

Computer Network Project 2

2015147533 유현석

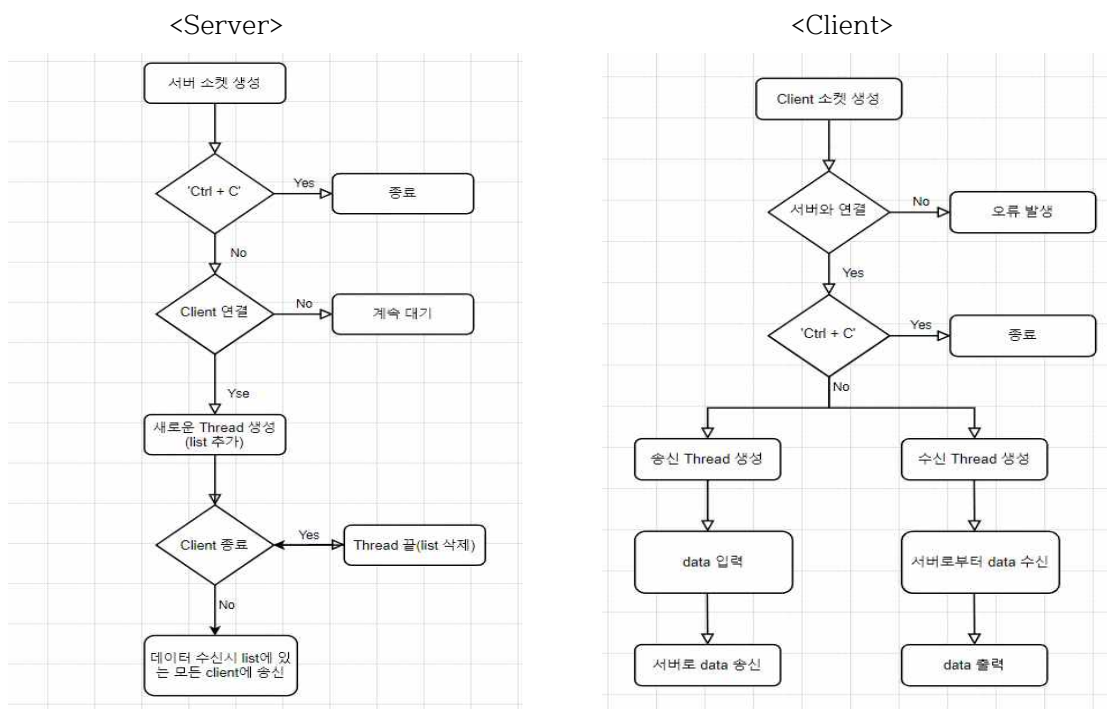
1. Introduction

Software environment : Ubuntu (Linux)

Programming language(version) : Python 3.7.9

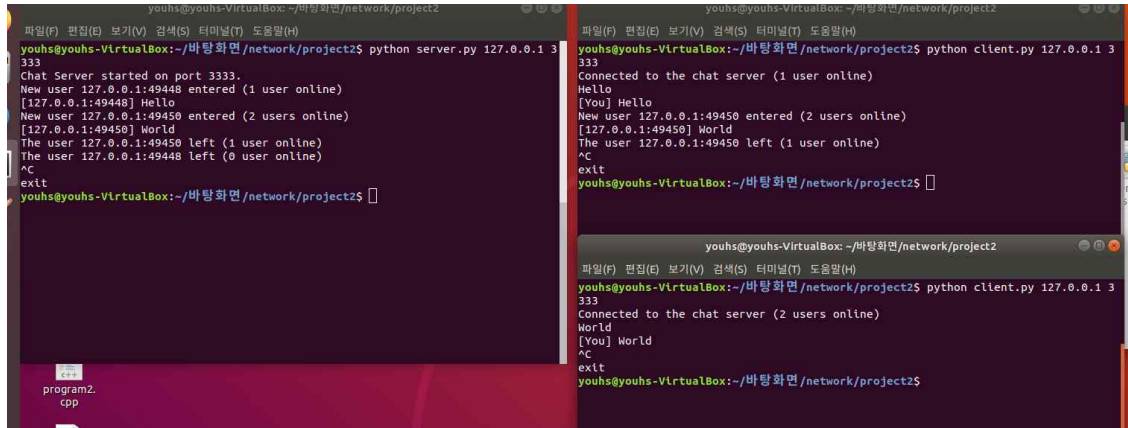
```
youhs@youhs-VirtualBox: ~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
youhs@youhs-VirtualBox:~$ lsb_release -a  
No LSB modules are available.  
Distributor ID: Ubuntu  
Description:    Ubuntu 18.04.5 LTS  
Release:        18.04  
Codename:       bionic  
youhs@youhs-VirtualBox:~$ python -V  
Python 3.7.9  
youhs@youhs-VirtualBox:~$ uname -a  
Linux youhs-VirtualBox 5.4.0-52-generic #57~18.04.1-Ubuntu SMP Thu Oct 15 14:04:  
49 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux  
youhs@youhs-VirtualBox:~$ cat /etc/issue  
Ubuntu 18.04.5 LTS \n \l  
youhs@youhs-VirtualBox:~$
```

2. Flow chart or Diagram



3. Snapshots

<2개의 Client와 통신>

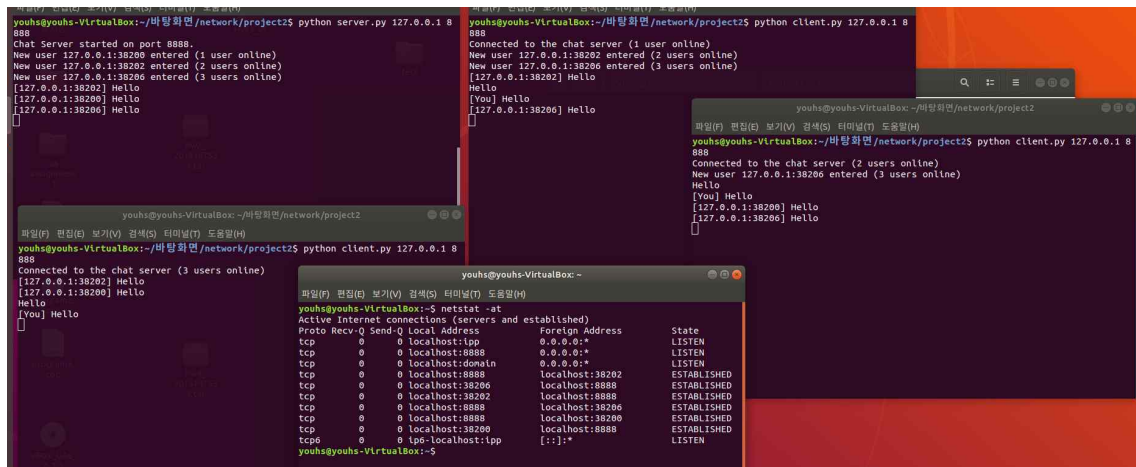


```
youhs@youhs-VirtualBox: ~/바탕화면/network/project2
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
youhs@youhs-VirtualBox:~/바탕화면/network/project2$ python server.py 127.0.0.1 3333
Chat Server started on port 3333.
New user 127.0.0.1:49448 entered (1 user online)
[127.0.0.1:49448] Hello
New user 127.0.0.1:49450 entered (2 users online)
[127.0.0.1:49450] World
The user 127.0.0.1:49450 left (1 user online)
The user 127.0.0.1:49448 left (0 user online)
^C
exit
youhs@youhs-VirtualBox:~/바탕화면/network/project2$

youhs@youhs-VirtualBox:~/바탕화면/network/project2$ python client.py 127.0.0.1 333
Connected to the chat server (1 user online)
[You] Hello
New user 127.0.0.1:49450 entered (2 users online)
[127.0.0.1:49450] World
The user 127.0.0.1:49450 left (1 user online)
^C
exit
youhs@youhs-VirtualBox:~/바탕화면/network/project2$

youhs@youhs-VirtualBox:~/바탕화면/network/project2$ python client.py 127.0.0.1 333
Connected to the chat server (2 users online)
[You] World
^C
exit
youhs@youhs-VirtualBox:~/바탕화면/network/project2$
```

<3개의 Client와 통신 + Socket 정보(연결 중)>

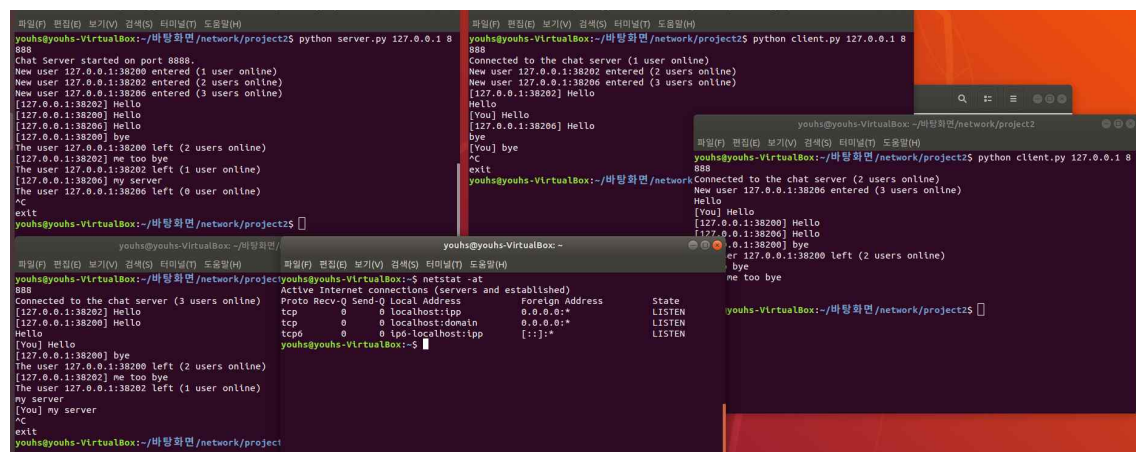


```
youhs@youhs-VirtualBox:~/바탕화면/network/project2$ python server.py 127.0.0.1 8888
Chat Server started on port 8888.
New user 127.0.0.1:38208 entered (1 user online)
New user 127.0.0.1:38202 entered (2 users online)
New user 127.0.0.1:38206 entered (3 users online)
[127.0.0.1:38202] Hello
[127.0.0.1:38208] Hello
[127.0.0.1:38206] Hello
^C
exit
youhs@youhs-VirtualBox:~/바탕화면/network/project2$

youhs@youhs-VirtualBox:~/바탕화면/network/project2$ python client.py 127.0.0.1 888
Connected to the chat server (3 users online)
[127.0.0.1:38202] Hello
[You] Hello
^C
exit
youhs@youhs-VirtualBox:~/바탕화면/network/project2$

youhs@youhs-VirtualBox:~/바탕화면/network/project2$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 localhost:ipp 0.0.0.0:* LISTEN
tcp 0 0 localhost:8888 0.0.0.0:* LISTEN
tcp 0 0 localhost:domain 0.0.0.0:* LISTEN
tcp 0 0 localhost:8888 localhost:38202 ESTABLISHED
tcp 0 0 localhost:38202 localhost:8888 ESTABLISHED
tcp 0 0 localhost:8888 localhost:38206 ESTABLISHED
tcp 0 0 localhost:8888 localhost:38208 ESTABLISHED
tcp 0 0 localhost:38208 localhost:8888 ESTABLISHED
tcp6 0 0 ip6-localhost:ipp ::::* LISTEN
youhs@youhs-VirtualBox:~/바탕화면/network/project2$
```

<3개의 Client와 통신 + Socket 정보(종료된 후)>



```
youhs@youhs-VirtualBox:~/바탕화면/network/project2$ python server.py 127.0.0.1 8888
Chat Server started on port 8888.
New user 127.0.0.1:38208 entered (1 user online)
New user 127.0.0.1:38202 entered (2 users online)
New user 127.0.0.1:38206 entered (3 users online)
[127.0.0.1:38202] Hello
[127.0.0.1:38208] Hello
[127.0.0.1:38206] Hello
The user 127.0.0.1:38208 left (2 users online)
[127.0.0.1:38202] ne too bye
The user 127.0.0.1:38202 left (1 user online)
[127.0.0.1:38206] my server
The user 127.0.0.1:38206 left (0 user online)
^C
exit
youhs@youhs-VirtualBox:~/바탕화면/network/project2$

youhs@youhs-VirtualBox:~/바탕화면/network/project2$ python client.py 127.0.0.1 888
Connected to the chat server (1 user online)
New user 127.0.0.1:38202 entered (2 users online)
New user 127.0.0.1:38206 entered (3 users online)
[127.0.0.1:38202] Hello
[You] Hello
[127.0.0.1:38206] Hello
[You] bye
^C
exit
youhs@youhs-VirtualBox:~/바탕화면/network/project2$

youhs@youhs-VirtualBox:~/바탕화면/network/project2$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 localhost:ipp 0.0.0.0:* LISTEN
tcp 0 0 localhost:domain 0.0.0.0:* LISTEN
tcp6 0 0 ip6-localhost:ipp ::::* LISTEN
youhs@youhs-VirtualBox:~/바탕화면/network/project2$
```

4. Logical explanations

1. srv.py

① main

```
#start of the main
HOST = sys.argv[1]
PORT = sys.argv[2]
print("Chat Server started on port " + PORT + ".")
#get the input for ip and port

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((HOST, int(PORT)))
server_socket.listen()
#setting the server option and function

while True:
    # when the client comes accept will return new socket and will communicate with socket in new thread
    try:
        client_socket, addr = server_socket.accept()
        usernum += 1
        addbool = 1
        client_list.append(client_socket)
        start_new_thread(threaded, (client_socket, addr))    # add the client list

    #when the KeyboardInterrupt occur
    except KeyboardInterrupt:
        server_socket.close()
        print("")
        print('exit')
        break
server_socket.close()
```

-> 인풋으로 들어오는 ip와 port를 저장하고 그 저장된 값을 바탕으로 서버 socket을 만듭니다. 그 과정에서 사용되는 함수들은 아래에서 설명될 함수 (socket, setsockopt, bind, listen)이 사용되었습니다. 서버를 설정한 후 client socket이 들어올 때까지 대기합니다. 만약 새로운 client가 들어오면 전체 client 숫자를 1 증가시키고 client list에 저장합니다. client list는 모든 client 정보가 저장되어있습니다. 만약 ctrl + c가 발생하면 exit를 실행하고 socket을 종료하고 끝냅니다.

② Threaded (각 thread에서 사용될 함수)

2-1 연결 됐을 때

```
def threaded(client_socket, addr):
    global usernum
    if usernum > 1:
        usertype = 'users'
    else:
        usertype = 'user'

    client_socket.send(str(usernum).encode())

    newInfo = "New user "+addr[0]+" "+str(addr[1])+" entered"+" (" +str(usernum)+" "+usertype+" online)"
    if addbool == 1:
        for i in range(len(client_list)-1):
            client_list[i].send(newInfo.encode())
    print(newInfo)
```

-> client와 연결된 후에는 그 정보를 출력하고 지금 연결되어있는 모든 client에 정보를 encode를 한 후 전송합니다.

2-2 data가 없을 때 (연결이 끊어짐)

```
# untill the client end |
while True:
    try:
        # when the data get in send to client again
        data = client_socket.recv(1024)

        if not data:
            usernum -= 1
            if usernum > 1:
                usertype = 'users'
            else:
                usertype = 'user'
            exitsrt = "The user "+addr[0]+":"+str(addr[1])+ " left"+ " (" +str(usernum)+ " "+usertype+" online)"
            print(exitsrt)
            client_list.remove(client_socket)
            for i in range(len(client_list)):
                client_list[i].send(exitsrt.encode())

            break
        #when the client leaving
```

-> 연결이 끊어져서 더이상 data가 수신되지 않으면 끊어진 client를 list에서 삭제하고 그 정보를 출력하고 연결된 모든 client에 전송합니다.

2-3 data가 수신되었을 때

```
print([' ' + addr[0],end=''])
print(':',end='')
print(addr[1],end='')
print(']', data.decode())

for i in range(len(client_list)):
    if client_list[i] == client_socket:
        you = "[You] "
        client_list[i].send(you.encode())
    else:
        ipad = "["+addr[0]+":"+str(addr[1])+ "]" + " "
        client_list[i].send(ipad.encode())

for i in range(len(client_list)):
    client_list[i].send(data)

#for receive the data and sending to all client
```

-> 새로운 data가 수신된다면 그 정보를 출력하고 연결된 모든 client에 전송합니다. 그 과정에서 data를 보낸 client한테는 [You]로 보내고 나머지 client에게는 data를 보낸 client의 정보를 담은 [ip : port] 형태로 전송합니다.

2-4 에러가 발생했을 때

```
except ConnectionResetError as e:
    usernum -= 1
    if usernum > 1:
        usertype = 'users'
    else:
        usertype = 'user'
    exitsrt = "The user "+addr[0]+":"+str(addr[1])+ " left"+ " (" +str(usernum)+ " "+usertype+" online)"
    print(exitsrt)
    client_list.remove(client_socket)
    break
#when the client occur the error
_socket.close()
```

-> client에서 에러가 발생해서 연결이 끊어지면 끊어진 client를 list에서 삭제하고 그 정보를 출력하고 연결된 모든 client에 전송합니다.

2. cli.py

① main

```
# start of main
user_type = 'user'
HOST = sys.argv[1]
PORT = sys.argv[2]
# get the info of input

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, int(PORT)))
data = client_socket.recv(1024)
# connect with the server

server_num = int(data.decode())
if server_num > 1:
    user_type = 'users'
else:
    user_type = 'user'
print('Connected to the chat server (' + end='')
print(server_num, user_type, 'online')
# check the number of user and print the info

while True:
    try:
        th_a = threading.Thread(target=received, args=(client_socket,))
        th_b = threading.Thread(target=sended, args=(client_socket,))
        th_a.daemon = True
        th_b.daemon = True
        th_a.start()
        th_b.start()
        th_a.join()
        th_b.join()
    except KeyboardInterrupt:
        client_socket.close()
        print("")
        print("exit")
        break;
# if KeyboardInterrupt occur finish the whole
client_socket.close()
```

-> 인풋으로 받은 정보를 저장하고 그 정보를 바탕으로 server랑 연결을 합니다. 이때 사용되는 함수는 마찬가지로 아래에서 설명된 함수를 사용합니다. (socket, connect) 그 후 서버로부터 전송받은 정보를 바탕으로 2 이상이면 users 아니면 user로 설정을 해준 후 출력합니다. 그 후 2개의 thread를 생성하고 그 thread는 각각 received 함수와 sendd 함수를 사용합니다. daemon을 사용한 이유는 전체 메인이 끝났을 때 thread도 마찬가지로 끝내기 위해서입니다. start로 thread를 시작하고 join을 통해서 thread가 끝날 때까지 main을 대기합니다. 만약 Ctrl + c가 입력되면 socket을 종료하고 전체 프로그램을 종료합니다.

② received

```
def received(client_socket):
    while True:
        data = client_socket.recv(1024)
        print(data.decode())
# when the data received
```

-> 서버로부터 전송받은 정보를 decode를 통해서 출력합니다. 정보를 전송받자마자 독립적으로 출력합니다.

③ sendd

```
def sendd(client_socket):
    while True:
        message = input('')
        client_socket.send(message.encode())
# when send the data
```

-> 전송하고자 하는 input을 입력하면 그 정보를 encode를 통해서 서버로 전송합니다.

5. Explanations of 8 functions

1. socket()

socket을 생성하는 함수로 첫 번째 인자는 family 두 번째 인자는 type을 나타냅니다. Family는 주소 체계를 지정하는 인자로 socket.AF_INET = IPv4, socket.AF_INET6 = IPv6 이 있습니다. Type은 socket 타입으로 raw, stream, datagram등이 있습니다.

2. setsockopt()

socket의 옵션값을 변경하기 위해서 사용되는 함수입니다. 인자는 다음과 같습니다. 그 중에서 SO_REUSEADDR 옵션은 기존에 할당된 socket 정보를 재사용할 수 있게 해줍니다.

socket : 소켓의 번호

level : 옵션의 종류. 보통 SOL_SOCKET와 IPPROTO_TCP 중 하나를 사용

optname : 설정을 위한 소켓 옵션의 번호

optval : 설정 값이 저장된 주소값.

optlen : optval 버퍼의 크기

3. bind()

클라이언트에서보다는 서버에서 필요한 함수로 서버를 만들 때 사용되는 함수입니다. 튜플 형식으로 받고 튜플의 앞부분은 ip address이고 뒷부분은 port number입니다.

4. listen()

서버가 데이터 수신을 기다리는 상태로 인자로 들어가는 숫자는 해당 socket이 몇 개의 동시 접속을 허용할 것이냐는 의미입니다.

5. connect()

클라이언트에서 서버에 접속하기 위한 함수입니다. 인자로 (호스트 주소, 포트번호)로 구성된 튜플이 사용됩니다.

6. close()

해당 socket을 그만 사용할 때 사용하는 함수입니다. 즉 연결 종료 상태로 하는 함수입니다.

7. Thread()

Thread 객체를 생성하는 함수입니다. 인자로 실행될 함수와 그 함수에서 사용될 매개변수들이 전달됩니다. start로 시작할 수 있으며 join으로 끝날 때까지 대기하게 할 수 있습니다. Daemon 설정을 통해서 전체 프로그램이 종료되면 자동으로 종료되도록 설정할 수 있습니다.

6. Difference multi-thread vs select()

Multi-thread : Thread란 프로세스 내에서 실행되는 여러 흐름의 단위입니다. 즉 multi-thread란 process 내에서 1개 이상의 thread가 존재하는 것입니다. 각각의 thread는 stack만 따로 할당 받고 code, data, heap 영역을 공유 받습니다. 이러한 특징 때문에 여러 장점과 단점을 갖습니다.

장점 : 메모리 공유로 인한 시스템 자원 소모가 줄어듭니다. 응답시간이 단축됩니다. Context Switching에 대한 오버헤드가 줄어듭니다.

단점 : 서로 데이터를 사용하다가 충돌이 일어날 가능성이 있습니다. (동기화 문제) 테스트 및 디버깅이 어렵습니다. 전체 프로세스에 영향을 줍니다.

Select() : Multi-thread를 사용하면서 발생하는 단점을 극복하기 위한 방법으로 여러 thread를 생성해서 독립적으로 실행하는 방법이 아닌 하나의 thread 내에서 select 함수를 이용하여 멀티 플렉싱하는 것입니다. 특정 조건을 만족했을 때 선택하여 진행되도록 하는 함수입니다. 이를 통해 다중화 입출력을 할 수 있습니다.

장점 : 위에서 발생할 수 있는 단점들을 해결할 수 있습니다. 즉 동기화 문제, 디버깅 및 테스트에서의 장점이 있습니다.

단점 : 조건에 만족하는 것을 계속 확인하고 체크하기 때문에 실행 시간이 길어집니다. 또한 채널의 수가 1024개로 제한되어 있습니다.

7. Reference

<https://bnzn2426.tistory.com/53>

<https://battlewithmyself.tistory.com/48>

<https://hamait.tistory.com/833>

<https://itholic.github.io/python-select/>

<https://asfirstalways.tistory.com/340>

<https://magi82.github.io/process-thread/>