**Final Project Report**

Course: Systems Programming

Instructor: Shabbir Mukhi

Zubair Irfan Khan – 14782

Date: 9th May 2020

**Help**

Run client program as ./client <ipaddress> <portnumber>

Client-side inputs:

Addition:  
add <operand1> <operand2> ….. <operand n>

Subtraction:  
sub <operand1> <operand2> ….. <operand n>

Multiplication:  
mul <operand1> <operand2> ….. <operand n>

Division:  
div <operand1> <operand2> ….. <operand n>

Run:  
run <arg1> <arg2> ….. <arg n>

Kill:  
kill <pid>

List:  
list

Exit:  
exit

Print(on server):  
print <message>

Server-side inputs

Print(on client):  
for all clients : print 0 <message>  
for specific client: print <client\_id> <message>   
note: client\_id is a unique numeric id assigned to every client upon connection starting from 1

List:  
for list of all clients: list 0  
for list of specific client: list <client\_id>

**Description**

Initially there is a single process running on the server which is the Connection handler. It has 2 threads. The main thread creates the socket, binds, listens and accepts connections and also maintains the list of all connections. The second thread, we will call it the input thread, takes input from user. The inputs are either to print a message on all connected clients or a specific client, or to view list of processes running under client handler (explained in next line) for every client, or a specific client. When a client requests a connection and is accepted, a child process under connection handler is created, called Client handler. This process has 2 thread. The main threads communicates with the client via a socket by accepting input from it, processing it(add, sub, run, list etc.) and returning back the answer, and confirmation and error messages. The second thread talks to the input thread via pipes in the connection handler to accept user commands from server and process them, and either return list of its child processes, or print a message on a client.

The process at the client has 2 threads. The main thread takes input from the user, and sends it to the client handler via a socket, while the second thread reads messages from the client handler and prints it to the screen for the user. The second thread also checks whether the client handler is closing or not (identified by the exit flag, 0 or 1, embedded at the start of every message the client handler is sending. This is useful for when the client wants to exit and disconnect, the client handler terminates all its running processes, and before terminating itself, sends a confirmation message to the client acknowledging termination).

The connection handler maintains a list of all active connections using a Linked list. Linked list was chosen because disconnected connections could be easily removed, and there was no limit to the number of connections that could connect.

The client handler maintains a list of all processes that the user asks it to run using an array. A user can run maximum 300 processes (including active and terminated processes). I believe linked list would be a better approach for this as well, but since I had implemented this list using arrays in previous assignments, I didn’t get the time to convert this to a linked list.

**Limitations:**

Signal handling not completed on the server end due to running out of time for when the connection handler or client handler is terminated externally. Upon terminating connection handler, all sub-processes of client handlers get terminated, client process terminates but not gracefully. I could’ve done it using return 0 from read but time finished.

Maximum 300 processes (active and terminated) can be run on the client handler by a single user. While this is a restriction, I think when it comes to restricting users to run only limited amount of processes to better utilize resources, it makes sense. However, the limit does not only include active processes, but also terminated processes. Again, could’ve been solved using Linked List.

When a process name is longer than 19 characters (list can contain names up to 19 characters), there is an issue in printing its name and status. Must be a silly error in padding strings.