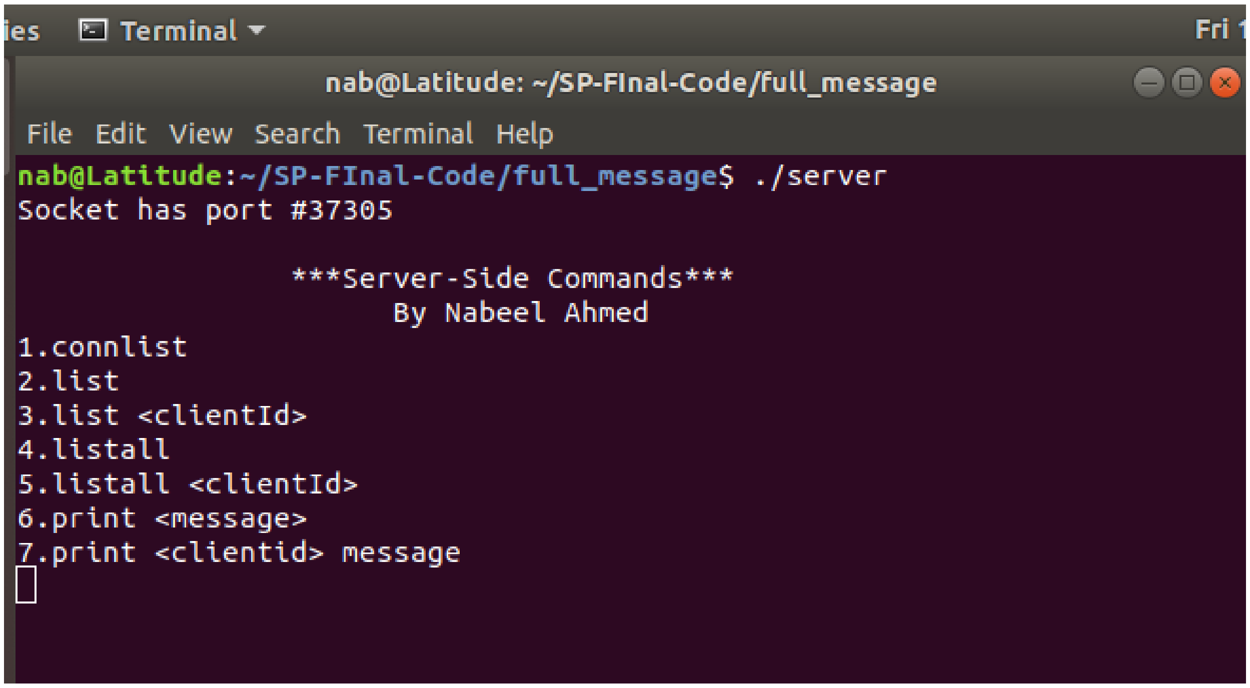
SP-Project Report

Nabeel Ahmed – 18024

Server-Side Commands:


ies
Terminal
Fri
nab@Latitude: ~/SP-FInal-Code/full_message
File Edit View Search Terminal Help

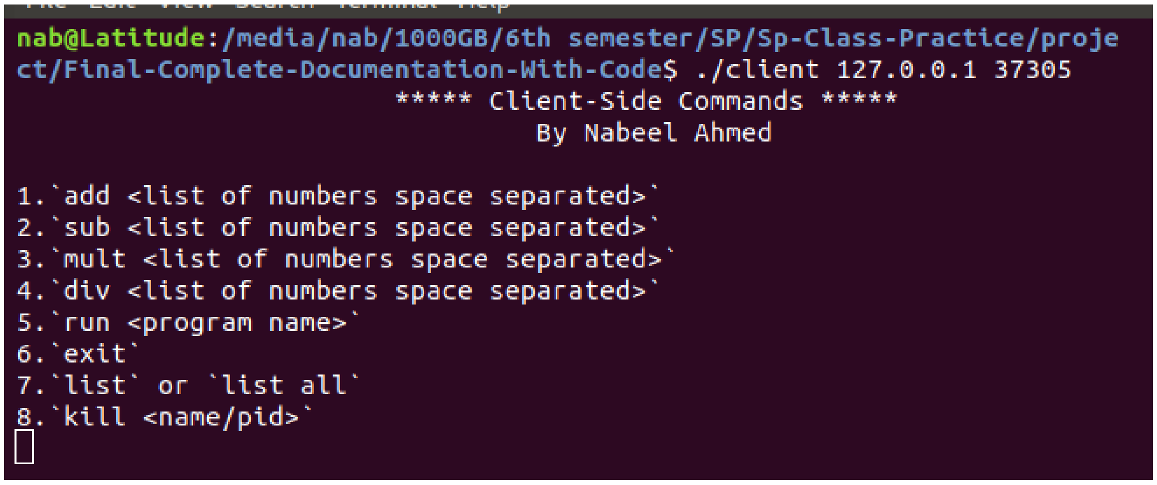
```
nab@Latitude:~/SP-FInal-Code/full_message$ ./server
Socket has port #37305

                ***Server-Side Commands***
                      By Nabeel Ahmed
1.connlist
2.list
3.list <clientId>
4.listall
5.listall <clientId>
6.print <message>
7.print <clientid> message
```

On server-Side, I have implemented 7 commands which are taken from user which are as follows:

1. **`connlist`**: It displays list of all active connections i.e., active client Handlers. The list gets updated once any client handler terminates.
2. **`list`**: It displays list of all **active processes** of all **active client Handlers.**
3. **`list` <ClientID>:** If after list command client id is also passed it will return active processes of that particular client only iff the client id is valid. Example: **list <c0>**. I have assigned every client Handler a unique identifier like c0, c1...etc.
4. **BONUS: `listall`:** It displays list of all **processes (active and inactive both)** of all **active client Handlers.**
5. **BONUS: `listall` <ClientID>:** If after **`listall`** command client id is also passed it will return all processes of that particular client only iff the client id is valid. Example: **`listall <c0>`**. I have assigned every client Handler a unique identifier like c0, c1...etc. Client ID is validated every time.
6. **`print <message>`:** This command will send the message to all active client Handlers whose connection hasn't been terminated yet.

7. **`print <ClientID> <message> `:** This command will send the message to only that client whose id we have specified provided the id is valid and that client Handler is also active.

Client-Side Commands:

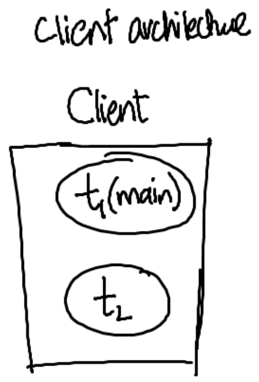


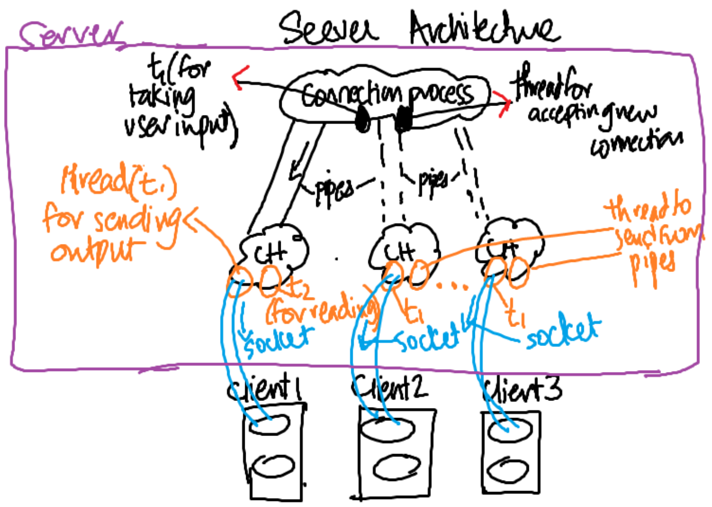
```
nab@Latitude:/media/nab/1000GB/6th semester/SP/Sp-Class-Practice/proje
ct/Final-Complete-Documentation-With-Code$ ./client 127.0.0.1 37305
                         ***** Client-Side Commands *****
                                  By Nabeel Ahmed

1.`add <list of numbers space separated>`
2.`sub <list of numbers space separated>`
3.`mult <list of numbers space separated>`
4.`div <list of numbers space separated>`
5.`run <program name>`
6.`exit`
7.`list` or `list all`
8.`kill <name/pid>`
```

On client-side, I have implemented the following commands:

1. **`add <list of int numbers>`:** This command accepts list of integer numbers space separated to perform additions, e.g., add 3 4 45 5 will return 57.
2. **`sub <list of int numbers> `:** This command accepts list of integer numbers space separated to perform subtraction e.g., sub 3 4 will return -1.
3. **`mult <list of int numbers>`:** This command accepts list of integer numbers space separated to perform multiplication e.g., mult 3 4 will return 12.
4. **`div <list of int numbers>`:** This command accepts list of integer numbers space separated to perform division e.g., mult 3 4 will return 0.75.
5. **`run <program exe name>`:** This command will run the executable passed in iff the executable exist and is valid.
6. **`exit`:** This command will gracefully terminate client and all its spawned processes.
7. **`list` or `list all`:** If only list token is passed it will print only active processes list, however if `all` token after list is found it will print all active and inactive processes.
8. **`kill <name/pid>`:** This command will kill the already running process you have created if it is still running. It can kill the process by name or if you Wanna kill specific instance of that process you can kill by its pid.

Client-Server Architecture Details:





Client – Side:

- On client-side I have implemented threads in order to provide parallelism so that my program doesn't get freeze when one of the reads get blocked. This will allow simultaneously to take command from user from one thread and display output from server on another thread. Multi-threaded Client also allow us to send multiple commands at a time without waiting for server to respond to one command and then we deliver next command.

Server – Side:

- On server side, I have a connection process in one thread while in other thread I take input command from user. The connection process does nothing but accepts and establishes new connection and maintain its entry in connection list.
  The other thread just takes commands from the user and process it accordingly.
- In client Handler also we create threads in order to provide parallelism to our program and to prevent our program from freezing otherwise we might be either stuck on read from pipe or on read from socket. Therefore, I made one thread for read from pipe and displaying output on server while other thread communicates with client and read commands sent from client. This way we can perform multiple tasks without getting block on any read.
- Another architecture level detail includes that I have initialized and created pipes for Inter process communication (IPC) between client handler and connection process. Every client handler has its own 2-way pipes with thread in connection process. Thereby having different communications channels for every Client handler. This allows to take input from server in its thread, then sent it via pipe to Client Handler and finally writing it to their respective socket.

Limitations of my Project:

- The very obvious limitation of my project so far is although I am sending multiple commands from client using multithreaded architecture on client side, the server still process one command at a time which is exactly one of the limitations to my project that server can't process multiple commands at a time since it is not designed to be multithreaded.
- My project can only process integer numbers for arithmetic operations, it cannot process fractional or decimal numbers.

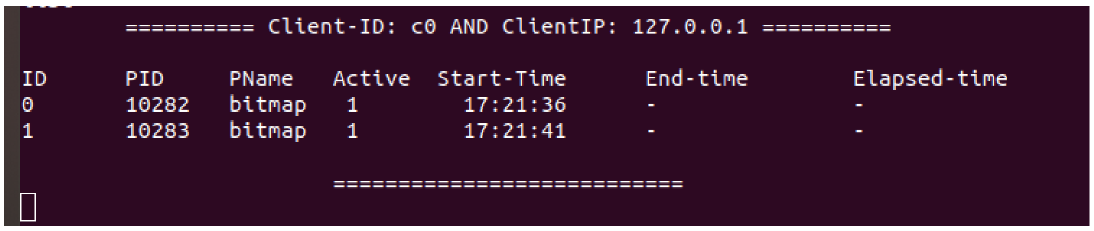
Show-Off-Area

- I have implemented my commands to be case sensitive that is it won't matter if you type the command in all capitalize format or if any of the letter gets capitalized, it still will function fine as expected. Eg. Add 3 4, add 3 4, AdD 3 4 are all valid commands.
- I have made intense error checking across my entire project. In order to get over the overhead of writing 3 lines error check every time, I made separate functions for checking of read, write and sprintf error. This allows me to just call that function every time with ret variable in just a single line.
- Moreover, although we were asked to just implement arithmetic operations for just +ve integer numbers. I did bit of logic building and by adding few lines of code and checks I made same code work for negative values while ensuring it does not break on any edge cases.
- Furthermore, good printing and displaying relevant statements is very essential part of your program so that the user can know exactly what his commands actually, so I made sure my every command is followed by message to show interactivity and if any invalid

command is given the relevant message is displayed. In addition to this in order to avoid any confusion of what commands work I follow a practice of adding list of command my program allows at the very start as my program runs. This ensures less chances of user entering invalid command.

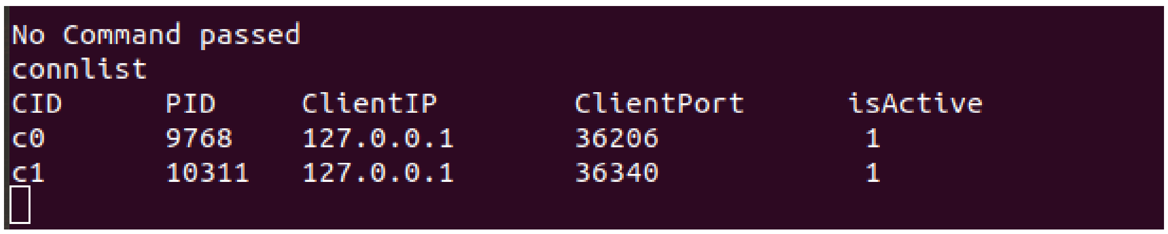
- The printing of processes and connection table was one of the challenging tasks I found. Since I was concatenating all my process in a large char array so I eliminated the chance of any discrepancies I might have in my list being sent to client or connection process for that matter. However, printing in table form required bit of my skills to organizes in well-structured which I thought was worth mentioning in show off area.
- Finally as we all know in a production level environment logging is very important to detect errors as without that it's very difficult to find source of problem in 100,000 lines of code. So in my project code what I did was I logged all the command which are entered from server side, so if my code crash on any sequence of command, I would know how to replicate the error and resolve it.

Process List Table Screenshot



```
        ========== Client-ID: c0 AND ClientIP: 127.0.0.1 ==========

ID      PID     PName   Active  Start-Time      End-time        Elapsed-time
0       10282   bitmap   1        17:21:36      -               -
1       10283   bitmap   1        17:21:41      -               -

                        ===========================
```

Connlist Table



```
No Command passed
connlist
CID       PID     ClientIP        ClientPort      isActive
c0        9768    127.0.0.1       36206            1
c1        10311   127.0.0.1       36340            1
```