CSE 325 **Spring 2021**

Computer Project #12

Assignment Overview

This assignment focuses on network programming with sockets. You will design and implement additional functionality to extend the previous project, as described below.

It is worth 45 points (4.5% of course grade) and must be completed no later than 11:59 PM on Wednesday, 4/21.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj12.client.makefile` – the makefile which produces **`proj12.client`**
`proj12.client.cpp` – the source code file for your client process
`proj12.server.makefile` – the makefile which produces **`proj12.server`**
`proj12.server.cpp` – the source code file for your server process

Be sure to use the specified file names and submit them for grading via the CSE Handin system before the project deadline.

Assignment Specifications

The server process will interact with a client process to provide a rudimentary mechanism for copying a text file from one system to another.

1. There are no required command-line arguments when the server process is executed. For example:

```
proj12.server
```

The server process will create a socket, bind it to a port, and listen for a connection request on that port. It will display the host name and port number on the standard output stream. For example:

```
arctic.cse.msu.edu 54321
```

The server process will accept a connection request from the client process and wait for it to send a file name.

If the server process is able to open the specified file name as an input file, it will send the four-character message OPEN to the client process and wait for the client process to respond. Otherwise, the server process will send the six-character message FAILED to the client process and terminate execution.

If the client process responds with the four-character message SEND, the server process will send the contents of the input file to the client process and then terminate execution.

2. The client process will accept three command-line arguments, in the following order: the host name, the port number being used by the server process, and the name of the desired file. For example:

```
proj12.client arctic.cse.msu.edu 54321 /user/cse325/Examples/example01
```

The client process will connect to the server process, send it the name of the desired file, and wait for a response.

If the server process responds with the four-character message OPEN, the client process will respond with the four-character message SEND and wait for the server process to send it the contents of the desired file. The client process will display the contents of the desired file on the standard output stream and then terminate execution.

3. The server and client processes will perform appropriate error-handling.

Assignment Notes

1. As stated above, your source code files must be named "proj12.server.cpp" and "proj12.client.cpp", and you must use "g++" to translate your source code files in the CSE Linux environment.

2. Your processes are required to use the system calls "send()" and "recv()" to exchange messages between the client process and the server process.

3. Your server process is required to use the system call "read()" to access the contents of the desired file.

4. Your client process is required to use the system call "write()" to display the contents of the desired file on the standard output stream (file descriptor 1).

5. If your server process produces any additional output (beyond the host name and port number), it must send that additional output to the standard error stream (file descriptor 2 or "cerr").

If your client process produces any additional output (beyond the contents of the desired file), it must send that additional output to the standard error stream (file descriptor 2 or "cerr").

If you wish, you may use optional command-line arguments for the server and/or client to control the display of additional information which might be useful for debugging.

6. Your client process must use a buffer of size 64 bytes to receive the contents of the desired file. For most files, it will require multiple calls to "recv()" and "write()" to process the entire file.

7. Information about some of the system calls you will use is available in section 2 of the "man" utility:

```
man 2 socket
man 2 bind
man 2 listen
man 2 accept
man 2 connect
man 2 send
man 2 recv
man 2 write
man 2 read
```

8. Be sure to test your solution in several different ways. For example, execute your server process and your client process on the same CSE server, then execute the two processes on two different CSE servers.

You should open two terminal windows: one to execute the server process and one to execute the client process.

Note that you must execute the server process before executing the client process.