

Hint 2 for Project 2.

Here are some things you might consider regarding project 2.

First, should you invoke your server program first or the telnet (ssh) client? The server should be invoked first. It needs to execute to the `accept` statement and then block. Then you invoke `telnet (ssh)` which executes a `connect`. Immediately after the `connect`, the client executes a network read expecting a command prompt to be sent from the server. Your server program should execute a network write to accommodate this request.

Then the client should block on a read from the keyboard expecting you to type in a string. When you type in the string, the client executes a network write, sending the string to the server.

The server executes a network read to read the string and then a network write sending the string back to the client.

The client executes a network read to read the string and then a write to the screen, which in effect echo's the string the client originally typed in.

Your server program should execute the `socket`, `bind`, `listen`, and `accept` system calls. The `accept` should be inside a loop, and be followed by a `pthread_create`. The function invoked by `pthread_create` should include the reads and writes specified above. Including `pthread_create` in the loop causes threads to be dynamically created each time the server gets past `accept`.

You undoubtedly are concerned that the reads and writes may go to or come from multiple clients at the same time. For example, it would be a problem if the string from one client were echoed at on the screen of several other clients. Keep in mind that information is only transferred when the client (or server) executes a read (or write) and the other executes a write (or read). You should insert semaphores as appropriate to ensure that the right thread is doing network reads (or writes). Be careful to block threads only where it is necessary.