Common Mistakes in Project 3

- [Project 3a] gcc was not installed in the VM. Requested students to install gcc using command `sudo apt-get install gcc`

- [Project 3a] Messages sent were not null-terminated which made auto-grader fail as the message was incorrect.

- [Project 3a] Sending an extra character from client to server and vice versa and this fails the auto-grader which listens in between your server and the client and expects the required format.

- [Project 3a] Command-line arguments: make sure you test using the specific commands as shown in TA demo video to run the code. Pay extra attention to types and order of arguments used when invoking the command. Additionally, make sure that you parse the arguments correctly and use them. An example error: *"My issue was that my tcpserver.c was requiring a host IP argument on the command line (like server.c) but for tcpserver.c, the instructions say that the argument should just be a port (as in the TA demo). So the host address is actually just supposed to default to localhost IP. When I made this fix, my submission worked."*

- [Project 3a] Message format: ensure that the message format while you communicate is "HELLO X" as specified in the project handout. If you send only a number "X" and print "HELLO" + "X", your test will fail.

- [Project 3a] Closing the server after 1 iteration of the test is finished: you need to make sure that your server does not crash/exit after the first test is run. It should continue to accept more connections.

- [Project 3a] Clearing the read buffer: make sure that you clear the buffer before reading from the socket. This ensures that your program does not read any junk values and you receive the expected message.

- [Project 3a] Memory initialization: when you allocate memory, make sure you initialize the memory, using memset for example, before you use it.

- [Project 3a] Wrong types: When working with signed and unsigned types, as you perform the variable assignment, make sure you understand the type of the assigned value and the type of the variable and their conversion. Particularly, in your code, if you assigned signed values to unsigned variables, this can trigger errors in the auto-grader under high loads.

- [Project 3a] In cases when the Makefile does not interact well with autograder, add `all: tcpclient tcpserver` above the tcpclient target line in the Makefile.

- [Project 3a] It cannot handle large inputs. Buffer is set too small so numbers above 4 or 5 digits do not work.

- [Project 3a] tcp.sh permissions denied. Run `chmod u+x ./tcp.sh`

- [Project 3a] The python script for concurrent client requests does not run. The main reason is passing in wrong or missing arguments. Read the Python script carefully to figure out what parameters to pass in.

- [Project 3a] If you use fputs and pass a string that does not have a newline character at the end, it will simply buffer the string to print but not actually print it. You must append a newline character to your string or make an additional call to fputs passing in a newline character if you want the original string to be printed. The problem may happen if you use printf as well.

- [Project 3a] The “Hello X” message sent from client to server should contain the original X passed into the client as argument. The autograder intercepts and checks the number. For example, when the client passes X as an argument, do NOT send “Hello [X-1]” to the server and then let the server print out (X-1)+1. It will look correct on your terminal but it will not be accepted by the autograder.

- [Project 3b] Closing the listening socket. The listening socket should be reused across different connections. Each successful connection is then assigned a temporary socket.

- [Project 3b] Closing the new socket from accept() as soon as you create the thread.

- [Project 3b] Not creating an array for the pids for multiple threads. Passing the same socket variable to all threads which fail auto-grader in under load. If the socket fd variable is being passed as a reference to the thread function and in load conditions, the value of the variable changes when a new connection is accepted, changing the old value of the socket fd. So, one solution can be to make an array of active connections fd values, maintaining different socket fd values for different connections.

- [Project 3b] Type casting error in pthread.

- [Project 3b] How to pass multiple arguments to worker thread via pthread.
    - Pass a pointer to an array or struct

- [Project 3b] Misunderstanding of client behaviors.
    - Clients should send messages twice: Hello X and Hello X+2.

- [Project 3b] “simplex-talk: bind: Cannot assign requested address" error. You can check the socket man page to know more about SO_REUSEPORT. To use this flag you need to call setsockopt(2) after you create your socket. Alternatively, on the same lines, you can also use SO_LINGER with a 0 timeout value. check this page with instructions on the order to use socket functions and setsockopt. https://www.geeksforgeeks.org/socket-programming-cc/.

- [Project 3b] Similar but wrong symbols in the commands for compiling, for example:
    - Using “gcc –o” instead of “gcc -o” fails gcc to compile.

- [Project 3b] Not using `concurrent-requests.py` to test concurrent requests.

- [Project 3b] Pass the same data to all threads instead of allocating memory for each thread. Problems happen when one thread exists and releases the memory.

- [Project 3b] The number of arguments does not match what the writeup specifies, for example, the server incorrectly expects 2 parameters while the auto-grader only provides one.

- [Project 3b] `gcc -o multi-tcpserver multi-tcpserver.c -l pthread` works but be careful of cut-and-paste from PDF file. Quoting a student: *“When I copy & pasted that from the PDF I also received compile errors, but when I typed it out, it worked great. Seems there's some extra character or something that shows up when you copy & paste from the PDF (on Mac at least).”*

- [Project 3b] Errors in Makefile.
  Example mistake:

```
all: tcpclient tcpserver
        gcc -o tcpclient tcpclient.c
        gcc -o multi-tcpserver multi-tcpserver.c -l pthread
clean:
        rm -f *.o tcpclient tcpserver
```

  Correction:

```
all: tcpclient multi-tcpserver

tcpclient: tcpclient.c
	gcc -o tcpclient tcpclient.c

multi-tcpserver: multi-tcpserver.c
	gcc -pthread -o multi-tcpserver multi-tcpserver.c

clean:
	rm -f *.o tcpclient multi-tcpserver
```

- [Project 3c] Not setting server socket to NON_BLOCKING.
- [Project 3c] Not reset FD_set after select.
- [Project 3c] Mistakes in checking client sockets.
    - Instead of looping through client sockets, students loop through a range of socket id, this is causing SEGFAULT problems.
- [Project 3c] Reset Buffer. Some students mistook the size of buf pointer as the third argument to memset, which should have been the actual buffer size.
- [Project 3c] File descriptors to monitor are not reset by FD_ZERO() at the beginning of each iteration.
- [Project 3c] Trying to read on a stale file descriptor that should have been cleared.
- [Project 3c] Not clearing buffers with memset() before or after use.
- [Project 3c] Using small MAX_PENDING in listen().
- [Project 3c] Not adding all file descriptors to monitor in the fd_set that is to be checked by select().
- [Project 3c] Not using the same port between the server and the client.
- [Project 3c] Making mistakes listed in project 3a and 3b. We had cases when students did not clear buffers, etc. Make sure you read the entire document again to make sure you did not make the same mistakes.