Ayan Abbas
1630145

CMPUT 379
Assignment 3

OBJECTIVES

The objective of this assignment was to gain experience with developing system programs that utilize TCP sockets for communication, and I/O multiplexing using poll or select for non-blocking I/O. In this assignment, we had to develop a C program, a3w23, which implements the functionality of the client-server system (modified version of assignment 2). The communication was done using sockets following the TCP protocol. There can be maximum of three clients and one server. The clients cannot communicate with each other and share the same input file. The message is sent using packets as described in the assignment description. Further implementation details is given below.

DESIGN OVERVIEW

- The program can be invoked as a client and as a server. To invoke the program as a server use "./a3w23 -s portNum" and to invoke the program as a client use "a3w23 -c idNumber inputFile serverAddress portNumber" where idNumber can be 1, 2 or 3; inputFile is a3-ex1.dat; serverAddress is the IP address in the form of symbolic name; portNum is 9145 where 145 is the last three digits of my student ID to avoid conflict. I have tried to check for invalid inputs but all the cases may not be covered so follow the above.
- The input file was parsed and the client IDs, Packet type, and object names were extracted using stringstream. The empty lines and commented lines are skipped and I have assumed a single space between each of the input. The input file should be in the format of the assignment description because I have not dealt with invalid lines in the input file. The client id matching the id entered while invoking the program are parsed. The other lines are ignored.
- To implement the packets, I have used a struct with a message and a type. The type was implemented using enum and the message is a union of all types of message which can be encountered. I have created separate functions to compose these messages and separate functions to send and receive the packets. Most of the packet types have a message though some of them are empty.
- For the client loop, I have created a socket and connected the socket. After that, I parsed through the input file and sent and received the required packets depending on the line. The information was also printed as I received or sent the packets.
- For the server loop, I created the socket, then I used bind to turn it into a managing socket. And I used listen. Then I used poll function to handle I/O from the keyboard and the sockets in a non-blocking manner. The user can enter list and quit from the keyboard. The poll function also handles the clients and the packages which are received. The server also detects any closed connections as well.

- The executable for the program can be made using the Makefile. Use “make a3w23” to make the executable of the program and run the client-server using the correct command for each of them. Run the server in a separate terminal. Then run the clients in a different terminal.
- The IP address should be the same as the lab machine which you are connected to.

Makefile

- “make all” creates the executable of all the files
- “make a3w23” creates the executable for a3w23.cpp
- “make clean” removed all the executables and object files
- “make tar” compresses the desired files

PROJECT STATUS

The project is incomplete and produces incorrect results. I have tried to create packets in form of structs and use read/write to receive/send the packets and the input file is parsed correctly. But for some reason, I do not get the desired output. The program sends the packet to the server correctly, but the packet received from the client has zero length. I believe there is something wrong with the poll function in handling the incoming packets from the client. The input from the keyboard is processed correctly. But after hours of debugging, I was not able to fix the issue. The program just stops when I invoke the client and when I use quit in the server terminal, all the output from the client can be seen. The packets are transmitted correctly from the client but the received packets have zero length and the server does not show any output. There is no output in the client but when quit is typed in the server terminal, the transferred packets can be seen. I believe there is an issue with the poll indexing but due to time constraint I was not able to fix this issue. The logic of my code is correct but there is some slight mistake in the implementation. All the packets are implemented correctly, the poll input from the keyboard is functional, the input file is parsed correctly, and all the error checks are done. After the marking, I would love to go through my code and find out the error which I am facing with the TA who marks the assignment. During the marking, kindly go through the code as the output is not produced in the server and the output in the client is produced only when quit is typed in the server terminal.

TESTING AND RESULTS

The program is incomplete/producing incorrect output, so I was not able to test it thoroughly. I tested the small functionalities separately. I tried to print out debug messages to ensure the input file is parsed correctly. I tried to test the socket functions by creating another small client server program with one client and tried to send a string message and that worked as well. The poll function takes in input from the keyboard as well. The program exits on quit and though the list of objects are not able to be implemented on typing list but I print a message to indicate that there is correct input received from the keyboard. The delay and the quit commands work correctly as there is a delay produced and the program exits when quit line is reached. Overall,

the complete functionality was not tested since the program is producing undesired results, but I tried to test each functionality separately to ensure correct behaviour.

ACKNOWLEDGEMENT

1. Poll program demonstrated in the lectures by Ehab Elmallah.
2. Concurrent Client-server program demonstrated in the class by Ehab Elmallah.
3. sockMsg.cc program on eClass by Ehab Elmallah
4. http://webdocs.cs.ualberta.ca/~cmput379/W23/379only/sockMsg.cc
5. https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
6. https://linux.die.net/man/2/setsockopt
7. https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-bind-bind-name-socket
8. https://linuxhint.com/use-poll-system-call-c/
9. https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/
10. https://www.tutorialspoint.com/how-to-use-enums-in-cplusplus
11. Linux Man Pages
12. Advanced Programming in Unix Environment