.com/docs/proto.cpp> // sim graphics app
<http://www.scratchapixel.com/docs/server.cpp> // sim server/x-window

I really apologize for that long post, but hopefully it will have some interesting answers and will help me and other people in the future, as a part from basic client-server code examples that deals with buff[1024], I didn't find anything else out there that answered clearly those questions.

Thank you so much -coralie

.