

소켓프로그래밍

라즈베리파이로 배우는 소켓 통신 프로그래밍



동양미래대학교
컴퓨터공학부 정석용



동양미래대학교

pwd : 현재 작업 중인 디렉터리의 이름을 출력 (**P**rint **W**orking **D**irectory)

```
① pi@raspberrypi: ~  
pi@raspberrypi:~ $ pwd  
/home/pi  
pi@raspberrypi:~ $
```

① **pwd** : 현재 작업 중인 디렉터리는 **"/home/pi"** 라는 것을 확인할 수 있음

ls : 파일 및 디렉터리 목록 보기

```

pi@raspberrypi: ~
① pi@raspberrypi:~ $ ls
Bookshelf Desktop Downloads Pictures Templates Videos
capture Documents Music Public thinclient_drives

② pi@raspberrypi:~ $ ls -a
. .cache .gnupg Public .xorgxrdp.10.log
.. .capture .local Templates .xorgxrdp.10.log.old
.bash_history .config Music thinclient_drives .xsession-errors
.bash_logout Desktop Pictures Videos .xsession-errors.old
.bashrc Documents .pki .vnc
Bookshelf Downloads .profile .Xauthority

③ pi@raspberrypi:~ $ ls -l
total 44
drwxr-xr-x 2 pi pi 4096 Aug 20 11:40 Bookshelf
drwxr-xr-x 2 pi pi 4096 Dec 14 09:36 capture
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Desktop
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Documents
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Downloads
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Music
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Pictures
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Public
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Templates
drwxr-xr-t 2 pi pi 4096 Nov 22 04:06 thinclient_drives
drwxr-xr-x 2 pi pi 4096 Aug 20 11:54 Videos
pi@raspberrypi:~ $
    
```

- ① ls : 파일이나 디렉터리 목록 출력
- ② ls -a : '.'으로 시작하는 숨김 파일(디렉터리) 목록 까지 출력
- ③ ls -l : 파일(디렉터리)에 대한 자세한 정보 까지 출력

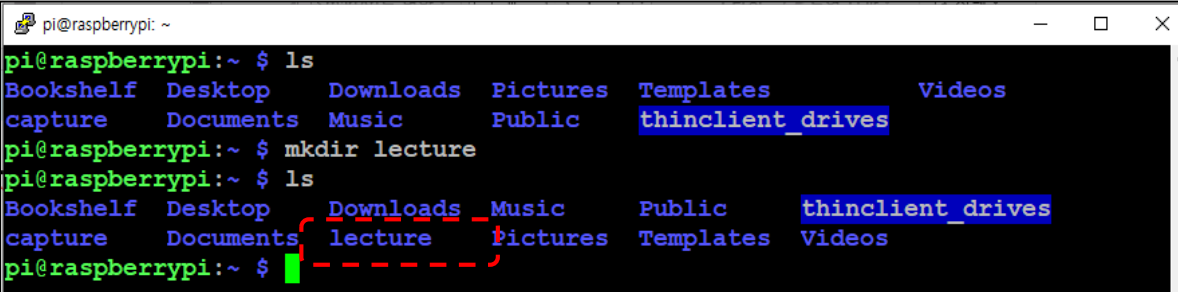
ls : 파일 및 디렉터리 목록 보기

```
pi@raspberrypi: ~
pi@raspberrypi:~ $ ls -l test.c
-rwxrw-r-x 1 pi pi 5 Dec 14 09:58 test.c
pi@raspberrypi:~ $
```

①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
-	rwx	rw-	r-x	2	pi	pi	5	Dec 14 09:58	test.c

- ① 종류 : '-' 파일, 'd' 디렉터리
- ② 소유자 접근 권한 : 접근 권한은 rwx(읽기, 쓰기, 실행)로 구분하고, r, w, x 권한을 가짐
- ③ 그룹 접근 권한 : 접근 권한은 rwx(읽기, 쓰기, 실행)로 구분하고, r, w 권한을 가짐
- ④ 다른 사람 접근 권한 : 접근 권한은 rwx(읽기, 쓰기, 실행)로 구분하고, r, x 권한을 가짐
- ⑤ 링크 수 : 이 파일이 몇 번 링크 되었는지 명시
- ⑥ 소유자 : 이 파일에 접근할 수 있는 파일 소유자
- ⑦ 그룹 : 이 파일에 접근할 수 있는 그룹
- ⑧ 파일 크기 : 바이트 단위의 파일 크기
- ⑨ 최종 변경 날짜 : 파일 최종 변경 날짜
- ⑩ 파일 이름

mkdir : 현재 작업 디렉터리에 새로운 디렉터를 생성 (**M**ake **D**irectory)

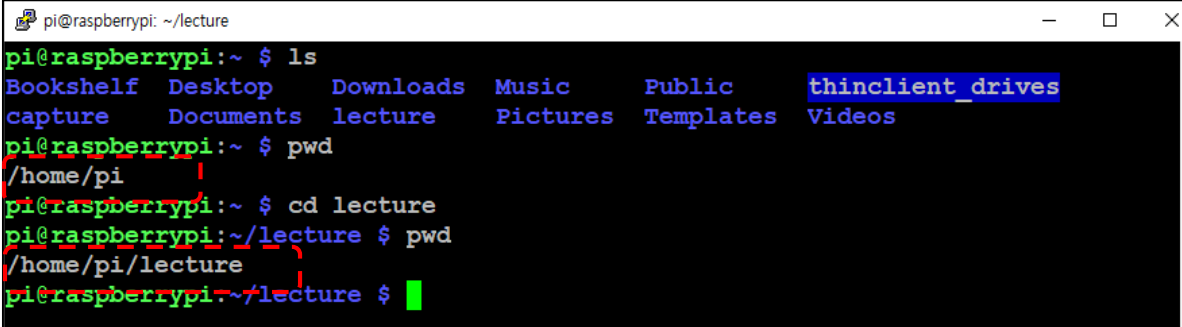


```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Pictures Templates Videos  
capture Documents Music Public thinclient_drives  
① pi@raspberrypi:~ $ mkdir lecture  
② pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Music Public thinclient_drives  
capture Documents lecture Pictures Templates Videos  
pi@raspberrypi:~ $
```

A terminal window titled 'pi@raspberrypi: ~' showing the execution of two commands. The first command, 'ls', lists the contents of the home directory, including 'thinclient_drives'. The second command, 'mkdir lecture', creates a new directory named 'lecture'. The third command, 'ls', is run again to confirm the new directory's presence in the list.

- ① **mkdir lecture** : 현재 작업 디렉터리에 새로운 디렉터리 lecture를 생성
- ② **ls** : 새로운 디렉터리 lecture 생성을 확인

cd : 디렉터리 이동 명령어 (Change Directory)



```
pi@raspberrypi: ~/lecture
pi@raspberrypi:~ $ ls
Bookshelf Desktop Downloads Music Public thinclient_drives
capture Documents lecture Pictures Templates Videos
① pi@raspberrypi:~ $ pwd
/home/pi
② pi@raspberrypi:~ $ cd lecture
③ pi@raspberrypi:~/lecture $ pwd
/home/pi/lecture
pi@raspberrypi:~/lecture $
```

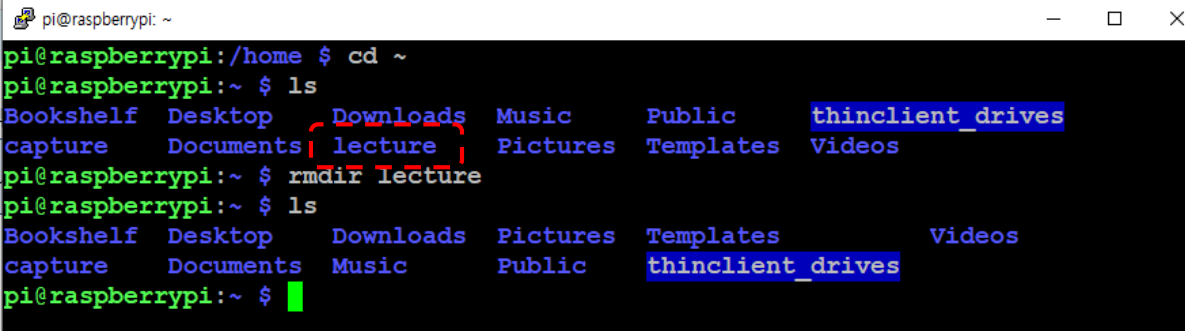
- ① **pwd** : 현재 작업 디렉터리가 **/home/pi** 임을 확인
- ② **cd lecture** : lecture 디렉터리로 이동
- ③ **pwd** : 디렉터리 이동 후, 작업 디렉터리가 **/home/pi/lecture** 임 확인

cd : 디렉터리 이동 명령어 (Change Directory)

```
pi@raspberrypi: /home
① pi@raspberrypi:~/lecture $ cd /
② pi@raspberrypi:/ $ pwd
/
③ pi@raspberrypi:/ $ cd /usr/lib
④ pi@raspberrypi:/usr/lib $ pwd
/usr/lib
⑤ pi@raspberrypi:/usr/lib $ cd ~
⑥ pi@raspberrypi:~ $ pwd
/home/pi
⑦ pi@raspberrypi:~ $ cd ..
⑧ pi@raspberrypi:/home $ pwd
/home
pi@raspberrypi:/home $
```

- ① `cd /` : 최상위 디렉터리 (root)로 이동
- ② `pwd` : 디렉터리 이동 후, 작업 디렉터리가 / (root)임을 확인
- ③ `cd /usr/lib` : 절대 경로 /usr/lib 디렉터리로 이동
- ④ `pwd` : 디렉터리 이동 후, 작업 디렉터리가 /usr/lib 임을 확인
- ⑤ `cd ~` : 사용자의 홈 디렉터리로 이동 (현재 사용자 계정은 pi)
- ⑥ `pwd` : 디렉터리 이동 후, 작업 디렉터리가 /home/pi (pi 홈 디렉터리)임을 확인
- ⑦ `cd ..` : 현재 작업 디렉터리(/home/pi)의 상위 디렉터리로 이동
- ⑧ `pwd` : 디렉터리 이동 후, 작업 디렉터리가 /home/pi에서 /home으로 이동 확인

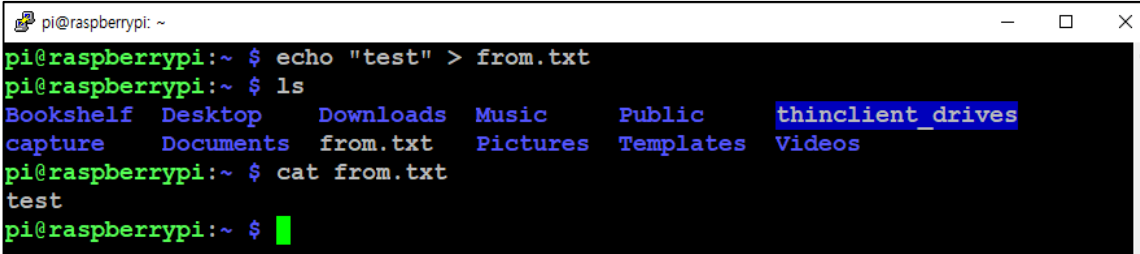
rmdir : 디렉토리를 삭제 (**Re**move **Di**rectory)



```
pi@raspberrypi: ~  
① pi@raspberrypi:/home $ cd ~  
② pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Music Public thinclient_drives  
capture Documents lecture Pictures Templates Videos  
③ pi@raspberrypi:~ $ rmdir lecture  
④ pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Pictures Templates Videos  
capture Documents Music Public thinclient_drives  
pi@raspberrypi:~ $
```

- ① `cd ~` : 사용자(pi) 홈 디렉터리(/home/pi)로 이동
- ② `ls` : 작업 디렉터리의 파일 목록에 `lecture` 디렉터리 존재 확인
- ③ `rmdir lecture`: 디렉터리 `lecture` 삭제
- ④ `ls` : 작업 디렉터리 목록에서 `lecture` 삭제 확인

cat : 파일 내용을 화면에 출력



```
pi@raspberrypi: ~  
① pi@raspberrypi:~ $ echo "test" > from.txt  
② pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Music Public thinclient_drives  
capture Documents from.txt Pictures Templates Videos  
③ pi@raspberrypi:~ $ cat from.txt  
test  
pi@raspberrypi:~ $
```

- ① echo "test" > from.txt : 문자열 "test"로 구성된 파일 from.txt 생성
- ② ls : 작업 디렉터리의 파일 목록에 from.txt 생성 확인
- ③ cat from.txt : 파일 from.txt 내용을 화면에 출력

cp : 파일을 복사하는 명령어 (copy)

```
pi@raspberrypi: ~  
① pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Music Public thinclient_drives  
capture Documents from.txt Pictures Templates Videos  
② pi@raspberrypi:~ $ cat from.txt  
test  
③ pi@raspberrypi:~ $ cp from.txt to.txt  
④ pi@raspberrypi:~ $ ls  
Bookshelf Documents Music Templates Videos  
capture Downloads Pictures thinclient_drives  
Desktop from.txt Public to.txt  
⑤ pi@raspberrypi:~ $ cat to.txt  
test  
pi@raspberrypi:~ $
```

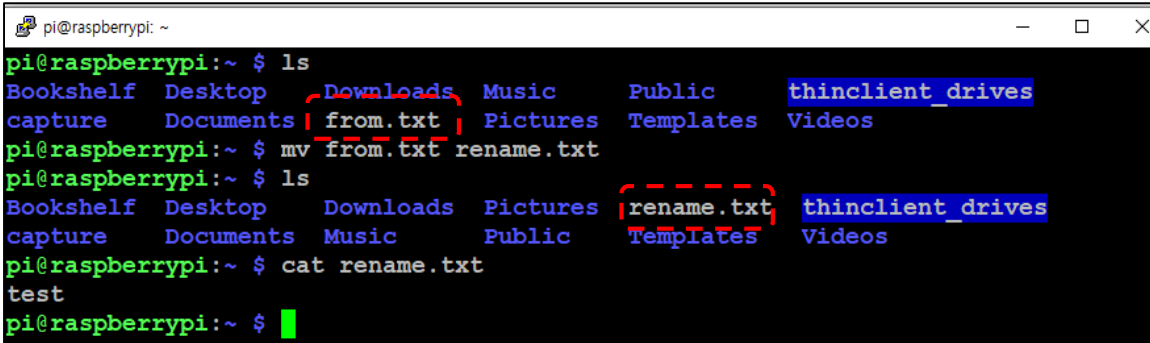
- ① ls : 파일 목록에 from.txt 존재 확인 (없으면, cat 명령으로 생성, 앞 쪽 자료 참조)
- ② cat from.txt : 파일 from.txt 내용을 화면에 출력
- ③ cp from.txt to.txt : from.txt 파일을 to.txt 파일로 복사
- ④ ls : 작업 디렉터리에 to.txt에 파일이 생성된 것을 확인
- ⑤ cat to.txt : 복사된 파일 to.txt 파일 내용이 from.txt 파일과 동일한지 확인

rm : 파일을 삭제하는 명령어 (remove)

```
pi@raspberrypi: ~  
① pi@raspberrypi:~ $ ls  
Bookshelf Documents Music Templates Videos  
capture Downloads Pictures thinclient_drives  
Desktop from.txt Public to.txt  
② pi@raspberrypi:~ $ rm to.txt  
③ pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Music Public thinclient_drives  
capture Documents from.txt Pictures Templates Videos  
pi@raspberrypi:~ $
```

- ① ls : 파일 목록에 to.txt 존재 확인
- ② rm to.txt : 파일 to.txt 내용을 삭제
- ③ ls : 파일 목록에 to.txt 가 삭제된 것을 확인

mv : 파일을 이동하는(파일 이름을 바꾸는) 명령어 (move)



```
pi@raspberrypi: ~  
① pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Music Public thinclient_drives  
capture Documents from.txt Pictures Templates Videos  
② pi@raspberrypi:~ $ mv from.txt rename.txt  
③ pi@raspberrypi:~ $ ls  
Bookshelf Desktop Downloads Pictures rename.txt thinclient_drives  
capture Documents Music Public Templates Videos  
④ pi@raspberrypi:~ $ cat rename.txt  
test  
pi@raspberrypi:~ $
```

- ① ls : 파일 목록에 from.txt 존재 확인
- ② mv from.txt rename.txt : 파일 from.txt를 rename.txt로 이동 (이름을 바꿈)
- ③ ls : 작업 디렉터리에 from.txt에 파일이 rename.txt로 이동한 것을 확인
- ④ cat rename.txt : rename.txt 파일 내용을 확인하여 from.txt 와 동일한 것을 확인

ifconfig : 네트워크 인터페이스의 세부 정보 확인

```

pi@raspberrypi: ~
pi@raspberrypi:~ $ ifconfig
① eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether dc:a6:32:b0:f6:83 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 12 bytes 720 (720.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12 bytes 720 (720.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

② wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.27 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::46f4:ebfd:160a:76a3 prefixlen 64 scopeid 0x20<link>
    ether dc:a6:32:b0:f6:85 txqueuelen 1000 (Ethernet)
    RX packets 3774 bytes 313994 (306.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2034 bytes 331701 (323.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~ $

```

① eth0 : 유선 LAN 인터페이스 eth0의 세부 정보(IP 주소, MAC 주소 등)

② wlan0 : 무선 LAN 인터페이스 wlan0의 세부 정보(IP 주소, MAC 주소 등)

```

pi@raspberrypi: ~
① pi@raspberrypi:~ $ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
② 64 bytes from 8.8.8.8: icmp_seq=1 ttl=106 time=63.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=106 time=60.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=106 time=60.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=106 time=61.3 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=106 time=60.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=106 time=63.4 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=106 time=61.2 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=106 time=63.8 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=106 time=61.0 ms
^C
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 21ms
rtt min/avg/max/mdev = 60.356/61.729/63.843/1.341 ms
pi@raspberrypi:~ $

```

- ① ping 8.8.8.8 : google DNS 서버 8.8.8.8의 연결 상태 확인
- ② 8.8.8.8로 64Byte 데이터를 보내고 받았다. 정상적으로 연결됨

Yum / apt : 패키지 관리 명령어

(yum) Redhat 계열에서 사용하는 패키지 관리 명령어

- yum install 패키지_이름 # 패키지를 설치
- yum remove 패키지_이름 # 패키지를 삭제
- yum update 패키지_이름 # 패키지를 업데이트
- yum list # 설치된 패키지 확인

(apt-get) debian 계열에서 사용하는 패키지 관리 명령어

- apt-get install 패키지_이름 # 패키지를 설치
- apt-get update # 패키지 업데이트 정보를 얻어옴
- apt-get upgrade # update 정보에 새로운 버전이 있으면 업데이트
- apt-get remove 패키지_이름 # 패키지를 삭제
- apt-get —purge remove 패키지_이름 # 패키지를 설정까지 함께 삭제
- apt —installed list # 설치된 패키지 확인

※ sudo 명령어 : root 권한으로 명령어 이행(Substitute user do)

Yum / apt : 패키지 관리 명령어 (설치는 관리자 권한)

```
pi@raspberrypi: ~  
① pi@raspberrypi:~ $ apt --installed list gcc  
Listing... Done  
gcc/stable,now 4:8.3.0-1+rpi2 armhf [installed,automatic]  
② pi@raspberrypi:~ $ apt --installed list telnet  
Listing... Done  
③ pi@raspberrypi:~ $ sudo apt-get install telnet  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
  telnet  
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 60.7 kB of archives.  
After this operation, 131 kB of additional disk space will be used.  
Get:1 http://ftp.harukasan.org/raspbian/raspbian buster/main armhf telnet armhf  
  0.17-41.2 [60.7 kB]  
Fetched 60.7 kB in 1s (46.2 kB/s)  
Selecting previously unselected package telnet.  
(Reading database ... 95552 files and directories currently installed.)  
Preparing to unpack .../telnet_0.17-41.2_armhf.deb ...  
Unpacking telnet (0.17-41.2) ...  
Setting up telnet (0.17-41.2) ...  
update-alternatives: using /usr/bin/telnet.netkit to provide /usr/bin/telnet (t  
elnet) in auto mode  
Processing triggers for man-db (2.8.5-2) ...  
pi@raspberrypi:~ $
```

- ① apt --installed installed gcc : gcc 컴파일러가 설치된 것을 확인 (설치됨: installed, automatic)
- ② apt --installed installed telnet : telnet 프로그램이 설치된 것을 확인 (설치 안됨)
- ③ sudo apt-get install telnet : 관리자 권한으로 telnet 프로그램 설치

nano : 문서 편집기

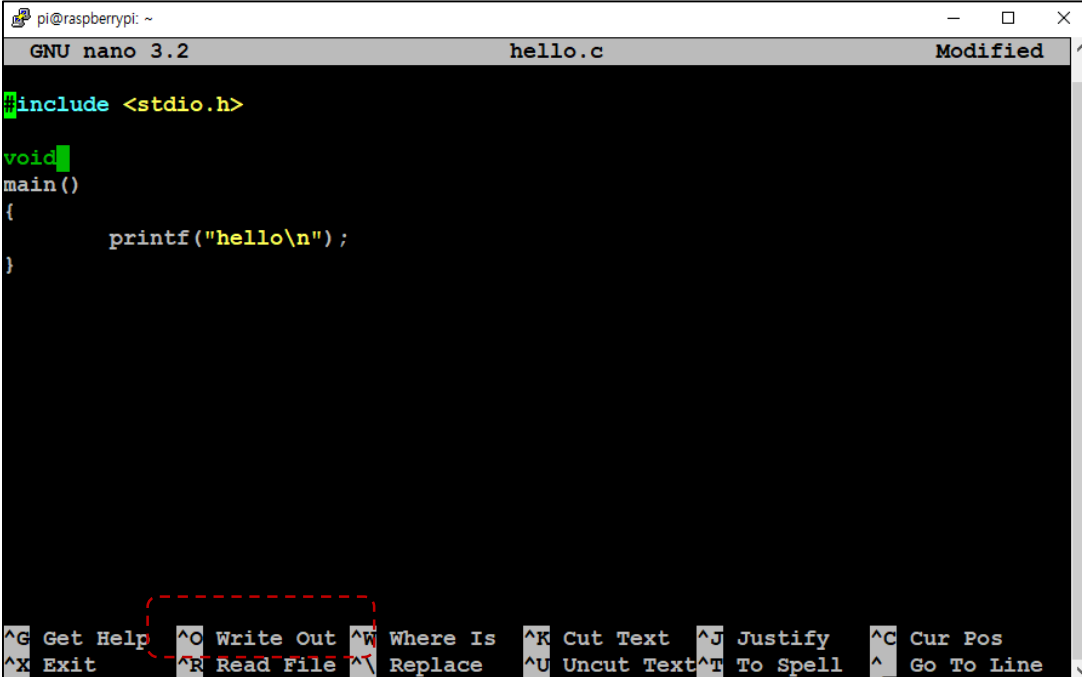
```
pi@raspberrypi: ~
① pi@raspberrypi:~ $ nano hello.c
```

```
pi@raspberrypi: ~
GNU nano 3.2      hello.c

[ New File ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

① nano hello.c : hello.c 파일을 nano 에디터로 편집합니다.

nano : 패키지 관리 명령어 (설치는 관리자 권한)



The screenshot shows a terminal window with the nano text editor open. The editor is editing a file named `hello.c`. The code visible in the editor is:

```
#include <stdio.h>

void
main()
{
    printf("hello\n");
}
```

At the bottom of the screen, the nano help menu is displayed. A red dashed box highlights the `^O Write Out` option, which is used to save the file. The help menu also includes other options like `^G Get Help`, `^X Exit`, `^R Read File`, `^W Where Is`, `^K Cut Text`, `^U Uncut Text`, `^J Justify`, `^T To Spell`, `^C Cur Pos`, and `^_ Go To Line`.

① `^O` : 편집이 완료 되면 `^O`를 눌러 저장한다.

nano : 패키지 관리 명령어 (설치는 관리자 권한)



The screenshot shows the GNU nano 3.2 text editor running on a Raspberry Pi. The editor window has a title bar that says "GNU nano 3.2" and "hello.c". The code being edited is a simple C program:

```
#include <stdio.h>

void
main()
{
    printf("hello\n");
}
```

At the bottom of the editor, there is a status bar that says "[Read 7 lines]". Below the status bar, there is a list of command shortcuts:

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos
^X Exit	^R Read File	^_ Replace	^U Uncut Text	^T To Spell	^_ Go To Line

A red dashed box highlights the first two shortcuts: **^G** Get Help and **^X** Exit. A red circle with the number 1 is next to the **^X** Exit shortcut.

① **^x** : **^x**를 눌러 편집기를 종료한다.

nano : 패키지 관리 명령어 (설치는 관리자 권한)

```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ nano hello.c  
① pi@raspberrypi:~ $ ls  
Bookshelf Documents Music rename.txt Videos  
capture Downloads Pictures Templates  
Desktop hello.c Public thinclient_drives  
pi@raspberrypi:~ $
```

① ls : 편집한 파일 hello.c가 생성된 것을 확인

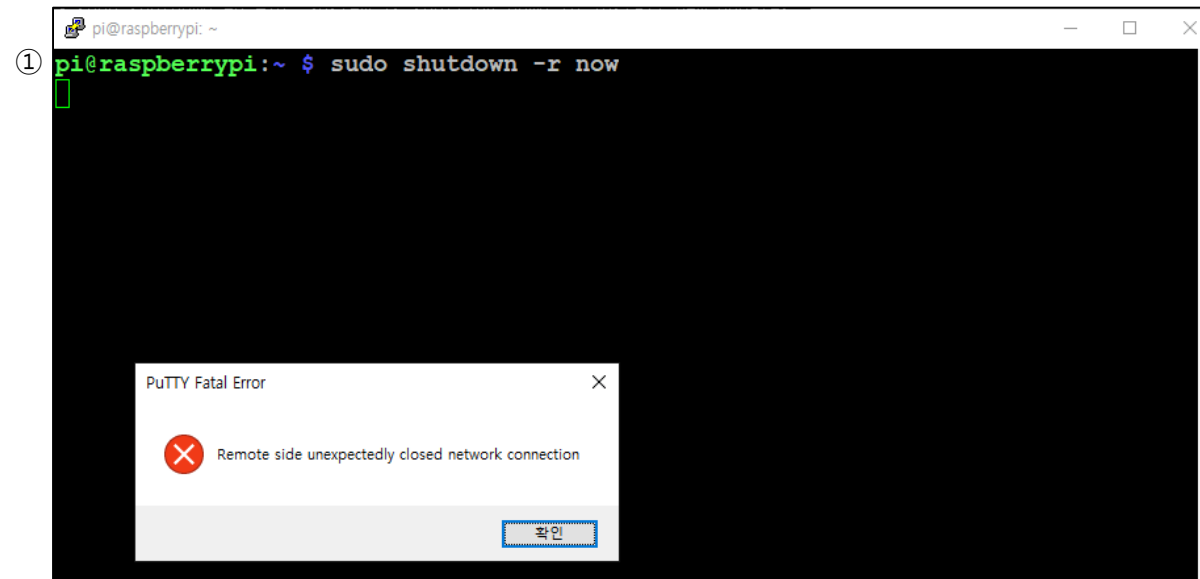
gcc : 컴파일



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ ls  
Bookshelf Documents Music rename.txt Videos  
capture Downloads Pictures Templates  
Desktop { hello.c } Public thinclient_drives  
① pi@raspberrypi:~ $ gcc hello.c -o hello  
② pi@raspberrypi:~ $ ls  
Bookshelf Documents hello.c Public thinclient_drives  
capture Downloads Music rename.txt Videos  
Desktop { hello } Pictures Templates  
③ pi@raspberrypi:~ $ ./hello  
hello  
pi@raspberrypi:~ $
```

- ① gcc hello.c -o hello : hello.c 파일을 컴파일한 후, hello라는 이름의 실행파일 생성
- ② ls : 컴파일 후, 실행파일 hello 생성 확인
- ③ ./hello : 실행파일 hello 실행

shutdown -r : 시스템 재부팅(관리자 권한 설치)

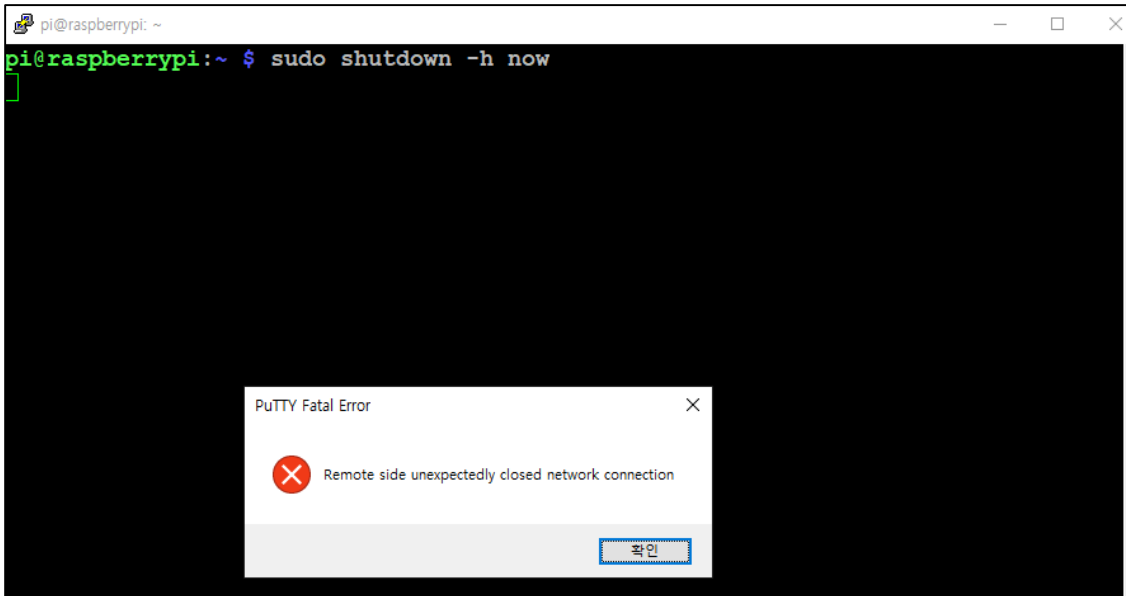


① `sudo shutdown -r now` : 즉시 시스템 재부팅

shutdown -h : 시스템 종료(관리자 권한 설치)

①

```
pi@raspberrypi:~ $ sudo shutdown -h now
```



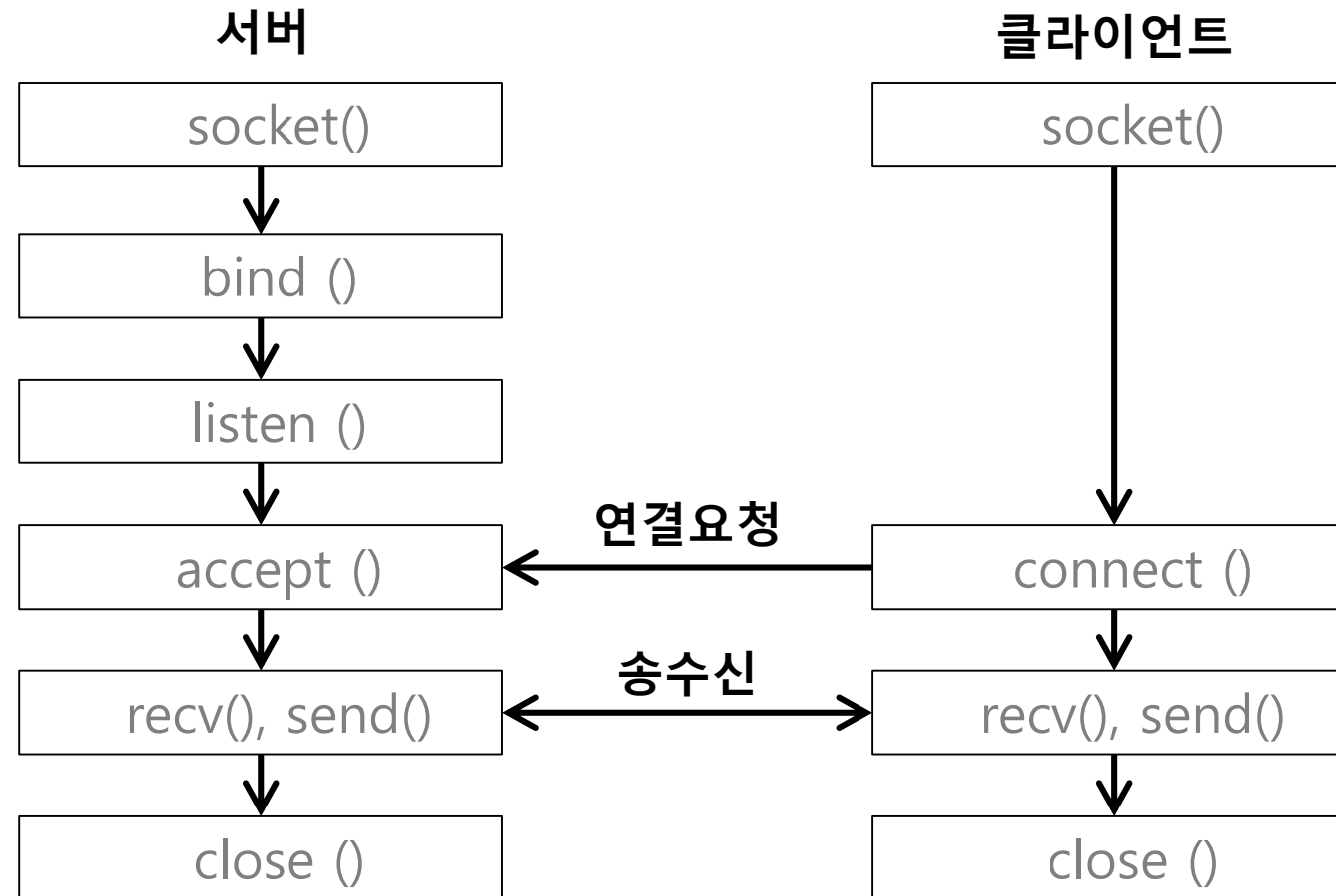
① `sudo shutdown -h now` : 즉시 시스템 종료

- 클라이언트 및 서버 소켓프로그래밍을 배운다.
- 서버 소켓 프로그램 멀티 프로세싱을 배운다.

[실습과제 1]

클라이언트 및 서버 소켓프로그램 (1대 1 통신)

TCP 소켓 프로그램



TCP 통신으로 1:1 통신 지원

TCP 소켓 프로그램

파일명 : server.py

```
import socket
s = socket.socket()
host = '127.0.0.1' # server IP address
port = 9000
s.bind((host, port))
s.listen(50)
while True:
    c, addr = s.accept()
    print('Got connection from', addr)
    c.send(b'Thank you for connecting')
    c.close()
s.close()
```

파일명 : client.py

```
import socket
s = socket.socket()
host = '127.0.0.1'
port = 9000
s.connect((host, port))
print(s.recv(1024))
s.close
```

```
> python server.py
> python client.py
```

실행

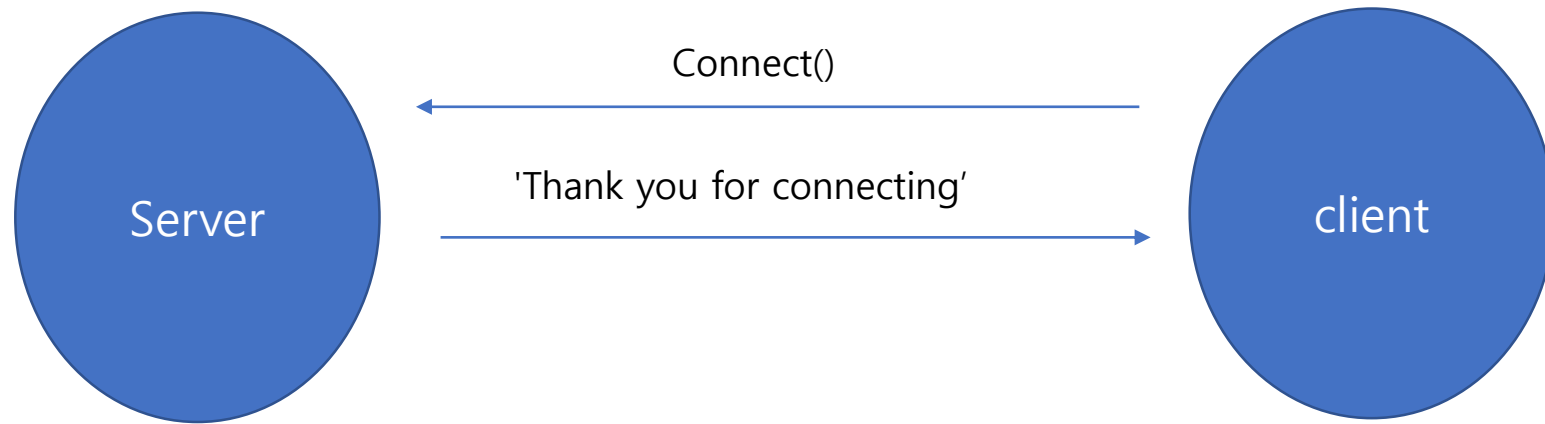
Terminal 1

```
> python server.py
```

Terminal 2

```
> python client.py
```

질문 : 서버와 클라이언트 중 어느 것을 먼저 실행 시켜야 할까요?



조금 바꿔 볼까요?

[실습과제 2]

클라이언트 및 서버 소켓프로그램 (1대 1 통신)

TCP 소켓 프로그램

파일명 : server.py

```
import socket
s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
host = '0.0.0.0'    # server IP address
port = 8000
s.bind((host, port))
s.listen(50)
while True:
    c, addr = s.accept()
    print ('Got connection from', addr)

    msg = ''
    while True:
        data = c.recv(2048)
        msg = data.decode()
        if msg == 'bye':
            break
        print ("\"{}\" from client [{}]".format(msg, addr))
        c.send(bytes(msg, 'UTF-8'))
    print('Client at', addr, 'disconnected..')
    c.close()
s.close()
```

TCP 소켓 프로그램

파일명 : client.py

```
import socket
s = socket.socket()
host = '127.0.0.1'
port = 8000
s.connect((host, port))
s.sendall(bytes('This is From c', 'UTF-8'))

While True:
    in_data = s.recv(1024)
    print('From Server: ', in_data.decode())

    out_data = input()
    s.sendall(bytes(out_data, 'UTF-8'))
    if out_data == 'bye':
        break

s.close
```

```
> python server.py
> python client.py
```


실행

Terminal 1

```
> python server.py
```

Terminal 2

```
> python client.py
```

Terminal 3

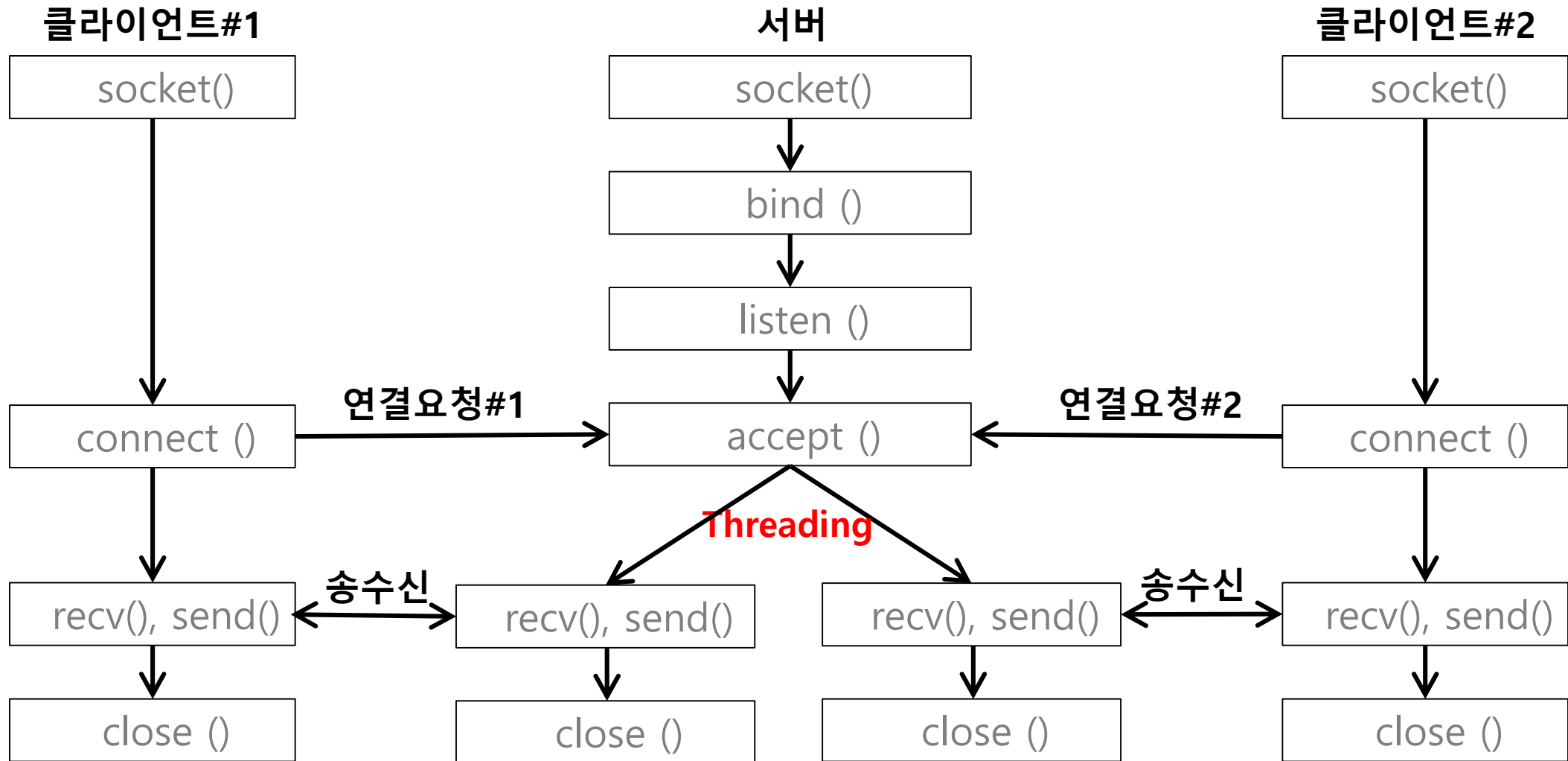
```
> python client.py
```

질문 : 여러 클라이언트가 동시에 서버에 접속할 경우 어떤 현상이 발생하는가?

[실습과제 3]

다중 클라이언트 접속 서버 소켓프로그램 (n대 1 통신)

TCP 소켓서버 확장



TCP 통신으로 1:N 통신 지원

TCP 소켓서버 확장

파일명 : mserver.py

```

import socket, threading
class ClientThread(threading.Thread):
    def __init__(self, addr, c):
        threading.Thread.__init__(self)
        self.csocket = c
    print ("New connection added: ", addr)
    def run(self):
        print ("Connection from : ", addr)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
                break
            print ("\n"{}"\n" from client [{}]"
                    .format(msg, addr))
            self.csocket.send(bytes(msg, 'UTF-8'))
        print ("Client at ", addr , " disconnected...")

```

```

host = "127.0.0.1"
port = 8080
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,
             socket.SO_REUSEADDR, 1)
s.bind((host, port))
print("Server started")
print("Waiting for client request..")
while True:
    s.listen(1)
    c, addr = s.accept()
    n = ClientThread(addr, c)
    n.start()

```

실행

> python mserver.py

복수 클라이언트에 응답할 수 있도록 thread 화

TCP 클라이언트

```
import socket
host = "127.0.0.1"
port = 8080
s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
s.connect((host, port))
s.sendall(bytes("This is from c",'UTF-8'))
while True:
    in_data = s.recv(1024)
    print("From Server :",in_data.decode())
    out_data = input()
    s.sendall(bytes(out_data,'UTF-8'))
    if out_data=='bye':
        break
s.close()
```

파일명 : mclient.py

실행

```
> python client.py
```

키보드 입력 메시지를 전송하고 "bye"를 입력하면 연결 종료

[실습과제 4]

UDP 소켓프로그래밍 프로그래밍

TCP v.s. UDP

	TCP	UDP
신뢰성 reliability	<ul style="list-style-type: none"> ● 오류제어 (사본 보유 및 재전송) ● 흐름제어 (flow control) 	<ul style="list-style-type: none"> ● 오류제어 기능 없음 ● 흐름제어 기능 없음
	신뢰성이 높다 (reliable)	신뢰성이 낮다 (unreliable)
연결 connection	<ul style="list-style-type: none"> ● 경로 설정 (path setup) ● 자료의 순서 유지 	<ul style="list-style-type: none"> ● 경로 설정 절차가 없음 ● 자료의 순서가 뒤바뀔 수 있음
	연결형(connection-oriented)	비연결형(connection-less)
비용 cost	<ul style="list-style-type: none"> ● more space, more time 	<ul style="list-style-type: none"> ● less space, less time
	고비용	저비용
응용 분야 application		<ul style="list-style-type: none"> ● 한 패킷으로 서비스(DNS, time) ● 안전한 LAN 환경 (NFS)
	TCP	UDP

TCP v.s. UDP

TCP 소켓 생성

```
t1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
```

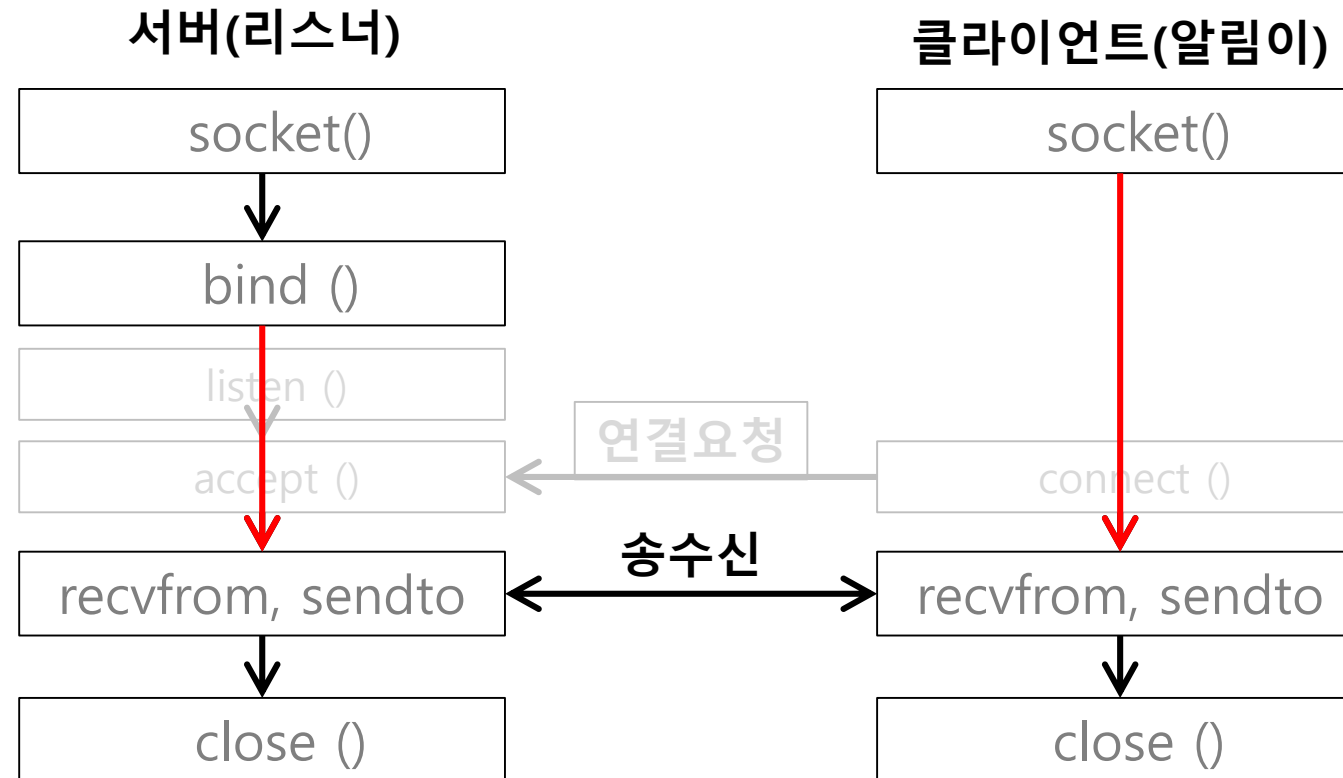
```
t2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

UDP 소켓 생성

```
u1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
```

```
u2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```


UDP 소켓



TCP 통신으로 1:1 통신 지원

UDP 소켓 프로그램

파일명 : listener.py

```
import socket
host = "0.0.0.0"
port = 20001

u = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
u.bind((host, port))
print("UDP server up and listening")
while(True):
    c, addr = u.recvfrom(1024)
    print("Message [{}] from {}".format(c, addr))
    u.sendto(b'Hello UDP Client', addr)
```

실행

> python listener.py

UDP 소켓 프로그램

파일명 : notify.py

```
import socket
name = socket.gethostname()
host = "127.0.0.1"
port = 20001

u = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
u.settimeout(5.0)
u.sendto("Hello UDP Listener".encode('UTF-8'),
        (host, port))
c, addr = u.recvfrom(1024)
print("Message from Server {}".format(addr))
Print('Recv Data :'.format(c.decode()))
```

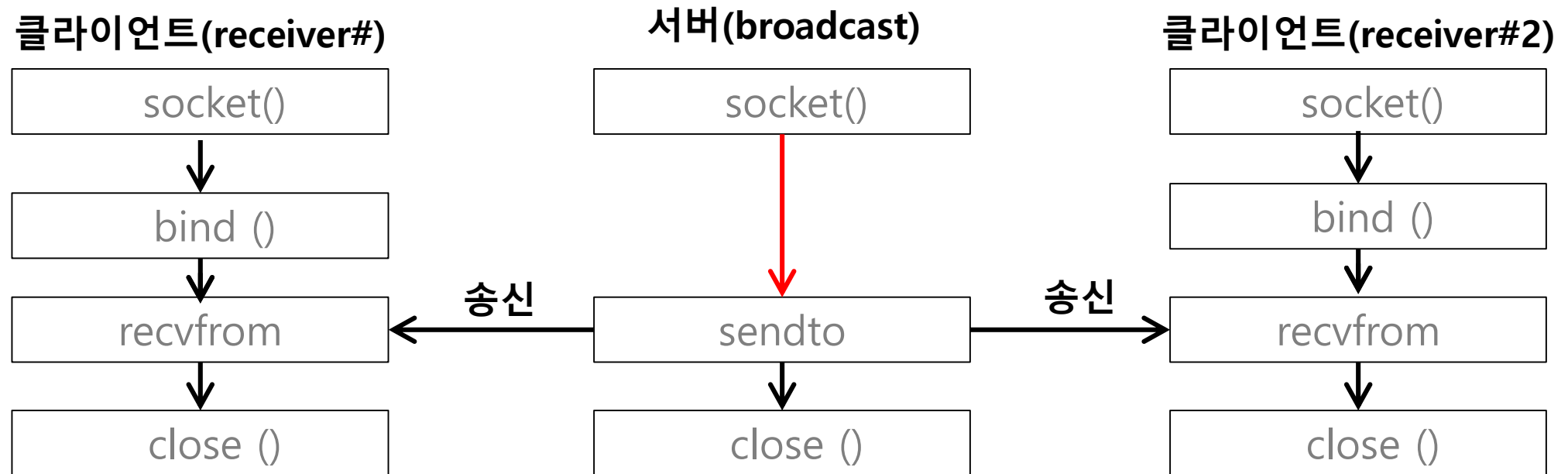
실행

```
> python notify.py
```

[실습과제 5]

UDP 브로드캐스팅 소켓프로그래밍 프로그래밍

UDP 브로드캐스트 - 메시지 방송



UDP 통신으로 다수 컴퓨터에 메시지 전송 (응답 기다리지 않음)

UDP 브로드캐스트 소켓 프로그램 - 서버

파일명 : broadcaster.py

```
import socket
import time
host = '<broadcast>'
port = 37020
u = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
u.settimeout(0.2)
print("UDP Broadcast server up and announce message")
while(True):
    u.sendto(b"broadcast message", (host, port))
    print("Broadcast message")
    time.sleep(1)
```

실행

```
> python broadcaster.py
```

UDP 브로드캐스트 - 클라이언트

파일명 : receiver.py

```
import socket

host = ""
port = 37020
u = socket.socket(family=socket.AF_INET,
                  type=socket.SOCK_DGRAM,
                  proto=socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET,
             socket.SO_BROADCAST, 1)
u.bind((host, port))
print("Broadcast receiver start")
while True:
    c, addr = u.recvfrom(1024)
    print("Message [{}] from Server {}".format(c, addr))
```

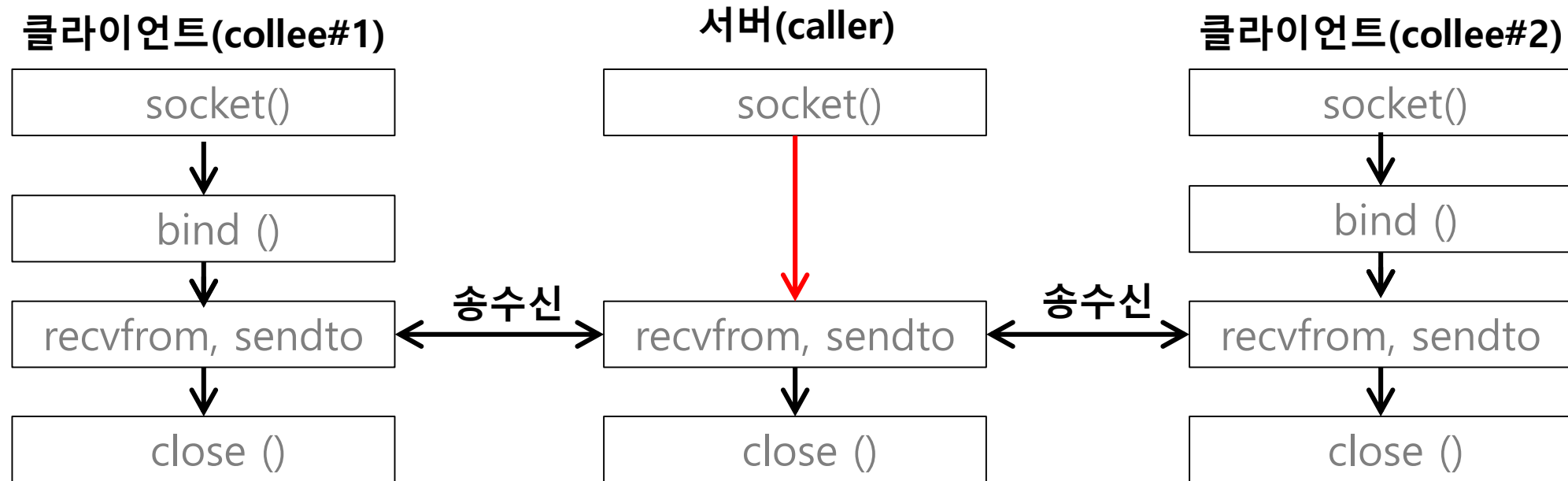
실행

```
> python receiver.py
```

UDP 소켓 클라이언트, 메시지 수신

UDP broadcast 확장

UDP 1:N caller, collee



UDP 1:N 통신으로 네트워크 주변 장비 확인

UDP 브로드캐스트 소켓 프로그램 - 서버

파일명 : caller.py

```
import socket
import time
host = '<broadcast>'
port = 37021
u = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
u.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
u.settimeout(3)
print("UDP Broadcast to check devices")
while(True):
    u.sendto(b"broadcast message", (host, port))
    print("Braodcast message")
    try:
        while True:
            c, addr = u.recvfrom(1024)
            print("Message from client {}".format(addr))
    except:
        pass
    time.sleep(1)
```

실행

```
> python caller.py
```

UDP 브로드캐스트 - 클라이언트

파일명 : callee.py

```
import socket
host = ""
port = 37021
u = socket.socket(family=socket.AF_INET,
                  type=socket.SOCK_DGRAM,
                  proto=socket.IPPROTO_UDP)

u.setsockopt(socket.SOL_SOCKET,
             socket.SO_BROADCAST, 1)
u.bind((host, port))
print("Answer to caller")
while True:
    c, addr = u.recvfrom(1024)
    print("Call from Server {}".format(addr))
    u.sendto(b"receive message", addr)
```

실행

```
> python callee.py
```

1. 웹서버의 동작 절차
2. 라즈베리파이 OS설치 및 환경 설정 방법
3. 라즈베리파이로 무선공유기 만들기
4. 디지털 액자 만들기
5. GPIO를 사용하여 정보를 표시하는 방법
6. GPIO를 사용하여 데이터를 수집하는 방법
7. 원격 장비의 데이터 수집과 전송 방법
8. 대형 디지털 시계 만들기
9. 다양한 센서 활용 방법
10. 사물인터넷 엣지 서버 만들기
11. 부저, 스위치 등을 이용하는 방법
12. 응용프로그램 자동 실행방법

출석 : 20%

중간 평가 : 40% (중간고사 30%, 수업참여도 10%)

기말 평가 : 40% (기말고사 30%, 수업참여도 10%)