

ENE 4019 – Computer Network:
Assignment 2
Socket Programming

2019014266 Lim Kyu Min

Code Analysis.

Presumption
<ul style="list-style-type: none">➤ IP 주소는 LOOPBACK 주소입니다. (127.0.0.1)➤ Port 번호는 5000입니다.

Makefile
<ul style="list-style-type: none">➤ Server.c와 Client.c를 한번에 컴파일하는 makefile입니다.➤ make 키워드를 입력하면 컴파일이 진행됩니다.

Server.c
<ul style="list-style-type: none">➤ Server.c를 실행하면 가장 먼저 연결을 위해 필요한 TCP socket을 활성화합니다.➤ 소켓을 오픈한 후, 서버를 Binding을 하여 해당 소켓에 지정된 포트 번호를 부여합니다.➤ Binding에 성공하면 listen을 하여 client로부터의 request를 기다립니다. Client의 connection request가 확인되면 accept 함수를 실행하여 client와의 connection을 성립시킵니다.➤ Connection이 성립되면 Fork를 활용하여 Child Process를 생성합니다. Server는 다시 다른 client로부터의 request를 받기 위해 listen 상태로 돌아갑니다.➤ Child Process에서는 Client로부터 전송된 메시지를 buffer에 받고 (recv()) 화면에 출력합니다. 이때 출력 형식은 “[SERVER – MESSAGE RECEIVED]: Received from client: <i>MESSAGE FROM CLIENT</i>” 입니다.<ul style="list-style-type: none">✧ 아래 실행화면에서는 [SERVER – MESSAGE RECRIVED]라고 오타가 나 있습니다. 이 점 참고 바랍니다.➤ 만약 메시지를 모두 받고 출력을 했다면, 해당 client로 메시지를 받았다는 내용의 메시지를 전송합니다(send()). 이때 메시지는 “Return Message from Server”입니다.➤ Client로 메시지를 보낸 후, child process는 종료를 합니다. Request 처리가 끝났다는 의미에서 종료 메시지를 서버 화면에 띄우고, child process는 close합니다. 이때 메시지 형식은 “[SERVER]: End of the

request, closing the client server”입니다.

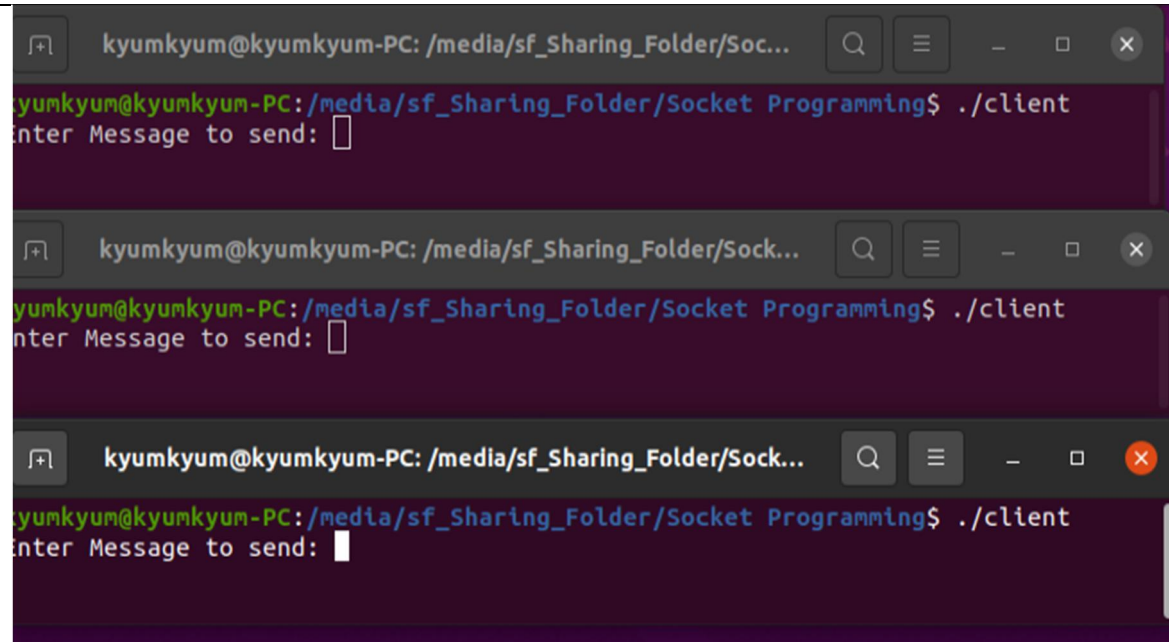
Client.c

- Client.c를 실행하면 가장 먼저 연결을 위해 필요한 TCP socket을 활성화합니다.
- Socket이 형성된 후에는, 연결 대상의 IP와 PORT정보를 넣어 connect 요청을 보냅니다. 연결에 성공하면 메시지 입력 대기 상태로 들어갑니다.
- 메시지 입력 대기상태의 client는 “Enter Message to send: ”라는 메시지를 띄우며 유저에게 입력을 받을 준비를 합니다. 메시지를 입력하고 Enter를 누르면 메시지가 서버로 전송됩니다 (send()).
- 메시지가 전송이 되면 서버가 보낸 응답 메시지를 받습니다 (recv()). 서버로부터 응답 메시지를 받으면 해당 메시지를 화면에 출력합니다. 이때 메시지의 형식은 “[CLIENT] Received From Server: MESSAGE FROM SERVER”
- 서버로부터 메시지를 받고 출력하였으면, 해당 Client는 종료됩니다.

Running Program.

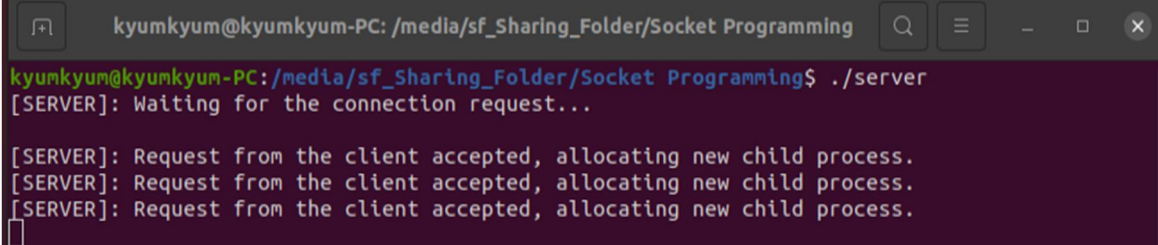
```
kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./server  
[SERVER]: Waiting for the connection request...
```

- ./server
- Server.c를 실행한 초기 화면입니다. 연결이 되어서 Client의 Request를 기다리고 있습니다.



The image shows three overlapping terminal windows. Each window has a title bar that reads 'kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Soc...'. The terminal content in each window is:
kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming\$./client
Enter Message to send: [cursor]

- ./client
- Client.c를 실행한 초기 화면입니다. 연결이 되어서 유저의 입력을 기다리고 있습니다.
- 총 3개의 터미널을 open한 후, 동시에 실행하여 3개의 client가 하나의 server에 동시 접속해 있는 상태입니다.



The image shows a single terminal window with the same title bar. The output is:
kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming\$./server
[SERVER]: Waiting for the connection request...

[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[cursor]

- 두번째 사진처럼 Client가 동시 접속했을 때의 Server의 화면입니다. 3개의 Client가 모두 연결된 것을 확인할 수 있습니다.

```

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./server
[SERVER]: Waiting for the connection request...

[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[]

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: []

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: []

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: []

```

- 현재까지의 상황입니다.

```

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: Client 1 says "Computer Network is fun!"[]

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: Client 2 says "Socket Programming is interesting!"[]

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: Client 3 shakes the carrot![]

```

- 다음과 같이 입력을 하여 서버에 전송을 합니다.
 - Client 1 says “Computer Network is fun!”
 - Client 2 says “Socket Programming is interesting!”
 - Client 3 shakes the carrot!

```

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming$ ./server
[SERVER]: Waiting for the connection request...

[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[SERVER - MESSAGE RECEIVED]: Received from client: Client 1 says "Computer Network is fun!"
[SERVER]: End of the request, closing the client server
[SERVER - MESSAGE RECEIVED]: Received from client: Client 2 says "Socket Programming is interesting!"
[SERVER]: End of the request, closing the client server
[SERVER - MESSAGE RECEIVED]: Received from client: Client 3 shakes the carrot!
[SERVER]: End of the request, closing the client server
[]

```

- Server에서는 Client에서 전송된 메시지를 출력합니다.
- 출력한 후, Server는 다시 Client로 Return Message를 보냅니다.
 - Return Message from Server
- Return Message를 보낸 후, 해당 Client를 담당하는 Child Process는 종료를 합니다.

```

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Soc...
kyumkyum@kyumkyum-PC:/media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: Client 1 says "Computer Network is fun!"
[CLIENT] Received From Server: Return Message from Server
kyumkyum@kyumkyum-PC:/media/sf_Sharing_Folder/Socket Programming$

kyumkyum@kyumkyum-PC:/media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: Client 2 says "Socket Programming is interesting!"
[CLIENT] Received From Server: Return Message from Server
kyumkyum@kyumkyum-PC:/media/sf_Sharing_Folder/Socket Programming$

kyumkyum@kyumkyum-PC:/media/sf_Sharing_Folder/Socket Programming$ ./client
Enter Message to send: Client 3 shakes the carrot!
[CLIENT] Received From Server: Return Message from Server
kyumkyum@kyumkyum-PC:/media/sf_Sharing_Folder/Socket Programming$
  
```

- Server가 해당 Client에 Return Message를 보내면 해당 Client는 그 Message를 출력합니다.
- 해당 메시지를 출력한 후, Client 프로그램은 종료됩니다.

```

kyumkyum@kyumkyum-PC: /media/sf_Sharing_Folder/Socket Programming
kyumkyum@kyumkyum-PC:/media/sf_Sharing_Folder/Socket Programming$ ./server
[SERVER]: Waiting for the connection request...

[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[SERVER]: Request from the client accepted, allocating new child process.
[SERVER - MESSAGE RECEIVED]: Received from client: Client 1 says "Computer Network is fun!"
[SERVER]: End of the request, closing the client server
[SERVER - MESSAGE RECEIVED]: Received from client: Client 2 says "Socket Programming is interesting!"
[SERVER]: End of the request, closing the client server
[SERVER - MESSAGE RECEIVED]: Received from client: Client 3 shakes the carrot!
[SERVER]: End of the request, closing the client server
  
```

- 모든 Client의 Request가 끝난 후의 전체화면입니다.

-END-