OS Lab-3

Mruganshi Gohel
B20CS014

Files:

server1.c – attends one client at a time and blocks other clients trying to connect
server2.c – uses forks to attend multiple clients at a time
server3.c – uses threads to attend multiple clients at a time
client.c – client program

How to run?

```
gcc server1.c -o s1 followed by ./s1 5555
gcc server2.c -o s2 followed by ./s2
gcc server3.c -o s3 -lpthread followed by ./s3
gcc client.c -o c followed by ./c 127.0.0.1 5555
```

Note :

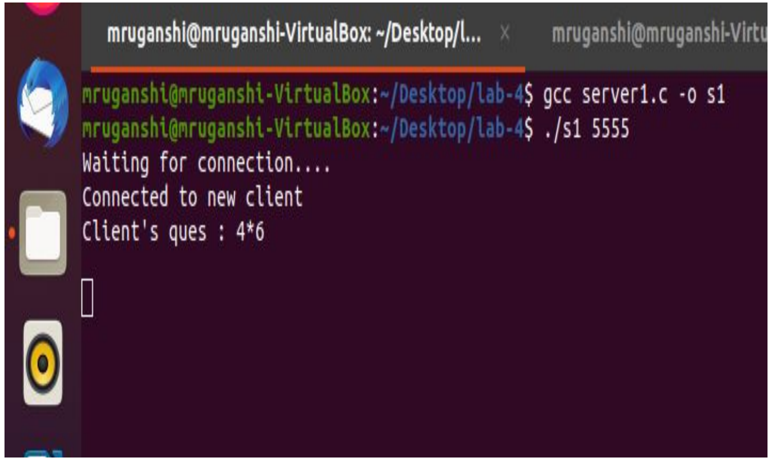
- You have to make sure to specify the port number in server1, address and port in client.
- For server3 while compiling its necessary to write -lpthread because pthread.h library is used

1 to 1 Server client communication

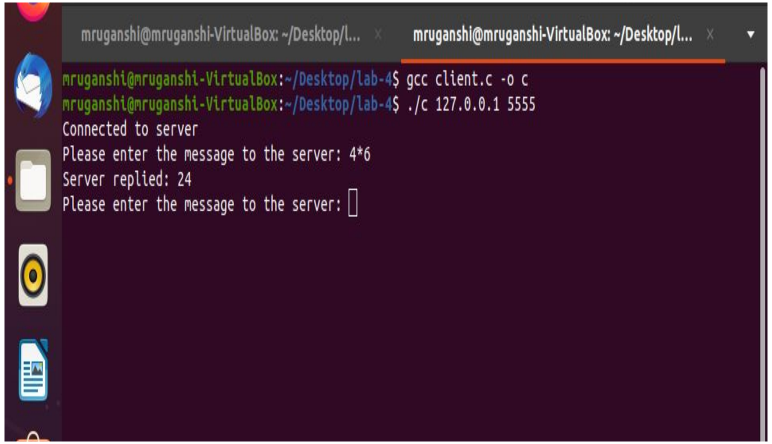
Functions used for basic sockets:

- **#include<sys/socket.h>** – Library used for implementation
- **sockaddr_in struct** – for storing client and server informations
- **socket()** – for creating a socket
- **bind()** – binding the socket to address and port
- **listen()** – to make client listen to socket
- **Strchr** – used for identifying operator
- **Strtok** – used for getting integer tokens

Later on calculated answer is parsed into buffer's data type by sprintf.



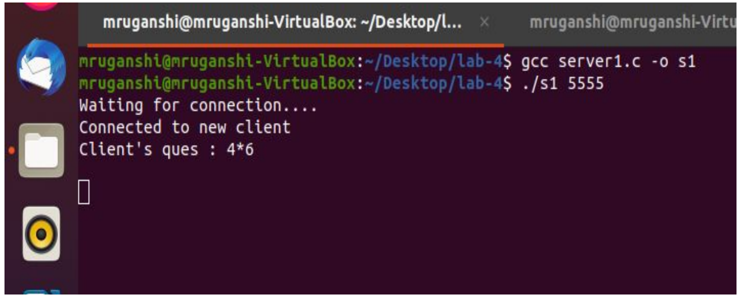
```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ gcc server1.c -o s1
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./s1 5555
Waiting for connection....
Connected to new client
Client's ques : 4*6
```



```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ gcc client.c -o c
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./c 127.0.0.1 5555
Connected to server
Please enter the message to the server: 4*6
Server replied: 24
Please enter the message to the server:
```

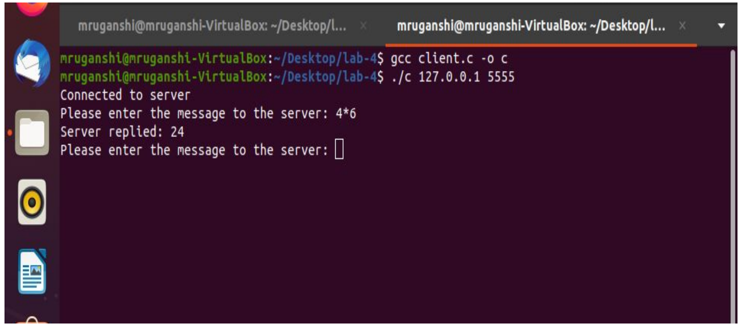
Server-1 Outputs

- The other client can’t connect as server is busy.
- For this feature , we close the listen fd as soon as a client is connected to let other clients know that server is busy and hence they cannot connect



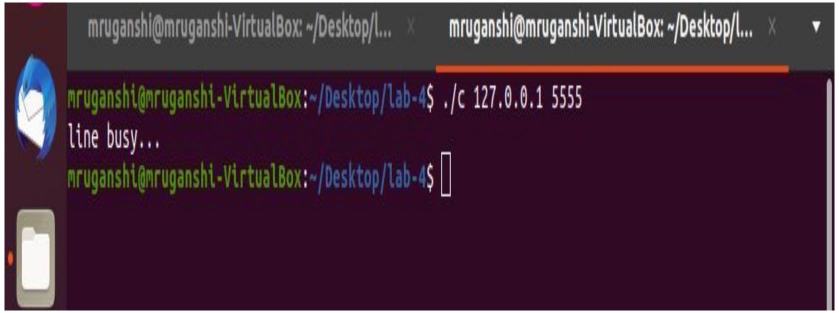
mruganshi@mruganshi-VirtualBox: ~/Desktop/l...
mruganshi@mruganshi-Virtu

```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ gcc server1.c -o s1
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./s1 5555
Waiting for connection....
Connected to new client
Client's ques : 4*6
```



mruganshi@mruganshi-VirtualBox: ~/Desktop/l...
mruganshi@mruganshi-VirtualBox: ~/Desktop/l...

```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ gcc client.c -o c
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./c 127.0.0.1 5555
Connected to server
Please enter the message to the server: 4*6
Server replied: 24
Please enter the message to the server:
```

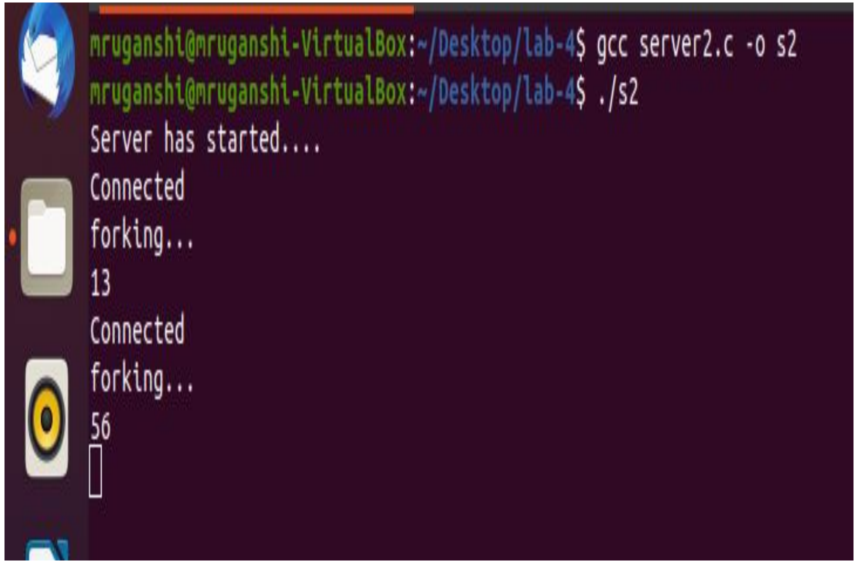


mruganshi@mruganshi-VirtualBox: ~/Desktop/l...
mruganshi@mruganshi-VirtualBox: ~/Desktop/l...

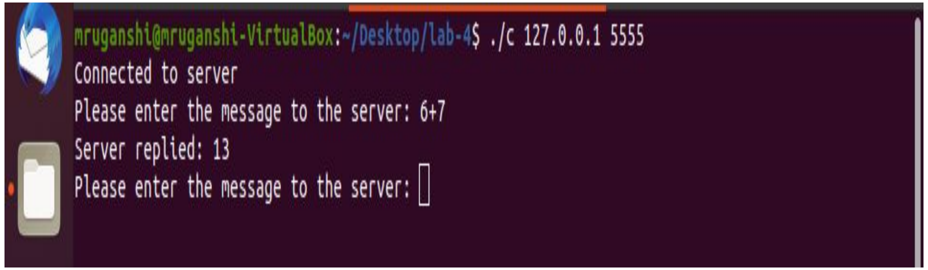
```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./c 127.0.0.1 5555
line busy...
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$
```

Server-2 Outputs

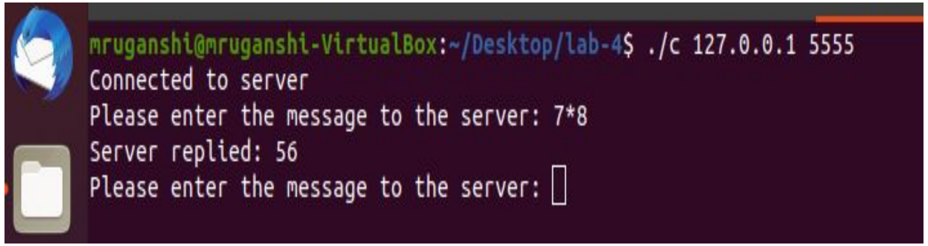
- Multi client service using fork
- We are distributing the tasks of setting up connection and request handling among parent and child processes respectively.



```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ gcc server2.c -o s2
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./s2
Server has started....
Connected
forking...
13
Connected
forking...
56
```



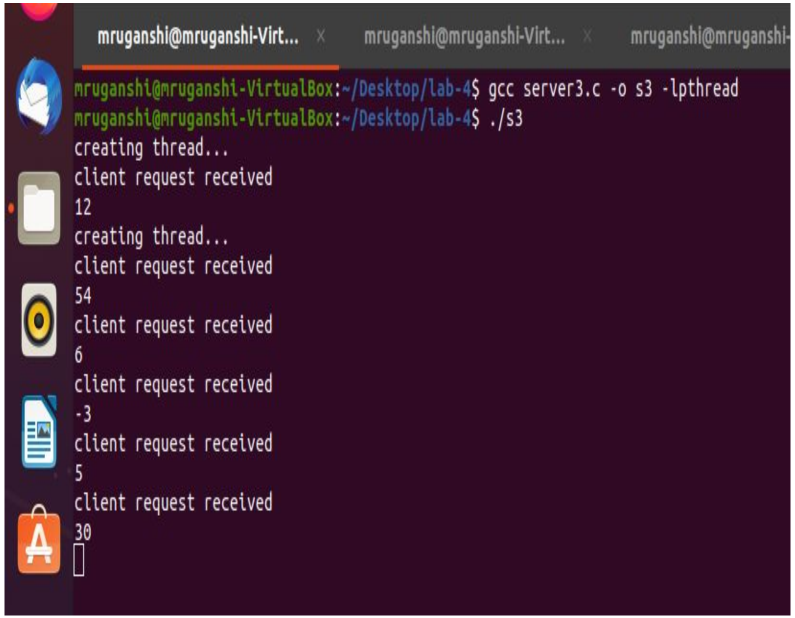
```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./c 127.0.0.1 5555
Connected to server
Please enter the message to the server: 6+7
Server replied: 13
Please enter the message to the server:
```



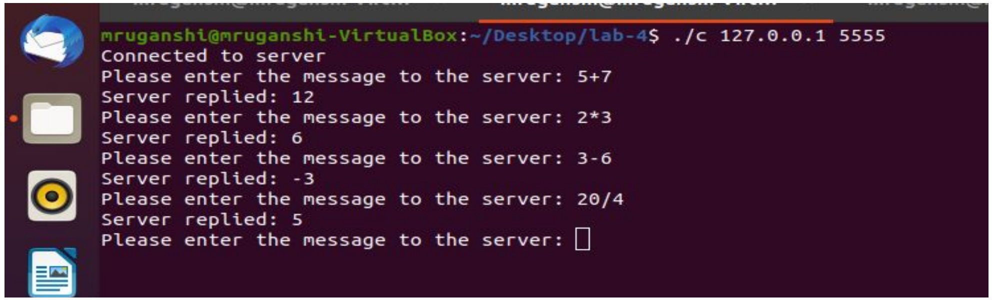
```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./c 127.0.0.1 5555
Connected to server
Please enter the message to the server: 7*8
Server replied: 56
Please enter the message to the server:
```

Server-3 Outputs

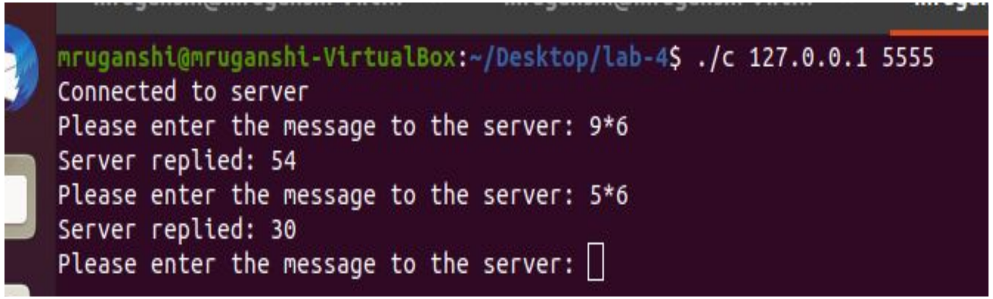
- Multi client service using threads
- We are creating a thread for handling of clients in different threads.



```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ gcc server3.c -o s3 -lpthread
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./s3
creating thread...
client request received
12
creating thread...
client request received
54
client request received
6
client request received
-3
client request received
5
client request received
30
```



```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./c 127.0.0.1 5555
Connected to server
Please enter the message to the server: 5+7
Server replied: 12
Please enter the message to the server: 2*3
Server replied: 6
Please enter the message to the server: 3-6
Server replied: -3
Please enter the message to the server: 20/4
Server replied: 5
Please enter the message to the server:
```



```
mruganshi@mruganshi-VirtualBox:~/Desktop/lab-4$ ./c 127.0.0.1 5555
Connected to server
Please enter the message to the server: 9*6
Server replied: 54
Please enter the message to the server: 5*6
Server replied: 30
Please enter the message to the server:
```

Performance Comparison

- **Clients attended :**
  Server1 - one client at a time
  Server2 - multiple clients at a time
  Server3 - multiple clients at a time
- **Comparison :** server1 < server2 = server3
- **Time taken for responding :** for huge number of requests server1 would be faster as it interacts with with client 1 only. But here since testing has been done on few clients time taken by all would be almost same.
- **Process / thread creation -** internally the creation of threads take lesser time than creation of processes. So for huge number of clients server3 should be preferred over server2 incase we need multiple server.
- **Memory :**
  Server1 - no extra memory
  Server2 - extra memory used
  Server3 - memory is shared

All possible use cases based on benefits

- **Server1** can be used when there are **less clients** and **waiting is allowed.**
- **Server2** takes more time (in process creation) and memory because it uses **multiple processes** using fork. However it **helps in isolation of process** , as in the code ,the parent handles **socket connection** while child handles **interaction with client**. So in case we need better process isolation and program which is **easier to debug** we can use server 2 .
- **Server3** uses **multi threads** which **share memory and are faster** too. Hence they can be used according to requirement