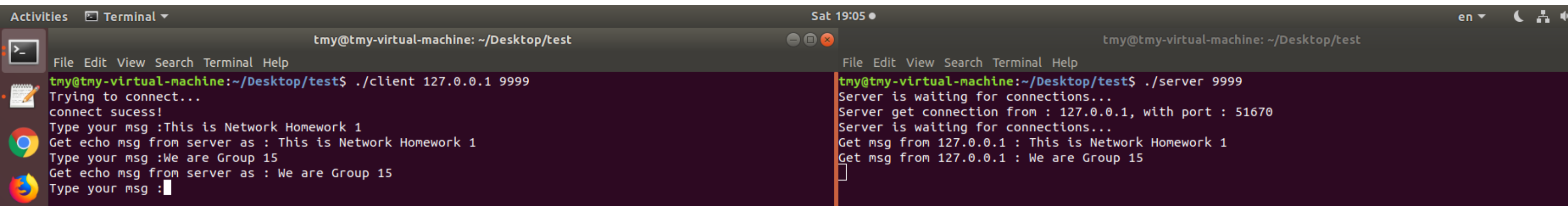
1. Test Cases:

(1) Lind of text terminated by a newline.

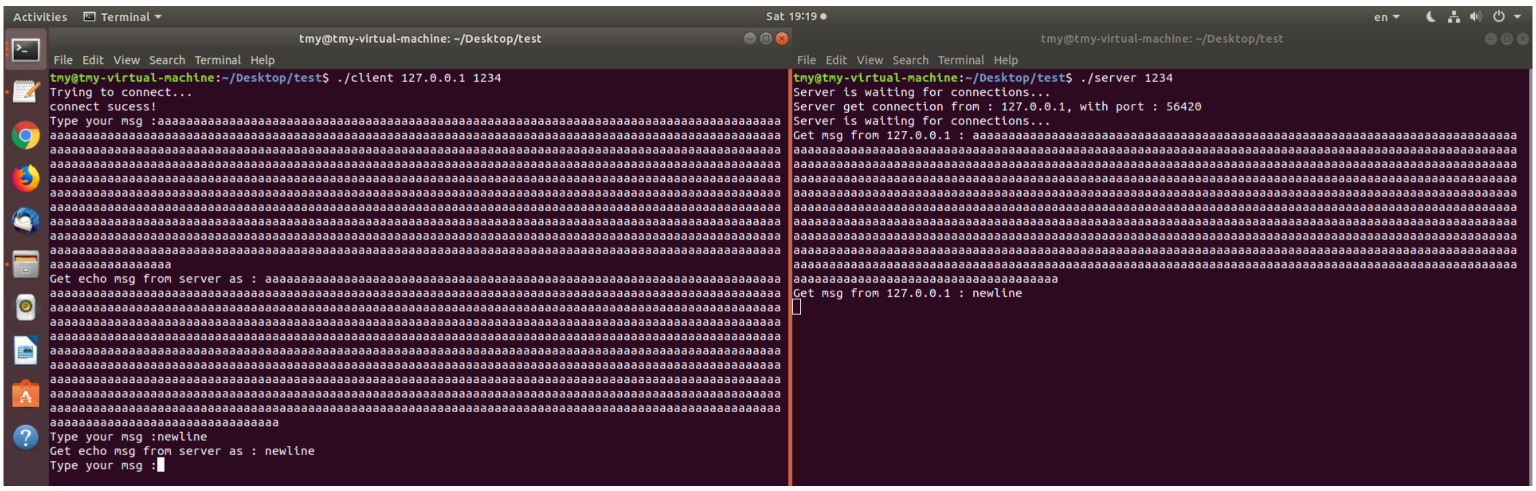


```
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 9999
Trying to connect...
connect sucess!
Type your msg :This is Network Homework 1
Get echo msg from server as : This is Network Homework 1
Type your msg :We are Group 15
Get echo msg from server as : We are Group 15
Type your msg :
```

```
tmy@tmy-virtual-machine:~/Desktop/test$ ./server 9999
Server is waiting for connections...
Server get connection from : 127.0.0.1, with port : 51670
Server is waiting for connections...
Get msg from 127.0.0.1 : This is Network Homework 1
Get msg from 127.0.0.1 : We are Group 15
```

(2) Line of text the maximum line length without a newline.

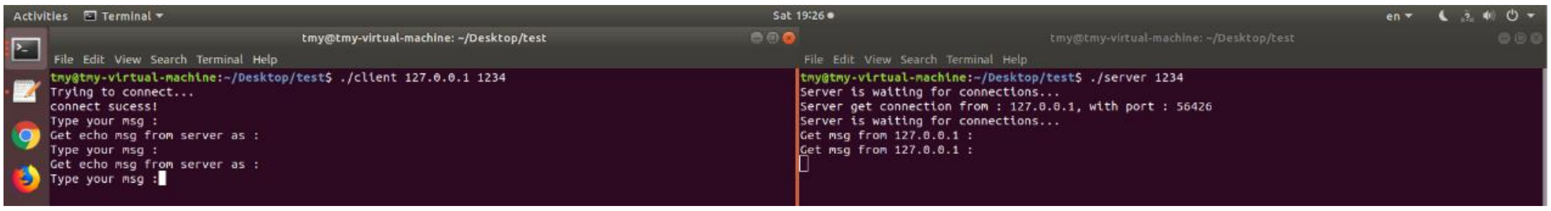
- We set our buffer as 1024 Bytes. After minus one byes of the null termination, “/0”, then the maximum line length without a newline is 1024 – 1 = 1023 Bytes. The result is show as figure below.



```
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 1234
Trying to connect...
connect sucess!
Type your msg :aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
Get echo msg from server as : aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Type your msg :newline
Get echo msg from server as : newline
Type your msg :
```

```
tmy@tmy-virtual-machine:~/Desktop/test$ ./server 1234
Server is waiting for connections...
Server get connection from : 127.0.0.1, with port : 56420
Server is waiting for connections...
Get msg from 127.0.0.1 : aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Get msg from 127.0.0.1 : newline
```

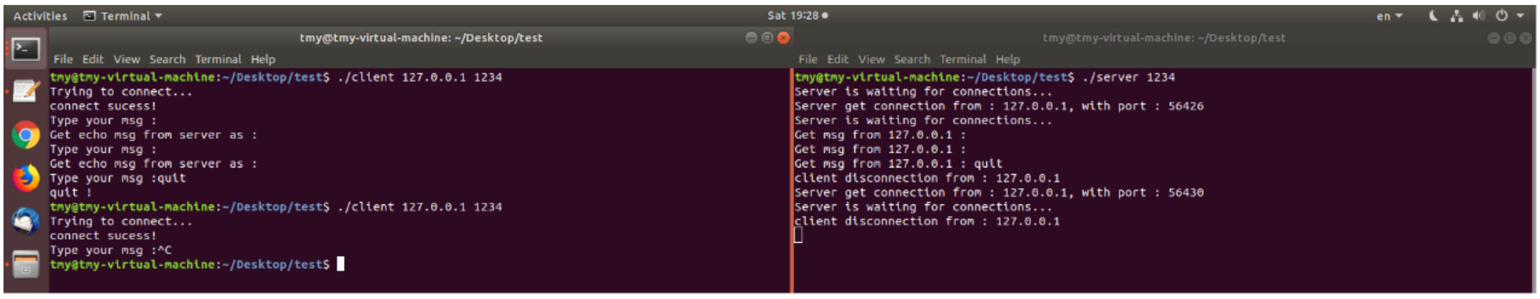
(3) Line with no characters and EOF.



```
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 1234
Trying to connect...
connect sucess!
Type your msg :
Get echo msg from server as :
Type your msg :
Get echo msg from server as :
Type your msg :
```

```
tmy@tmy-virtual-machine:~/Desktop/test$ ./server 1234
Server is waiting for connections...
Server get connection from : 127.0.0.1, with port : 56426
Server is waiting for connections...
Get msg from 127.0.0.1 :
Get msg from 127.0.0.1 :
```

(4) Client terminated after entering text.

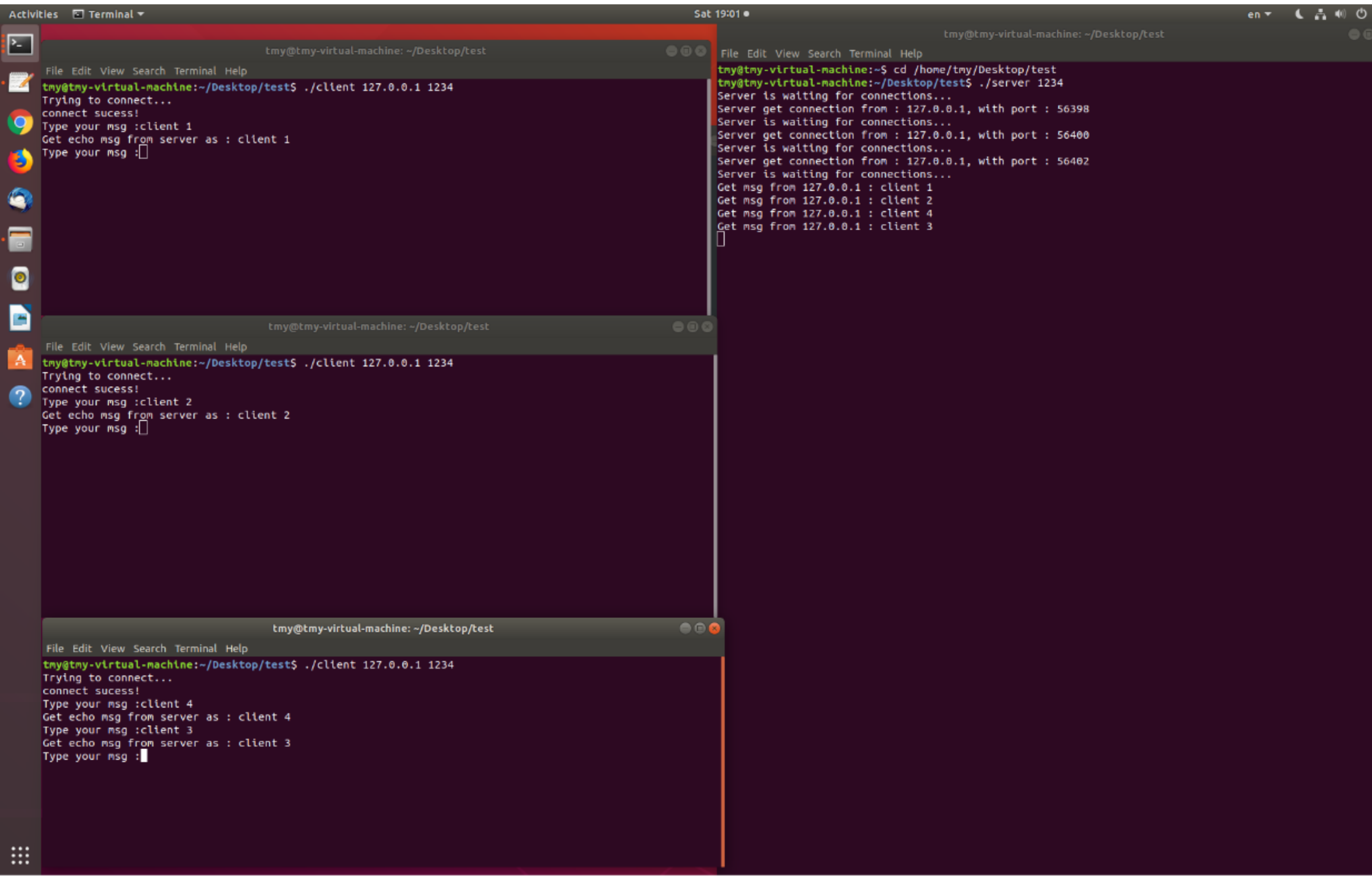


```
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 1234
Trying to connect...
connect sucess!
Type your msg :
Get echo msg from server as :
Type your msg :
Get echo msg from server as :
Type your msg :quit
quit !
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 1234
Trying to connect...
connect sucess!
Type your msg :^C
tmy@tmy-virtual-machine:~/Desktop/test$
```

```
tmy@tmy-virtual-machine:~/Desktop/test$ ./server 1234
Server is waiting for connections...
Server get connection from : 127.0.0.1, with port : 56426
Server is waiting for connections...
Get msg from 127.0.0.1 :
Get msg from 127.0.0.1 :
Get msg from 127.0.0.1 : quit
client disconnection from : 127.0.0.1
Server get connection from : 127.0.0.1, with port : 56430
Server is waiting for connections...
client disconnection from : 127.0.0.1
```

(5) Three clients connected to the server.

- For each client, we sent different messages to the server at the same connection, and the server could get the message from every client.


Activities
Terminal
Sat 19:01
en
tmy@tmy-virtual-machine: ~/Desktop/test
File Edit View Search Terminal Help

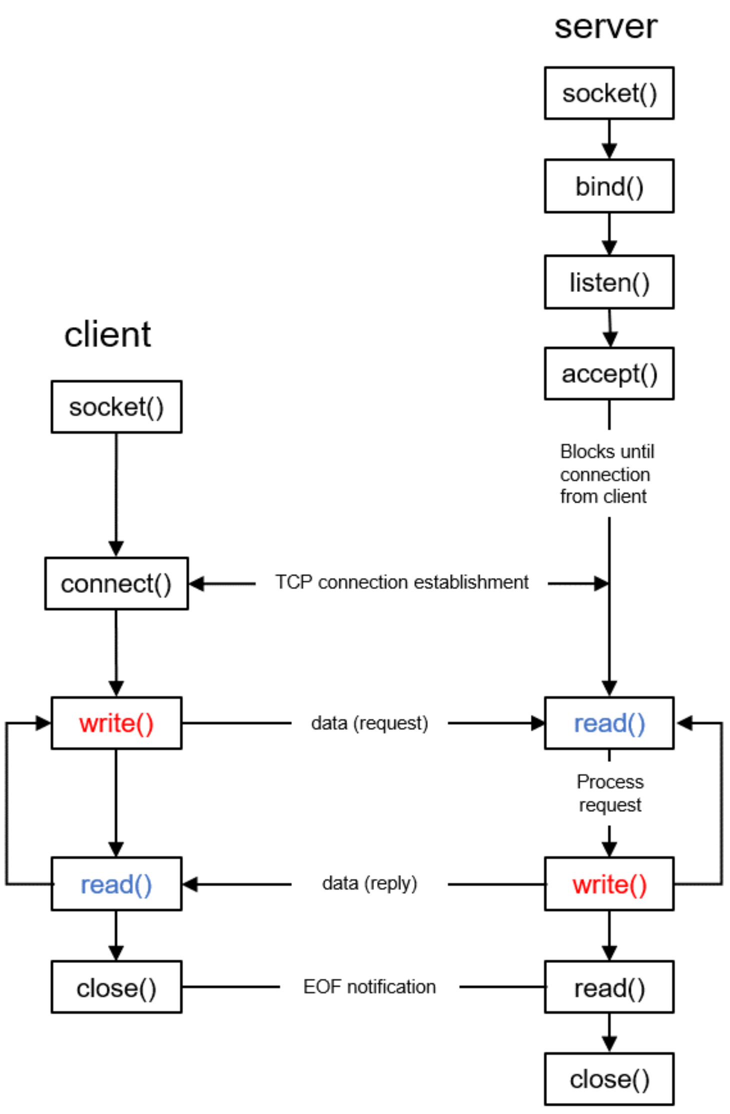
```
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 1234
Trying to connect...
connect sucess!
Type your msg :client 1
Get echo msg from server as : client 1
Type your msg :
```

```
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 1234
Trying to connect...
connect sucess!
Type your msg :client 2
Get echo msg from server as : client 2
Type your msg :
```

```
tmy@tmy-virtual-machine:~/Desktop/test$ ./client 127.0.0.1 1234
Trying to connect...
connect sucess!
Type your msg :client 4
Get echo msg from server as : client 4
Type your msg :client 3
Get echo msg from server as : client 3
Type your msg :
```

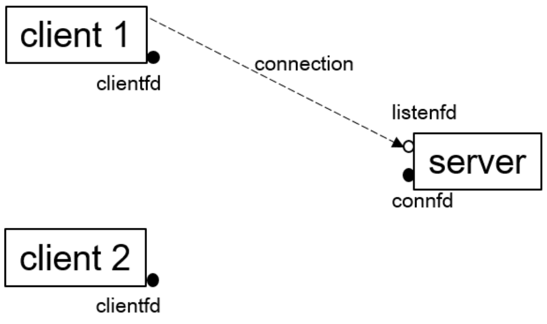
```
tmy@tmy-virtual-machine:~$ cd /home/tmy/Desktop/test
tmy@tmy-virtual-machine:~/Desktop/test$ ./server 1234
Server is waiting for connections...
Server get connection from : 127.0.0.1, with port : 56398
Server is waiting for connections...
Server get connection from : 127.0.0.1, with port : 56400
Server is waiting for connections...
Server get connection from : 127.0.0.1, with port : 56402
Server is waiting for connections...
Get msg from 127.0.0.1 : client 1
Get msg from 127.0.0.1 : client 2
Get msg from 127.0.0.1 : client 4
Get msg from 127.0.0.1 : client 3
```

2. The architecture of our code:



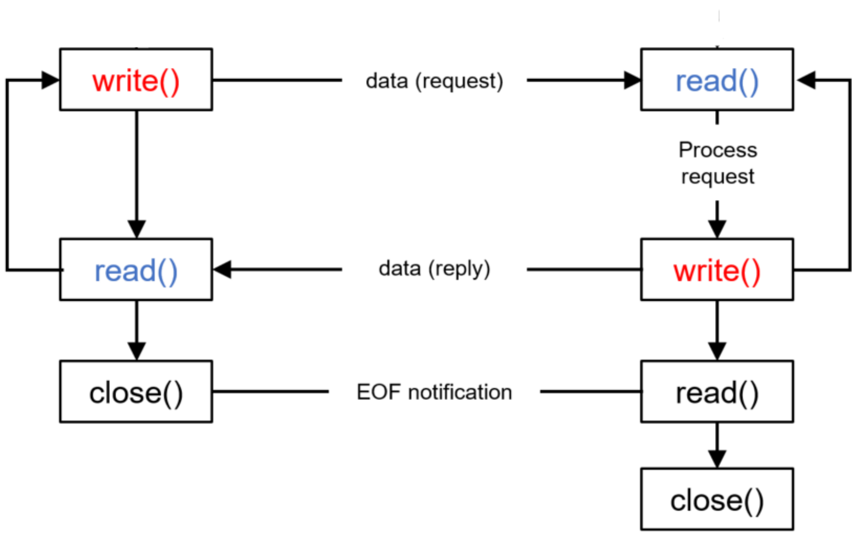
- The concept of supporting multiple simultaneous connections from clients by using fork():

  a) First step is building connection from the first client.

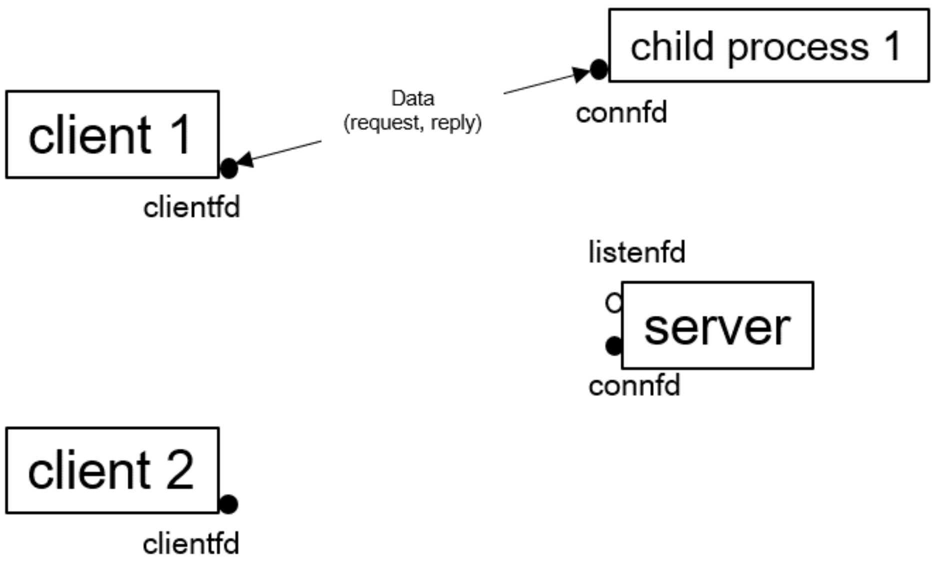


b) Second step is using fork to duplicate the process of reading and writing from the server to become a child process.

Client 1 & child process:



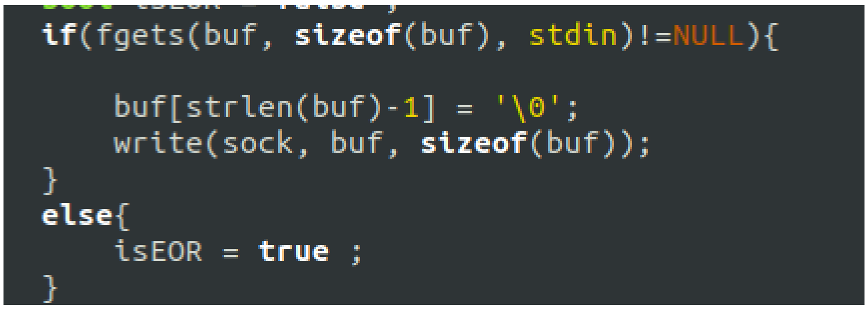
Concept of fork():



Afterward, the client 2 would follow the same steps as a) and b) to finish the connection and data transmission with the server.

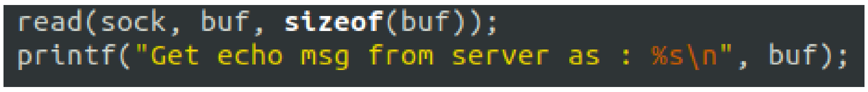
3. Usage:

a) We have handled the **writen()** and **readline()** at the client side.



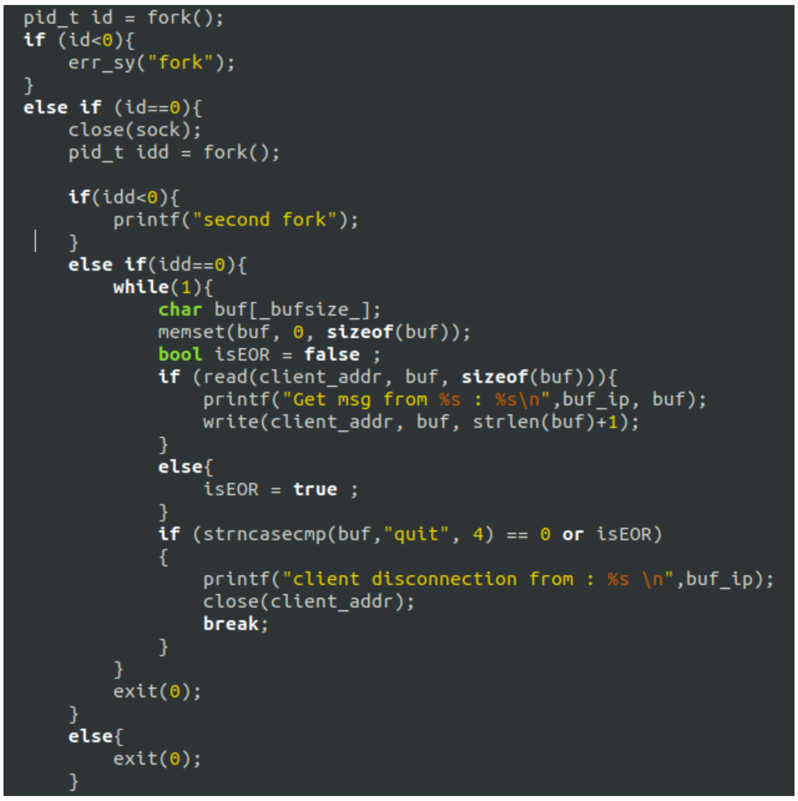
```
if(fgets(buf, sizeof(buf), stdin)!=NULL){

     buf[strlen(buf)-1] = '\0';
     write(sock, buf, sizeof(buf));
}
else{
     isEOR = true ;
}
```



```
read(sock, buf, sizeof(buf));
printf("Get echo msg from server as : %s\n", buf);
```

b) **fork()** was used at the server side.



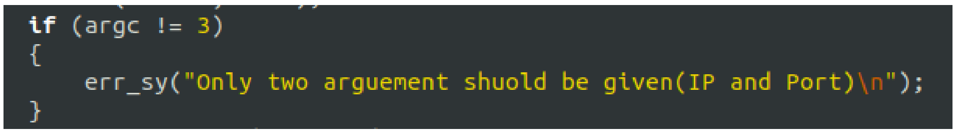
```
pid_t id = fork();
if (id<0){
    err_sy("fork");
}
else if (id==0){
    close(sock);
    pid_t idd = fork();

    if(idd<0){
        printf("second fork");
    }
    else if(idd==0){
        while(1){
            char buf[_bufsize_];
            memset(buf, 0, sizeof(buf));
            bool isEOR = false ;
            if (read(client_addr, buf, sizeof(buf))){
                printf("Get msg from %s : %s\n",buf_ip, buf);
                write(client_addr, buf, strlen(buf)+1);
            }
            else{
                isEOR = true ;
            }
            if (strncasecmp(buf,"quit", 4) == 0 or isEOR)
            {
                printf("client disconnection from : %s \n",buf_ip);
                close(client_addr);
                break;
            }
        }
        exit(0);
    }
    else{
        exit(0);
    }
```

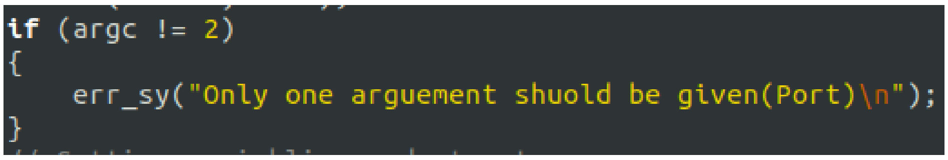
c) To disconnect the client and the server, just type a "quit" or use Ctrl + D to stop the connection.

4. Error Handling:

a) We set up some arguments for input from both server and client sides. While testing, if we did not enter enough arguments, then the system would show the error log to ask us to put the argument.

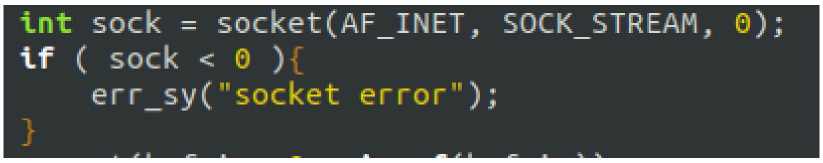


```
if (argc != 3)
{
    err_sy("Only two arguement shuold be given(IP and Port)\n");
}
```



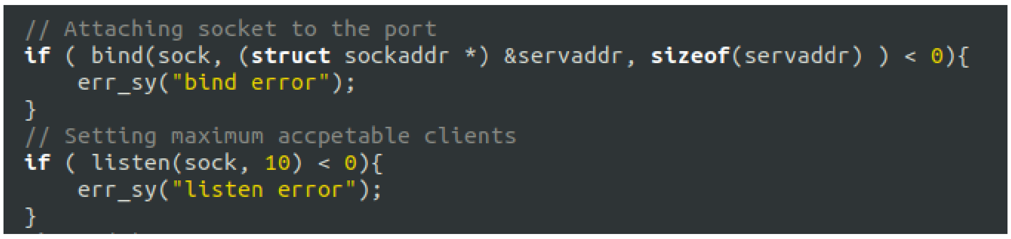
```
if (argc != 2)
{
    err_sy("Only one arguement shuold be given(Port)\n");
}
```

b) At the both server and client sides, we have constructed a socket error check. If the socket was failed to be created, then the error log would show "socket error."

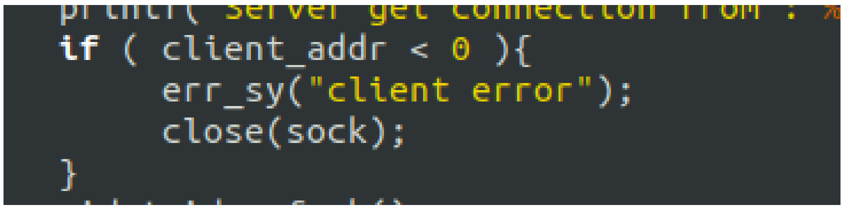


```
int sock = socket(AF_INET, SOCK_STREAM, 0);
if ( sock < 0 ){
    err_sy("socket error");
}
```

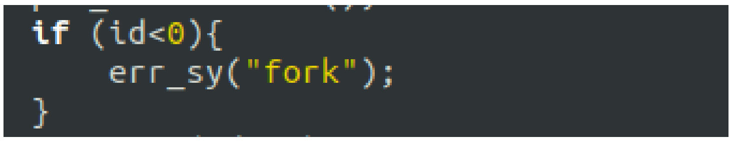
c) At the server side, we have set bind error, listen error, client connection error, fork error.



```
// Attaching socket to the port
if ( bind(sock, (struct sockaddr *) &servaddr, sizeof(servaddr) ) < 0){
    err_sy("bind error");
}
// Setting maximum accpetable clients
if ( listen(sock, 10) < 0){
    err_sy("listen error");
}
```

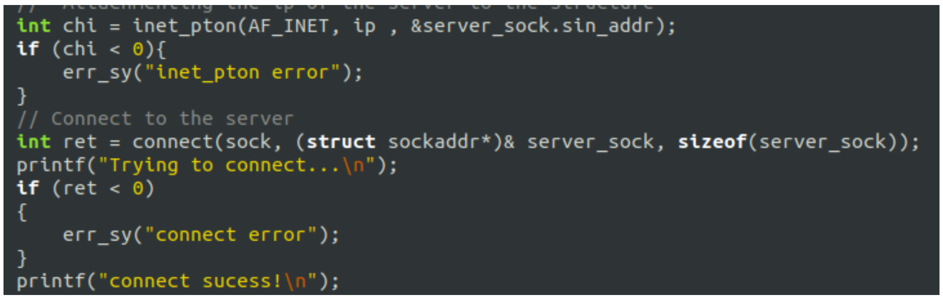


```
if ( client_addr < 0 ){
    err_sy("client error");
    close(sock);
}
```



```
if (id<0){
    err_sy("fork");
}
```

d) At the client side, we have set inet_pton error, connection error.



```
//  Attachmenting the ip of the server to the structure
int chi = inet_pton(AF_INET, ip , &server_sock.sin_addr);
if (chi < 0){
    err_sy("inet_pton error");
}
// Connect to the server
int ret = connect(sock, (struct sockaddr*)& server_sock, sizeof(server_sock));
printf("Trying to connect...\n");
if (ret < 0)
{
    err_sy("connect error");
}
printf("connect sucess!\n");
```