

네트워크프로그래밍-4주

소켓프로그래밍-2

정인환교수

4주 강의 내용

- ▶ Windows Socket 프로그래밍 환경
- ▶ Time Client / Server 상세 설명 및 응용
 - TCP Time Client/Server 응용
- ▶ 소켓프로그래밍 예2 - echo client / server
 - echo client/server 기본/응용1
 - 응용2,3 - Server가 대문자/소문자 변환
- ▶ 과제
 - echo client/server 응용
 - Client 에서 대/소문자 변환 Menu 사용
 - Application Protocol 만들기
 - Login 기능 추가하기
 - login protocol 만들기

Windows Socket 프로그래밍 환경

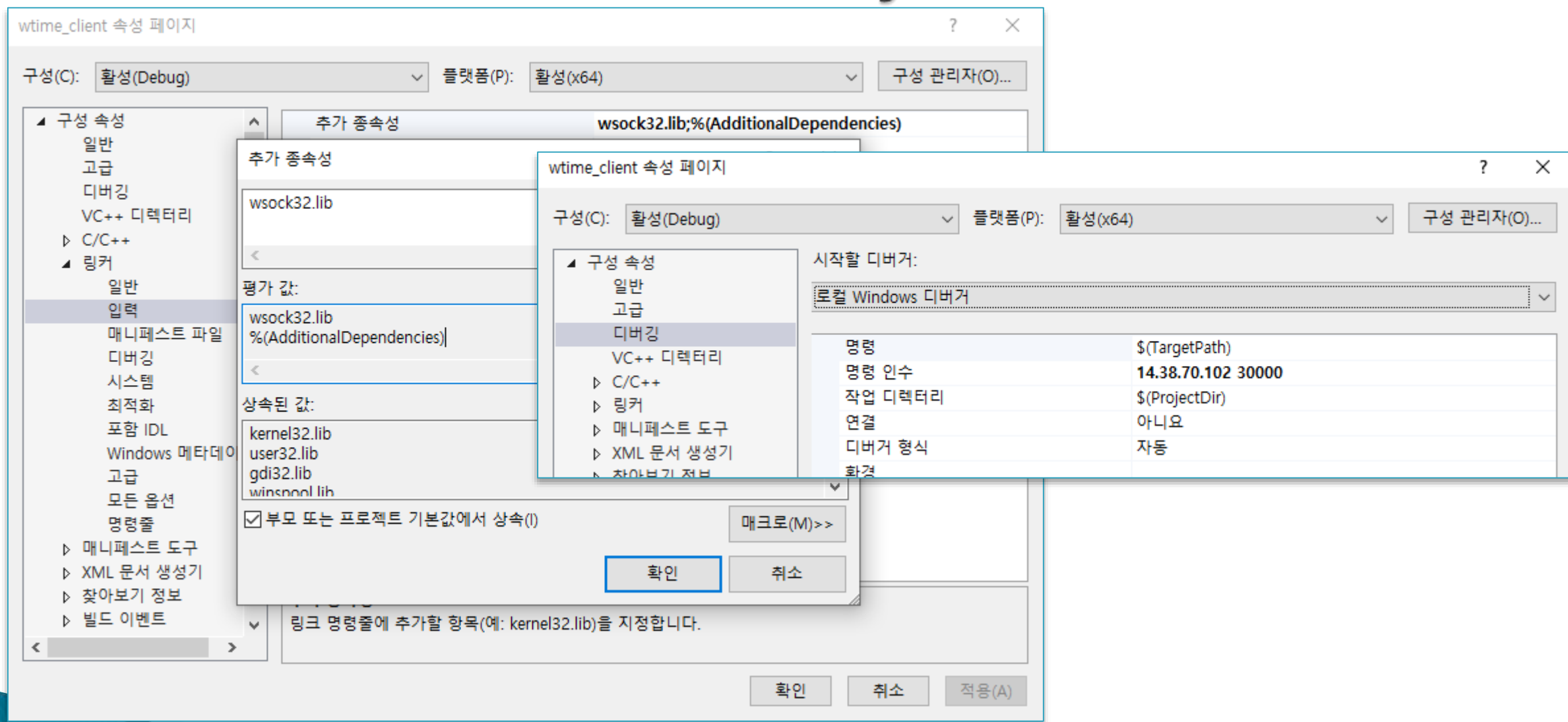
- ▶ Linux << 99% 그대로 사용 >> Windows
- ▶ Linux `time_client.c` >> Windows `wtime_client.c`
 - 헤더파일 필요

```
#include <winsock.h>
#include <signal.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
```
- ▶ Library 환경 설정
 - `wsock32.lib` 링커에 추가
- ▶ 프로그램 순서
 - 시작시
 - `WSAStartup()`
 - `Winsock.dll`을 초기화
 - 종료시
 - 소켓닫을때
 - `closesocket()`
 - `WSACleanup()`
 - `Winsock.dll`의 사용 종료

```
WORD version = MAKEWORD(1,1);
WSADATA wsadata;
WSAStartup(version, &wsadata);
```

```
closesocket();
WSACleanup();
exit(0);
```

Windows Visual Studio Project 설정



```

/* Linux
파일명 : time_client.c
기능 : time 서비스를 요구하는
TCP(연결형) 클라이언트
사용법 : time_client[ip][port] //
default는 127.0.0.1 30000
*/
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/signal.h>
#include <unistd.h>

#define BUF_LEN 128
#define TIME_SERVER "127.0.0.1"
#define TIME_PORT "30000"

void main(int argc, char *argv[]) {
    int sock;
    struct sockaddr_in server;
    char *haddr;
    char buf[BUF_LEN+1] = {0};

    sock = socket(...)

    closesocket(sock);
}

```

Linux time_client.c



```

/* Windows
파일명 : wtime_client.c
기능 : time 서비스를 요구하는 TCP(연결형)
클라이언트
사용법 : time_client[ip][port] // default는
127.0.0.1 30000
*/
#include <winsock.h>
#include <signal.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

WSADATA wsadata;
int main_socket;

void exit_callback(int sig)
{
    closesocket(main_socket);
    WSACleanup();
    exit(0);
}

void init_winsock()
{
    WORD sversion;
    u_long iMode = 1;

    // winsock 사용을 위해 필수적임
    signal(SIGINT, exit_callback);
    sversion = MAKEWORD(1, 1);
    WSStartup(sversion, &wsadata);
}

```

```

#define BUF_LEN 128
#define TIME_SERVER "127.0.0.1"
#define TIME_PORT "30000"

void main(int argc, char* argv[]) {
    int sock;
    struct sockaddr_in server;
    char* haddr;
    char buf[BUF_LEN + 1] = { 0 };
    char* ip_addr = TIME_SERVER, * port_no = TIME_PORT;

    init_winsock();

    if (argc == 3) {
        ip_addr = argv[1];
        port_no = argv[2];
    }
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("can't create socket\n");
        exit(0);
    }
    main_socket = sock; // 추가

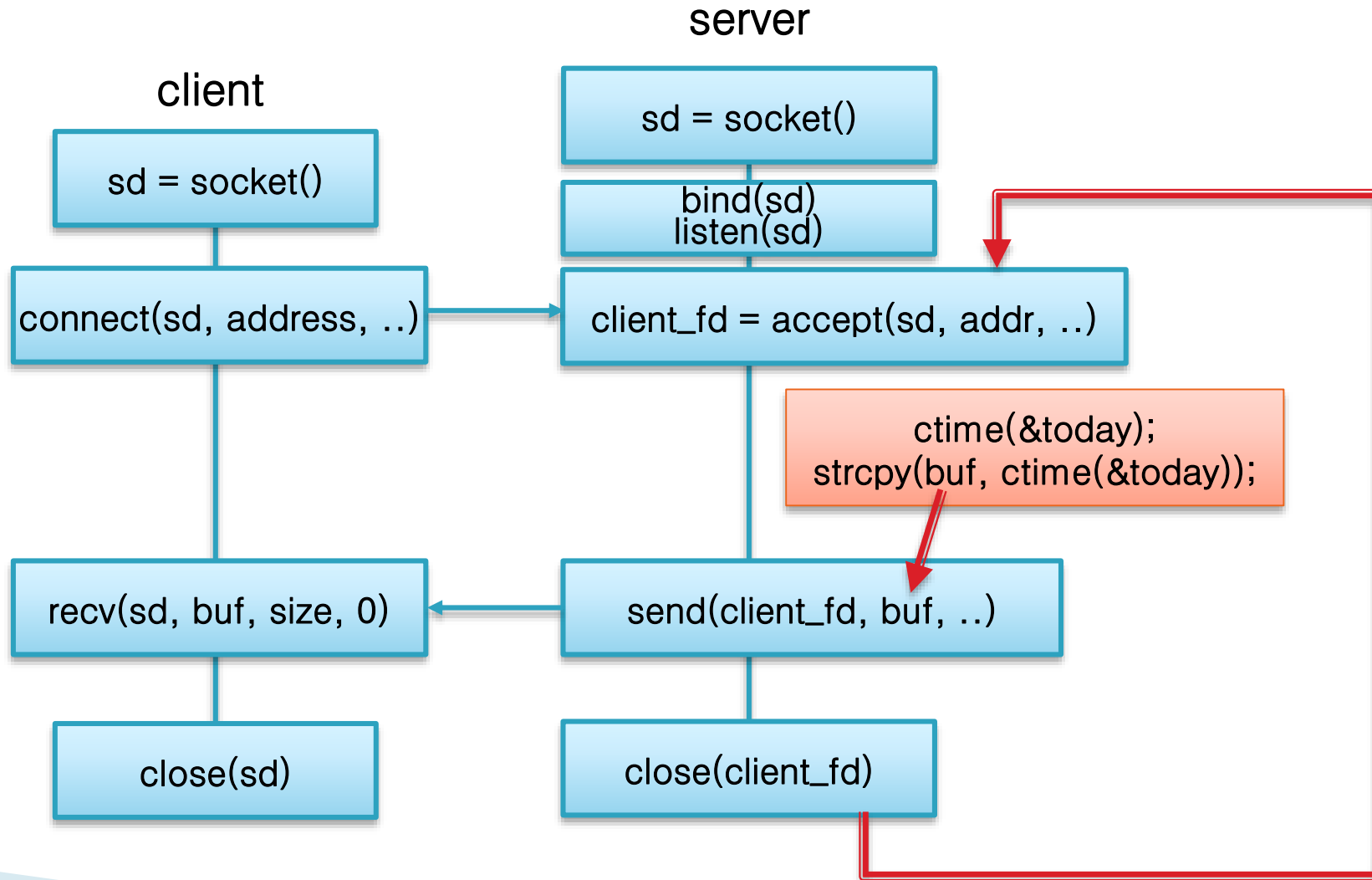
    // 생략

    printf("Time information from server is %s", buf);

    closesocket(sock);
    return(0);
}

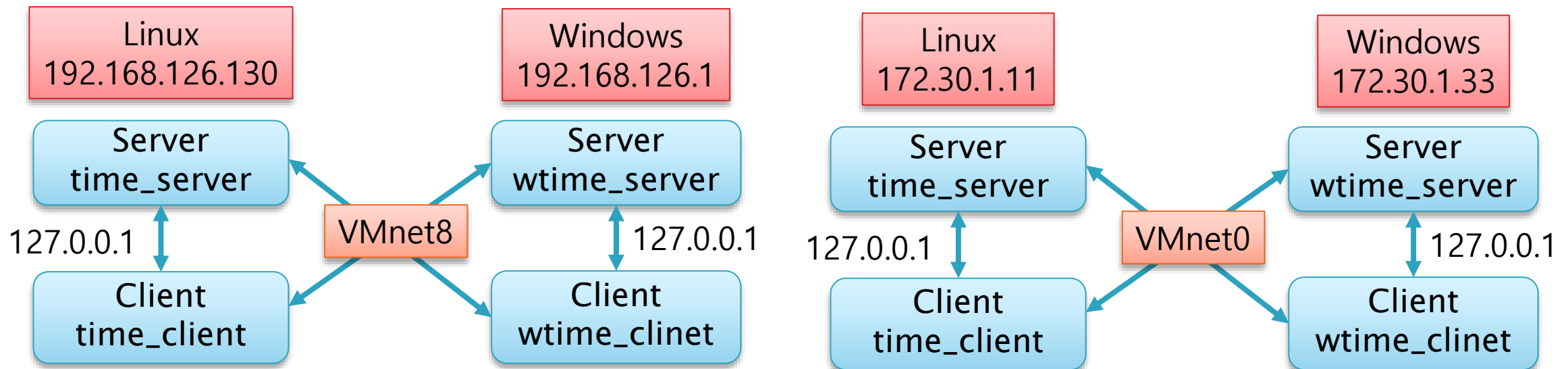
```

소켓프로그래밍 예1-1 – TCP time client/server



소켓프로그래밍 예1 – TCP/UDP time client/server

- ▶ Linux Client / Linux Server
- ▶ Windows Client / Windows Server
- ▶ Windows Client / Linux Server
- ▶ Linux Client / Windows Server



소켓프로그래밍 예1 – TCP time client/server

▶ time_server.c

```
# define TIME_PORT    "30000"
main (int argc, char *argv[])
{
    int sock, sock2;
    struct sockaddr_in server, client;
    int len;
    char buf [256];
    time_t today;

    sock = socket (AF_INET, SOCK_STREAM, 0);
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl (INADDR_ANY);
    server.sin_port = htons (atoi(TIME_PORT));
    bind (sock, (struct sockaddr *)&server, sizeof (server));
    listen (sock, 5);
    while (1) {
        sock2 = accept (sock, (struct sockaddr *)&client, &len);
        time (&today);
        strcpy (buf, ctime (&today));
        send (sock2, buf, strlen (buf) + 1, 0);
        close (sock2);
    }
}
```

TCP SOCK_STREAM
UDP SOCK_DGRAM

INADDR_ANY = 0.0.0.0
모든 연결에서 대기한다.

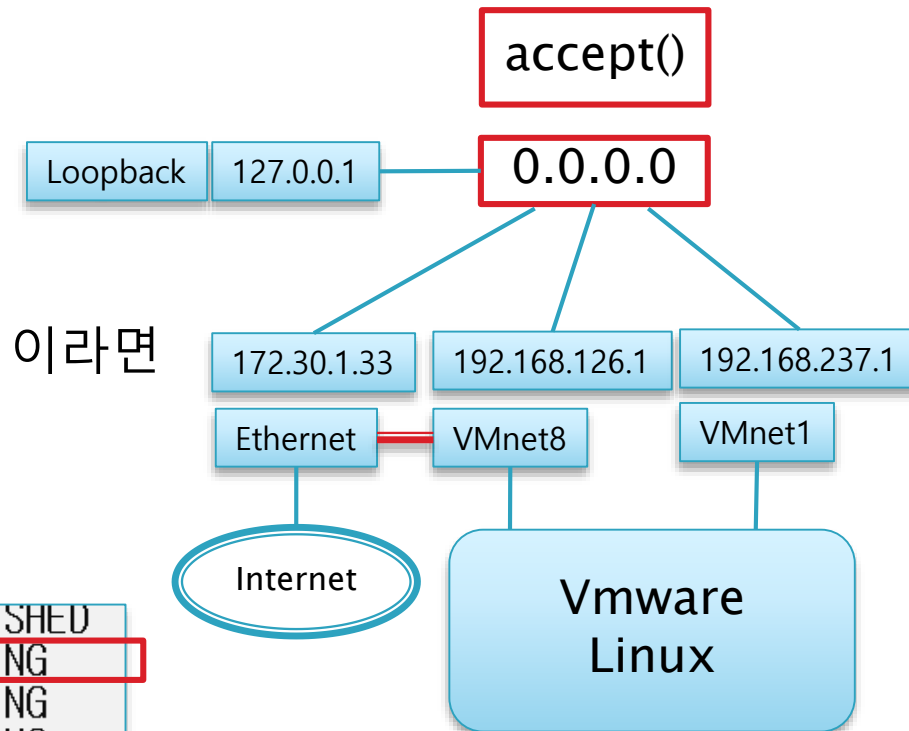
bind(), accept() 함수에서 기다리는 IP 주소 결정

- ▶ Network Interface가 2개 이상인 경우
 - ipconfig 로 확인 가능
 - 이더넷(172.30.1.33), VMnet1(192.168.237.1), VMnet8(192.168.126.1), Loopback(127.0.0.1)
- ▶ `server.sin_addr.s_addr = htonl(INADDR_ANY); // “0.0.0.0”`
 - 모든 Network Interface에서 `accept()` 기다린다.
 - `Netstat -na` 로 확인
 - client 는 위의 4가지 주소 모두 접속 가능

TCP	0.0.0.0:21300	0.0.0.0:0	LISTENING
TCP	0.0.0.0:30000	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49152	0.0.0.0:0	LISTENING

- ▶ `INADDR_ANY` 아닌 실제 IP 주소
 - `#define TIME_SERVER “0.0.0.0”` 대신 “172.30.1.33” 이라면
 - `ip_addr = TIME_SERVER`
 - `server.sin_addr.s_addr = inet_addr(ip_addr);`
 - client는 172.30.1.33 30000 만 접속 가능

TCP	172.30.1.33:10182	157.240.215.63:443	ESTABLISHED
TCP	172.30.1.33:30000	0.0.0.0:0	LISTENING
TCP	192.168.126.1:139	0.0.0.0:0	LISTENING



소켓프로그래밍 예1-1 – TCP time client/server

▶ time_client.c

```
#define TIME_SERVER    "127.0.0.1"
#define TIME_PORT      30000

void main(int argc, char *argv[]) {
    int sock;
    struct sockaddr_in server;
    char buf[BUF_LEN+1] = {0};
    char *ip_addr = TIME_SERVER, *port_no = TIME_PORT;
    if (argc == 3) {
        ip_addr = argv[1];
        port_no = argv[2];
    }
    sock = socket(AF_INET, SOCK_STREAM, 0);

    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr(ip_addr); // "127.0.0.1" → 0x7f000001
    server.sin_port = htons(atoi(port_no));

    connect(sock, (struct sockaddr *)&server, sizeof(server));

    if (recv(sock, buf, sizeof(buf), 0) == -1)
        exit(1);
    printf("Time information from server is %s", buf);
    close(sock);
}
```

Windows time_client <127.0.0.1> time_server

*Adapter for loopback traffic capture (tcp and port 30000)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	14332 → 30000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_
2	0.000081	127.0.0.1	127.0.0.1	TCP	56	30000 → 14332 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 W
3	0.000113	127.0.0.1	127.0.0.1	TCP	44	14332 → 30000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
4	0.003420	127.0.0.1	127.0.0.1	TCP	70	30000 → 14332 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=26
5	0.003463	127.0.0.1	127.0.0.1	TCP	44	14332 → 30000 [ACK] Seq=1 Ack=27 Win=2619648 Len=0
6	0.003508	127.0.0.1	127.0.0.1	TCP	44	30000 → 14332 [FIN, ACK] Seq=27 Ack=1 Win=2619648 Len=0
7	0.003529	127.0.0.1	127.0.0.1	TCP	44	14332 → 30000 [ACK] Seq=1 Ack=28 Win=2619648 Len=0
8	0.003640	127.0.0.1	127.0.0.1	TCP	44	14332 → 30000 [FIN, ACK] Seq=1 Ack=28 Win=2619648 Len=0
9	0.003687	127.0.0.1	127.0.0.1	TCP	44	30000 → 14332 [ACK] Seq=28 Ack=2 Win=2619648 Len=0

< >

> Frame 4: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_Loopback, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 30000, Dst Port: 14332, Seq: 1, Ack: 1, Len: 26

▼ Data (26 bytes)

Data: 536174205365702031392031333a34323a35322032303230...

[Length: 26]

0000	02 00 00 00 45 00 00 42 88 d6 40 00 80 06 00 00E..B..@....
0010	7f 00 00 01 7f 00 00 01 75 30 37 fc 4b 15 a8 19u07.K...
0020	8a 9a c5 ec 50 18 27 f9 7a 40 00 00 53 61 74 20P.'..z@..Sat
0030	53 65 70 20 31 39 20 31 33 3a 34 32 3a 35 32 20	Sep 19 1 3:42:52
0040	32 30 32 30 0a 00	2020..

Data (data.data), 26 byte(s)

Packets: 9 · Displayed: 9 (100.0%) · Dropped: 0 (0.0%) | Profile: Default

Windows time_client(192.168.126.1) <VMnet8> Linux time_server(192.168.126.130)

Capturing from VMware Network Adapter VMnet8 (tcp and port 30000)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.126.1	192.168.126.130	TCP	66	5572 → 30000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000168	192.168.126.130	192.168.126.1	TCP	66	30000 → 5572 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1
3	0.000255	192.168.126.1	192.168.126.130	TCP	54	5572 → 30000 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
4	0.000550	192.168.126.130	192.168.126.1	TCP	80	30000 → 5572 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=26
5	0.000620	192.168.126.130	192.168.126.1	TCP	60	30000 → 5572 [FIN, ACK] Seq=27 Ack=1 Win=64256 Len=0
6	0.000706	192.168.126.1	192.168.126.130	TCP	54	5572 → 30000 [ACK] Seq=1 Ack=28 Win=1051136 Len=0
7	0.000777	192.168.126.1	192.168.126.130	TCP	54	5572 → 30000 [FIN, ACK] Seq=1 Ack=28 Win=1051136 Len=0
8	0.000861	192.168.126.130	192.168.126.1	TCP	60	30000 → 5572 [ACK] Seq=28 Ack=2 Win=64256 Len=0

> Frame 4: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface \Device\NPF_{60640543-7BC8-4A60-A5A7-A657E328C1FF}, id 0

> Ethernet II, Src: VMware_9b:0d:00 (00:0c:29:9b:0d:00), Dst: VMware_c0:00:08 (00:50:56:c0:00:08)

> Internet Protocol Version 4, Src: 192.168.126.130, Dst: 192.168.126.1

> Transmission Control Protocol, Src Port: 30000, Dst Port: 5572, Seq: 1, Ack: 1, Len: 26

▼ Data (26 bytes)

Data: 4d6f6e205365702032312031343a35393a32382032303230...

[Length: 26]

```

0000  00 50 56 c0 00 08 00 0c 29 9b 0d 00 08 00 45 00  .PV.... )....E.
0010  00 42 bc 66 40 00 40 06 00 7b c0 a8 7e 82 c0 a8  .B.f@.@. -{...
0020  7e 01 75 30 15 c4 83 56 92 6f 57 65 a4 42 50 18  ~.u0...V .oWe.BP.
0030  01 f6 77 e7 00 00 4d 6f 6e 20 53 65 70 20 32 31  .w...Mo n Sep 21
0040  20 31 34 3a 35 39 3a 32 38 20 32 30 32 30 0a 00  14:59:2 8 2020..
    
```

Data (data,data), 26 byte(s) | Packets: 8 · Displayed: 8 (100,0%) | Profile: Default

Linux time_client(192.168.126.128) <VMnet8> Widows time_server(192.168.126.1)

Capturing from VMware Network Adapter VMnet8 (tcp and port 30000)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.126.130	192.168.126.1	TCP	74	49294 → 30000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
2	0.000228	192.168.126.1	192.168.126.130	TCP	66	30000 → 49294 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS
3	0.000422	192.168.126.130	192.168.126.1	TCP	60	49294 → 30000 [ACK] Seq=1 Ack=1 Win=64256 Len=0
4	0.001402	192.168.126.1	192.168.126.130	TCP	80	30000 → 49294 [PSH, ACK] Seq=1 Ack=1 Win=1051136 Len=26
5	0.001432	192.168.126.1	192.168.126.130	TCP	54	30000 → 49294 [FIN, ACK] Seq=27 Ack=1 Win=1051136 Len=0
6	0.001655	192.168.126.130	192.168.126.1	TCP	60	49294 → 30000 [ACK] Seq=1 Ack=27 Win=64256 Len=0
7	0.001770	192.168.126.130	192.168.126.1	TCP	60	49294 → 30000 [FIN, ACK] Seq=1 Ack=28 Win=64256 Len=0
8	0.002084	192.168.126.1	192.168.126.130	TCP	54	30000 → 49294 [ACK] Seq=28 Ack=2 Win=1051136 Len=0

> Frame 4: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface \Device\NPF_{60640543-7BC8-4A60-A5A7-A657E328C1FF}, id 0

> Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_9b:0d:00 (00:0c:29:9b:0d:00)

> Internet Protocol Version 4, Src: 192.168.126.1, Dst: 192.168.126.130

> Transmission Control Protocol, Src Port: 30000, Dst Port: 49294, Seq: 1, Ack: 1, Len: 26

▼ Data (26 bytes)

Data: 4d6f6e205365702032312031353a30343a32302032303230...

[Length: 26]

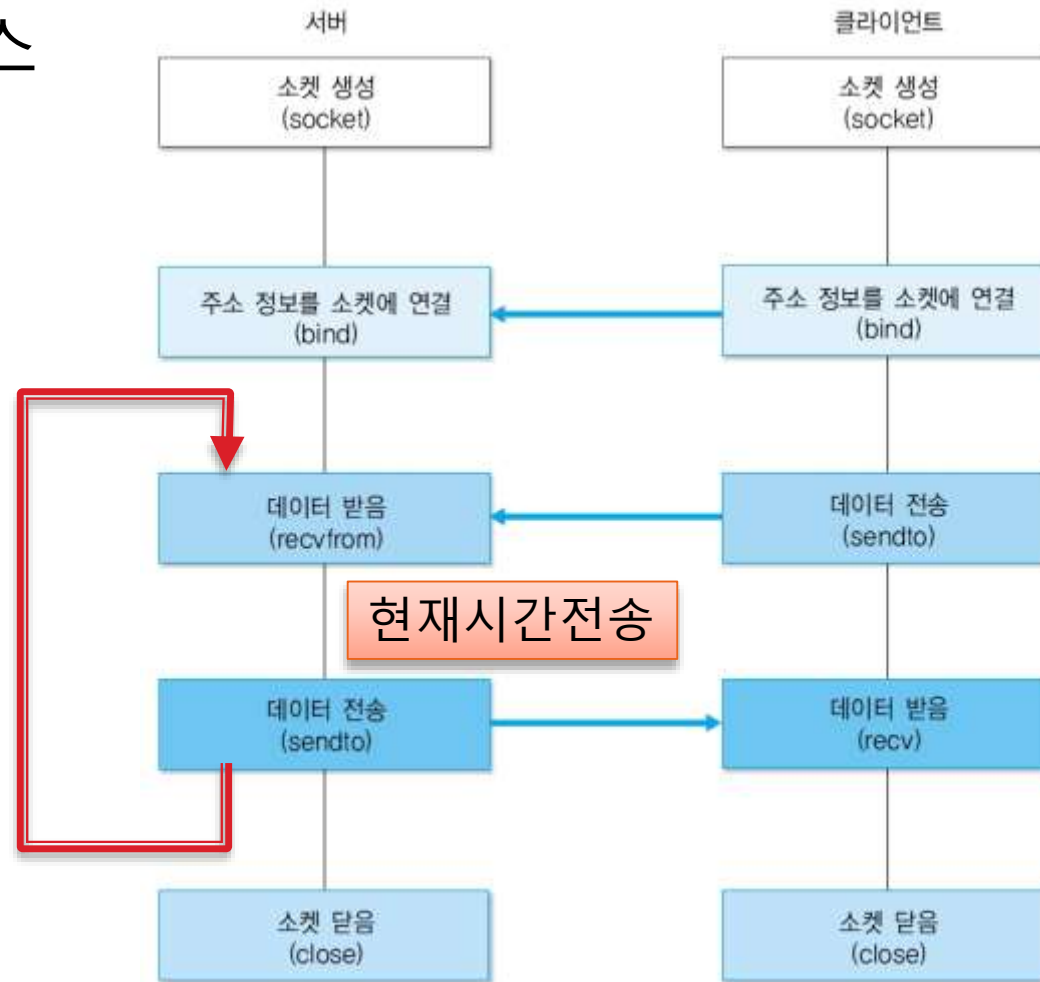
```

0000  00 0c 29 9b 0d 00 00 50 56 c0 00 08 08 00 45 00  ..)....P V....E.
0010  00 42 d0 03 40 00 80 06 ac dd c0 a8 7e 01 c0 a8  .B..@... ..~...
0020  7e 82 75 30 c0 8e 4a e3 6b 4e 33 af a9 fe 50 18  ~u0..J. kN3...P.
0030  10 0a 48 9c 00 00 4d 6f 6e 20 53 65 70 20 32 31  ..H...Mo n Sep 21
0040  20 31 35 3a 30 34 3a 32 30 20 32 30 32 30 0a 00  15:04:2 0 2020..
    
```

VMware Network Adapter VMnet8: <live capture in progress> | Packets: 8 · Displayed: 8 (100.0%) | Profile: Default

소켓프로그램 예1-2 – UDP time client/server

▶ 비연결형 서비스



[그림 12-2] UDP를 이용한 통신 절차

소켓프로그램 예1-2 – UDP time client/server

▶ udp_time_server.c

```
main (int argc, char *argv[])
{
    time_t today;
    char *port_no = TIME_PORT;
    if (argc==2) port_no = argv[1];
    sock = socket (AF_INET, SOCK_DGRAM, 0);
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl (INADDR_ANY);
    server.sin_port = htons (atoi(port_no));
    bind (sock, (struct sockaddr *)&server, sizeof (server));

    while (1) {
        client_len = sizeof(client);
        buf_len = recvfrom (sock, buf, 256, 0, (struct sockaddr *)&client, &client_len);
        if (buf_len < 0)
            exit (1);
        printf("Server : A client data recvfrom msg = %s.\n", buf);
        time (&today);
        strcpy (buf, ctime (&today));
        printf("Server time=%s", buf);
        sendto (sock, buf, strlen (buf) + 1, 0, (struct sockaddr *)&client, client_len);
    }
}
```

TCP SOCK_STREAM
UDP SOCK_DGRAM

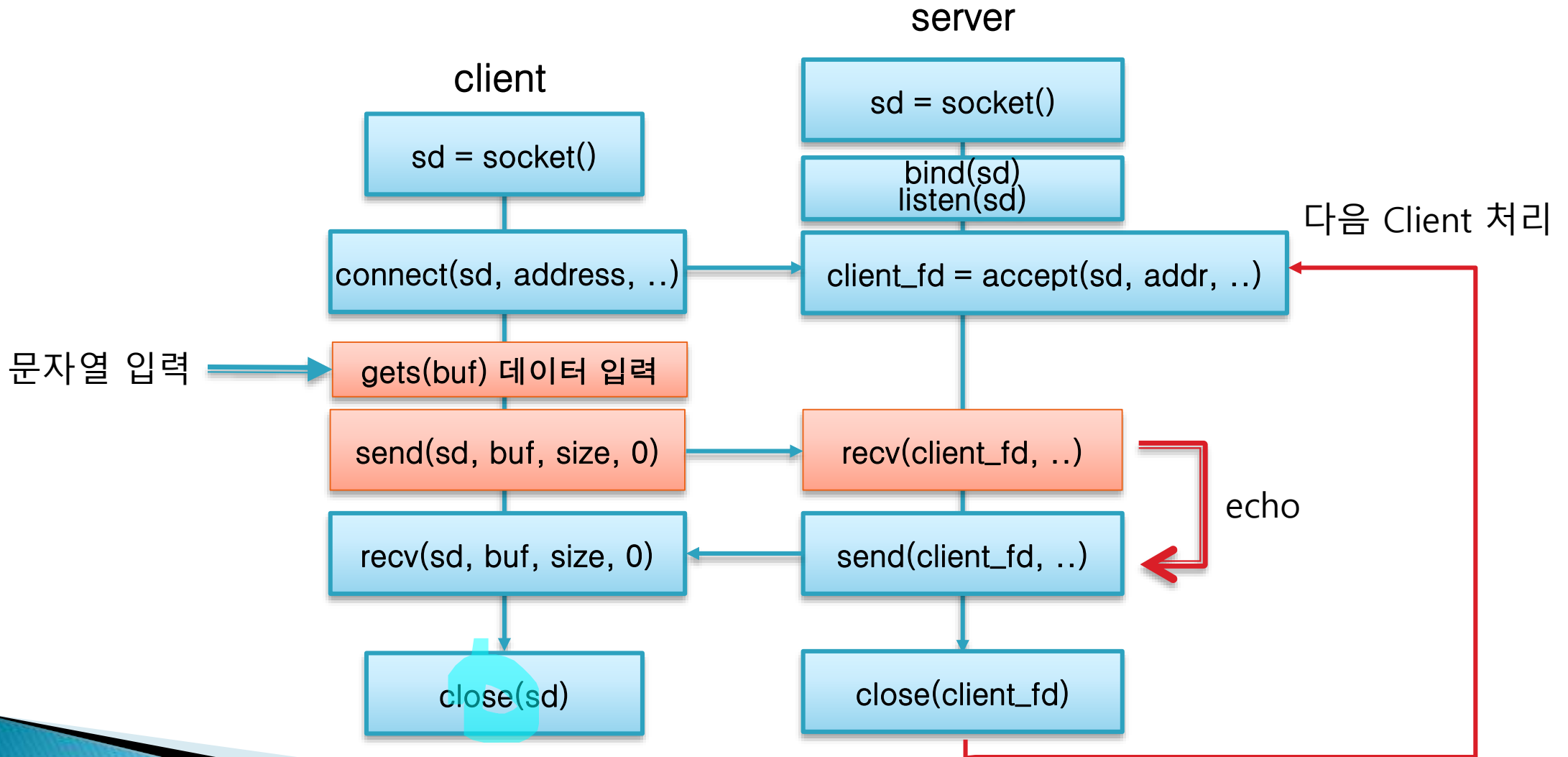
INADDR_ANY = 0.0.0.0
모든 연결에서 대기한다.

소켓프로그램 예1-2 – UDP time client/server

▶ udp_time_client.c

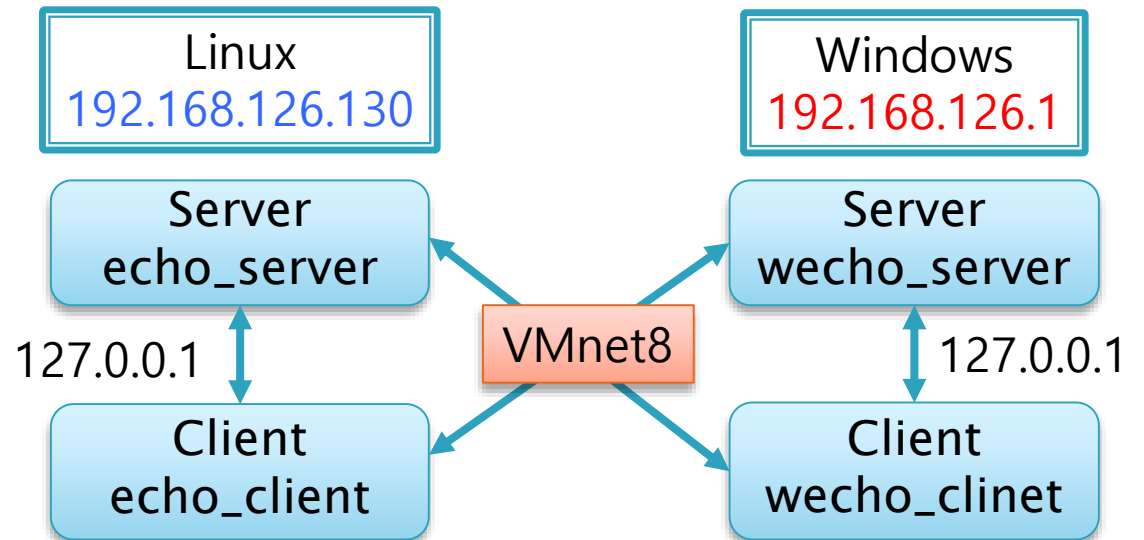
```
main ()
{
    sock = socket (AF_INET, SOCK_DGRAM, 0);
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl (inet_addr (TIME_SERVER));
    server.sin_port = htons (TIME_PORT);
    buf[0] = '?'; buf[1] = '\0';
    server_len = sizeof(server);
    buf_len = sendto (sock, buf, strlen(buf) + 1, 0, (struct sockaddr *)&server,
        server_len);
    if (buf_len < 0) exit (1);
    buf_len = recvfrom(sock, buf, 256, 0, (struct sockaddr *)&server,
        &server_len);
    if (buf_len < 0) exit (1);
    printf ("Time information from server is %s", buf);
}
```


소켓프로그램 예제 2 – Echo Client/Server



소켓프로그래밍 예제 2 - echo client server

- ▶ Linux(Vmware)와 Windows IP 확인
 - Linux : ip a 또는 ifconfig
 - 192.168.126.128 이라고 가정
 - Windows : ipconfig 로 확인
 - 192.168.126.1 라고 가정
- ▶ Linux <-> Linux
 - ./echo_server [30000]
 - ./echo_client [127.0.0.1] [30000]
- ▶ Windows <-> Windows
 - wecho_server [30000]
 - wecho_client [127.0.0.1] [30000]
- ▶ Linux <-> Window
 - Linux server / Windows client
 - Linux : echo_server [30000]
 - Window : wecho_client 192.168.126.130 30000
 - Windows server / Linux client
 - Window : wecho_server [30000]
 - Linux : ./echo_client 192.168.126.1 30000



echo client/server 실행 화면 (127.0.0.1 Loopback)

```
user@user-virtual-machine: ~/netprog/NetP04-linux
user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_server
echo_server waiting connection..
server_fd = 3
Server : waiting connection request.
Client connected from 127.0.0.1:45614
client_fd = 4
Received len=6 : hello
Sending len=6 : hello
Client connected from 127.0.0.1:45616
client_fd = 4
Received len=19 : Hansung University
Sending len=19 : Hansung University

user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_client
Connecting 127.0.0.1 30000
Input string : hello
Sending len=6 : hello
Received len=6 : hello
user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_client
Connecting 127.0.0.1 30000
Input string : Hansung University
Sending len=19 : Hansung University
Received len=19 : Hansung University
user@user-virtual-machine:~/netprog/NetP04-linux$
```

Linux echo_client/server

```
C:\Windows\system32\cmd.exe
echo_server waiting connection..
server_fd = 248
Server : waiting connection request.
Client connected from 127.0.0.1:4527
client_fd = 256
Received len=23 : Windows Client Message
Sending len=23 : Windows Client Message
Received len=23 : Windows Client Message
Client connected from 127.0.0.1:4529
client_fd = 256
Received len=19 : Hansung University
Sending len=19 : Hansung University

C:\Windows\system32\cmd.exe
Connecting 127.0.0.1 30000
Input string : Windows Client Message
Sending len=23 : Windows Client Message
Received len=23 : Windows Client Message
계속하려면 아무 키나 누르십시오 . . .

C:\Windows\system32\cmd.exe
Connecting 127.0.0.1 30000
Input string : Hansung University
Sending len=19 : Hansung University
Received len=19 : Hansung University
계속하려면 아무 키나 누르십시오 . . .
```

Windows wecho_client/server

echo client/server 실행 화면 (Linux <> Windows)

Linux Client

```
user@user-virtual-machine: ~/netprog/NetP04-linux
user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_client
Connecting 127.0.0.1 30000
Input string : hello
Sending len=6 : hello
Received len=6 : hello
user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_client 192.168.126.1 3000
Connecting 192.168.126.1 3000
^C
user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_client 192.168.126.1 30000
Connecting 192.168.126.1 30000
Input string : Linux client
Sending len=13 : Linux client
Received len=13 : Linux client
user@user-virtual-machine:~/netprog/NetP04-linux$
```

Windows Server

```
C:\Windows\system32\cmd.exe
echo_server waiting connection..
server_fd = 244
Server : waiting connection request.
Client connected from 192.168.126.130:49330
client_fd = 248
Received len=13 : Linux client
Sending len=13 : Linux client
```

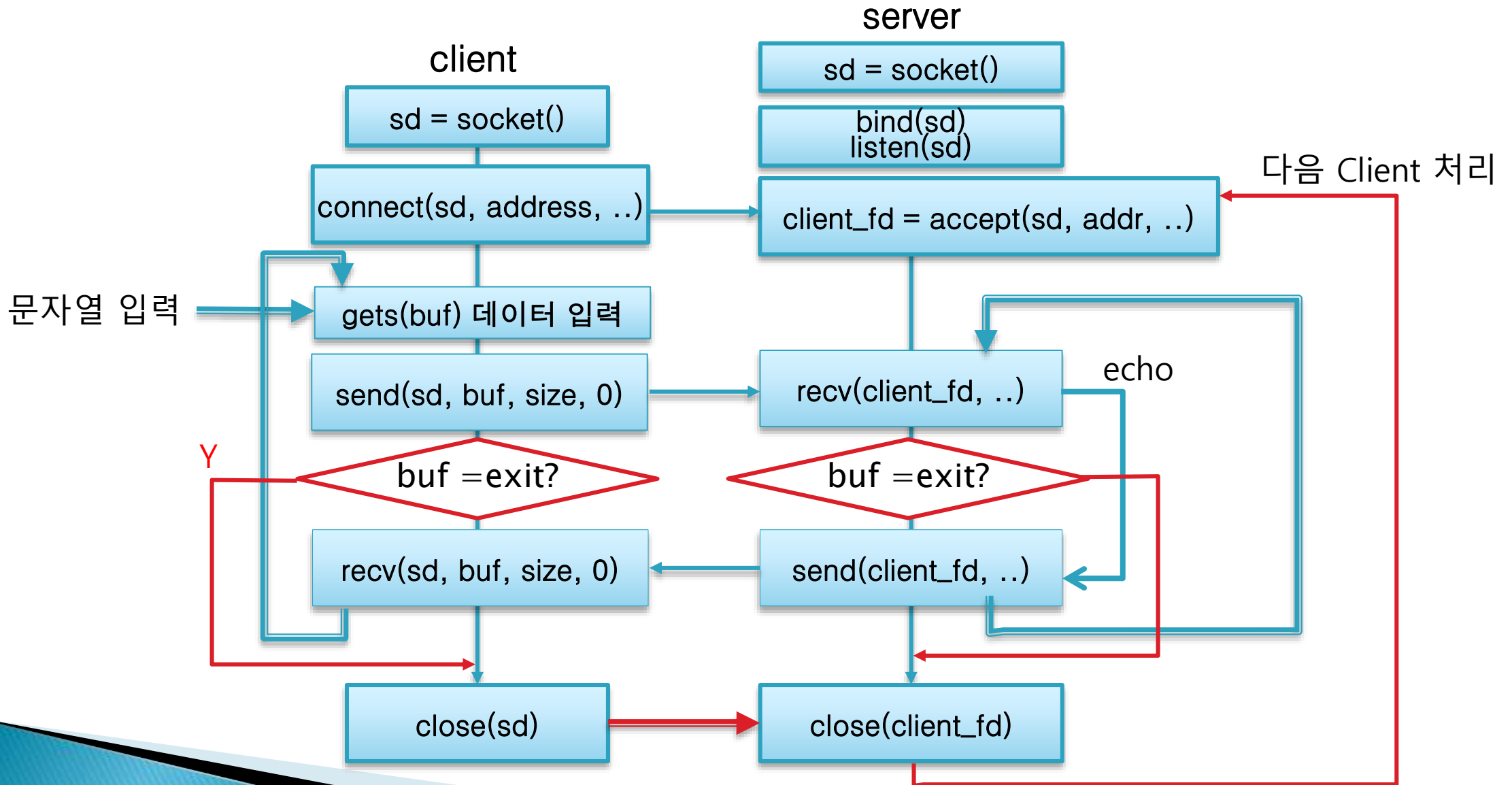
Linux Server

```
user@user-virtual-machine: ~/netprog/NetP04-linux
user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_server
echo_server waiting connection..
server_fd = 3
Server : waiting connection request.
Client connected from 127.0.0.1:55162
client_fd = 4
Received len=6 : hello
Sending len=6 : hello
Client connected from 192.168.126.1:1136
client_fd = 4
Received len=15 : Windwos client
Sending len=15 : Windwos client
```

Windows Client

```
C:\Windows\system32\cmd.exe
Connecting 192.168.126.130 30000
Input string : Windwos client
Sending len=15 : Windwos client
Received len=15 : Windwos client
계속하려면 아무 키나 누르십시오 . . .
```

응용1 echo client1/server1 – exit 입력 전까지 반복



응용1 echo client1/server1 "exit" 까지 반복 실행 화면

<pre>C:\Windows\system32\cmd.exe echo_server1 waiting connection.. server_fd = 264 Server : waiting connection request. Client connected from 127.0.0.1:1284 client_fd = 76 Received len=6 : hello Sending len=6 : hello Received len=3 : hi Sending len=3 : hi Received len=8 : Hansung Sending len=8 : Hansung Received len=5 : exit Client connected from 127.0.0.1:1296 client_fd = 76 Received len=7 : hello Sending len=7 : hello Received len=3 : hi Sending len=3 : hi Received len=5 : test Sending len=5 : test Received len=5 : exit</pre>	<pre>C:\Windows\system32\cmd.exe Connecting 127.0.0.1 30000 Input string : hello Sending len=7 : hello Received len=7 : hello Input string : hi Sending len=3 : hi Received len=3 : hi Input string : test Sending len=5 : test Received len=5 : test Input string : exit Sending len=5 : exit 계속하려면 아무 키나 누르십시오 . . .</pre>
---	--

Windows >> Linux 복제하기

▶ Windows Visual Studio

- 일단, Windows 환경에서 완벽하게 구현 (127.0.0.1 사용)
- wecho_client1.c, wecho_server1.c 에서 변경된 부분만 복사
 - connect/accept 이후 부분만 복사하면 됨

▶ Linux

- cp echo_client.c echo_client1.c
- cp echo_server.c echo_server1.c
- Windows 에서 변경된 부분만 복사
 - connect/accept 이후 부분만 복사하면 됨
 - closesocket() >> close() 로 변경
- Makefile 에 echo_client echo_server 추가
- make 로 compile
- Windwos<>Linux 상호 Test

응용2 echo client2/server2 – 대문자로 변경하기

- ▶ Client 에서 보낸 문자열을 Server 가 대문자로 변경하여 Return
- ▶ Client – echo_client.c 그대로 사용
 - Local에서 대문자 변환 대신 Server에 변환 요청

```
// Local 에서 변환  
gets(str);  
char *s = str;  
while (*s)  
    *s = toupper(*s)
```

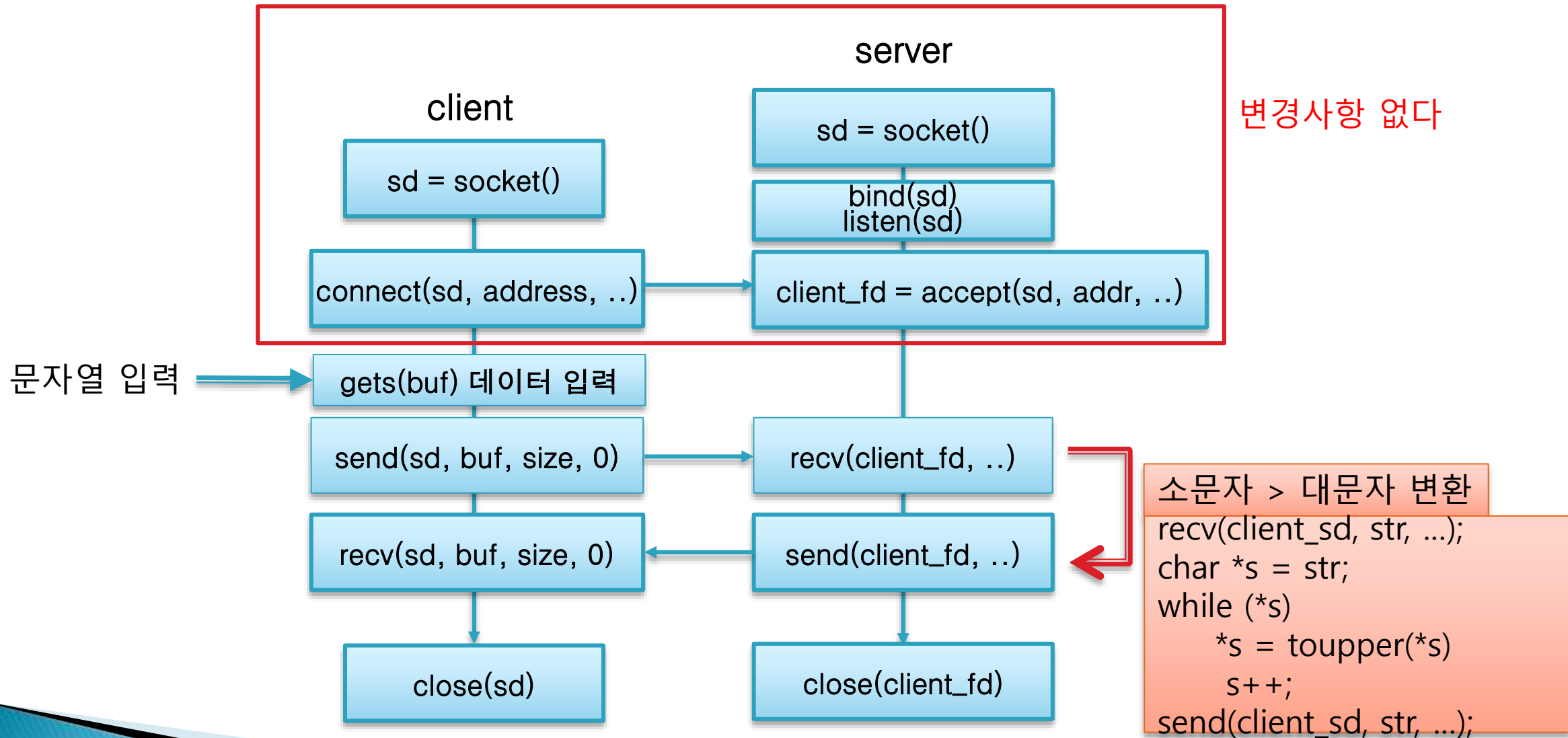


```
// Server에게 변환 요청  
// Echo client에서 수정할 것이 없다.  
gets(str);  
send(server_sd, str, ...);  
recv(server_sd, str, ...)
```

- ▶ Server – echo_server.c
- ▶ Client에서 받은 문자열을 대문자로 변환하고 송신

```
recv(client_sd, str, ...);  
char *s = str;  
while (*s)  
    *s = toupper(*s)  
send(client_sd, str, ...);
```


응용2 echo client2/server2 – 대문자로 변환하기



echo client2/server2 모두 대문자로 변환

<pre>C:\Windows\system32\cmd.exe echo_server2 waiting connection.. server_fd = 248 Server : waiting connection request. Client connected from 127.0.0.1:1755 client_fd = 256 Received len=6 : hello Sending len=6 : HELLO Received len=9 : Computer Sending len=9 : COMPUTER Received len=19 : Hansung University Sending len=19 : HANSUNG UNIVERSITY Received len=5 : exit Client connected from 127.0.0.1:1760 client_fd = 256 Received len=6 : Hello Sending len=6 : HELLO Received len=8 : Hansung Sending len=8 : HANSUNG Received len=11 : University Sending len=11 : UNIVERSITY Received len=9 : Computer Sending len=9 : COMPUTER Received len=12 : Engineering Sending len=12 : ENGINEERING Received len=5 : Exit Sending len=5 : EXIT Received len=5 : exit</pre>	<pre>C:\Windows\system32\cmd.exe Connecting 127.0.0.1 30000 Input string : Hello Sending len=6 : Hello Received len=6 : HELLO Input string : Hansung Sending len=8 : Hansung Received len=8 : HANSUNG Input string : University Sending len=11 : University Received len=11 : UNIVERSITY Input string : Computer Sending len=9 : Computer Received len=9 : COMPUTER Input string : Engineering Sending len=12 : Engineering Received len=12 : ENGINEERING Input string : Exit Sending len=5 : Exit Received len=5 : EXIT Input string : exit Sending len=5 : exit 계속하려면 아무 키나 누르십시오 . . .</pre>
--	---

응용3 echo client3/server3 – 대/소문자 서로 변경하기

- ▶ Client 에서 보낸 문자열을 Server 가

- 대문자는 > 소문자로
- 소문자는 > 대문자로

- ▶ Client

- echo_client.c 그대로 사용

- ▶ Server

- 대/소문자 상호 변환 구현

```
mgs_size = recv(client_sd, buf, ...);
buf[msg_size] = '\0';
char *s = buf;
while (*s) {
    if (islower(*s))
        *s = toupper(*s);
    else
        *s = tolower(*s);
    s++;
}
send(client_sd, buf, ...);
```

echo client3/server3 대>소 소>대 변환

C:\Windows\system32\cmd.exe

```
echo_server2 waiting connection..  
server_fd = 272  
Server : waiting connection request.  
Client connected from 127.0.0.1:1772  
client_fd = 216  
Received len=6 : Hello  
Sending len=6 : hELLO  
Received len=8 : Hansung  
Sending len=8 : hANSUNG  
Received len=32 : University Computer Engineering  
Sending len=32 : uNIVERSITY cOMPUTER eNGINEERING  
Received len=5 : exit  
Client connected from 127.0.0.1:1773  
client_fd = 276  
Received len=3 : Hi  
Sending len=3 : hI  
Received len=20 : Network Programming  
Sending len=20 : nETWORK pROGRAMMING  
Received len=5 : exit
```

C:\Windows\system32\cmd.exe

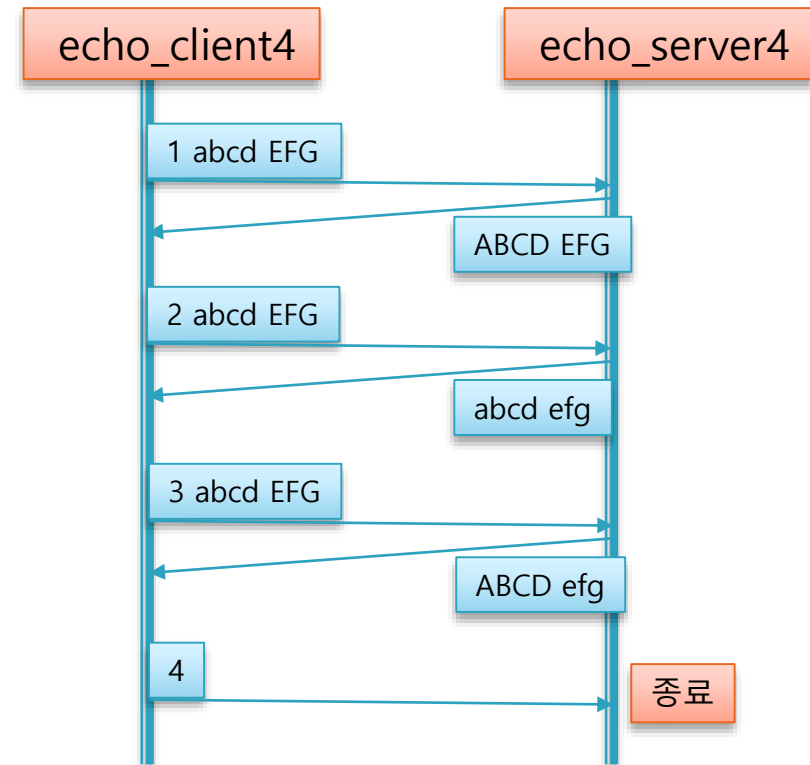
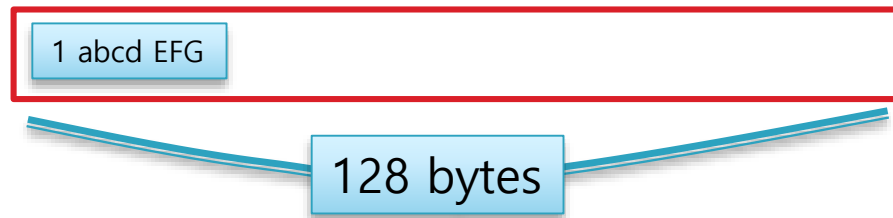
```
Connecting 127.0.0.1 30000  
Input string : Hi  
Sending len=3 : Hi  
Received len=3 : hI  
Input string : Network Programming  
Sending len=20 : Network Programming  
Received len=20 : nETWORK pROGRAMMING  
Input string : exit  
Sending len=5 : exit  
계속하려면 아무 키나 누르십시오 . . .
```

4주 과제1 Echo Client / Server 동영상 강의 내용까지 복습

1. `echo client/server` 기본
2. `echo client1/server1` `exit` 입력때까지 반복하기
3. `echo client2/server2` 대문자로 변경하기
4. `echo client3/server3` 대>소 소>대문자로 변경하기

4주 과제2 Echo Client4 / Server4 대/소문자 변경 메뉴방식 구현

- ▶ Linux Server / Windows Client 로 구현
- ▶ 송/수신 data를 128 byte로 고정시켜서 전송하기
- ▶ Client 메뉴와 Application Protocol 만들기
 - 1 소문자->대문자, 2 대문자->소문자, 3 대(소)문자 → 소(대)문자
 - 송신: 1 abcd EFG → 수신: ABCD EFG
 - 송신: 2 ABCD efg → 수신: abcd efg
 - 송신: 3 abcd EFG → 수신: ABCD efg
 - 송신: 4 → 수신 session 종료



```
C:\Windows\system32\cmd.exe
Connecting 127.0.0.1 30000
*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 1
Input string : Hansung University
Received 128 bytes : HANSUNG UNIVERSITY

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 2
Input string : COMPUTER Engineering
Received 128 bytes : computer engineering

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 3
Input string : Network Programming
Received 128 bytes : nETWORK pROGRAMMING

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 4
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
Connecting 127.0.0.1 30000
*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 1
Input string : Computer Engineering
Received 128 bytes : COMPUTER ENGINEERING

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 2
Input string : Network Programming
Received 128 bytes : network programming

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 3
Input string : Socket Programming
Received 128 bytes : sOCKET pROGRAMMING

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 4
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
echo_server waiting connection..
server_fd = 252
Server : waiting connection request.
Client connected from 127.0.0.1:4862
client_fd = 248
Received len=128 : 1 Hansung University
Sending len=128 : HANSUNG UNIVERSITY
Received len=128 : 2 COMPUTER Engineering
Sending len=128 : computer engineering
Received len=128 : 3 Network Programming
Sending len=128 : nETWORK pROGRAMMING
Received len=128 : 4
Client connected from 127.0.0.1:5229
client_fd = 256
Received len=128 : 1 Computer Engineering
Sending len=128 : COMPUTER ENGINEERING
Received len=128 : 2 Network Programming
Sending len=128 : network programming
Received len=128 : 3 Socket Programming
Sending len=128 : sOCKET pROGRAMMING
Received len=128 : 4
```


Windows Client > Linux Server

```
C:\Windows\system32\cmd.exe
Connecting 192.168.126.130 30000
*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 1
Input string : Windows message
Received 128 bytes : WINDOWS MESSAGE

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 2
Input string : HANSUNG UNIVERSITY
Received 128 bytes : hansung university

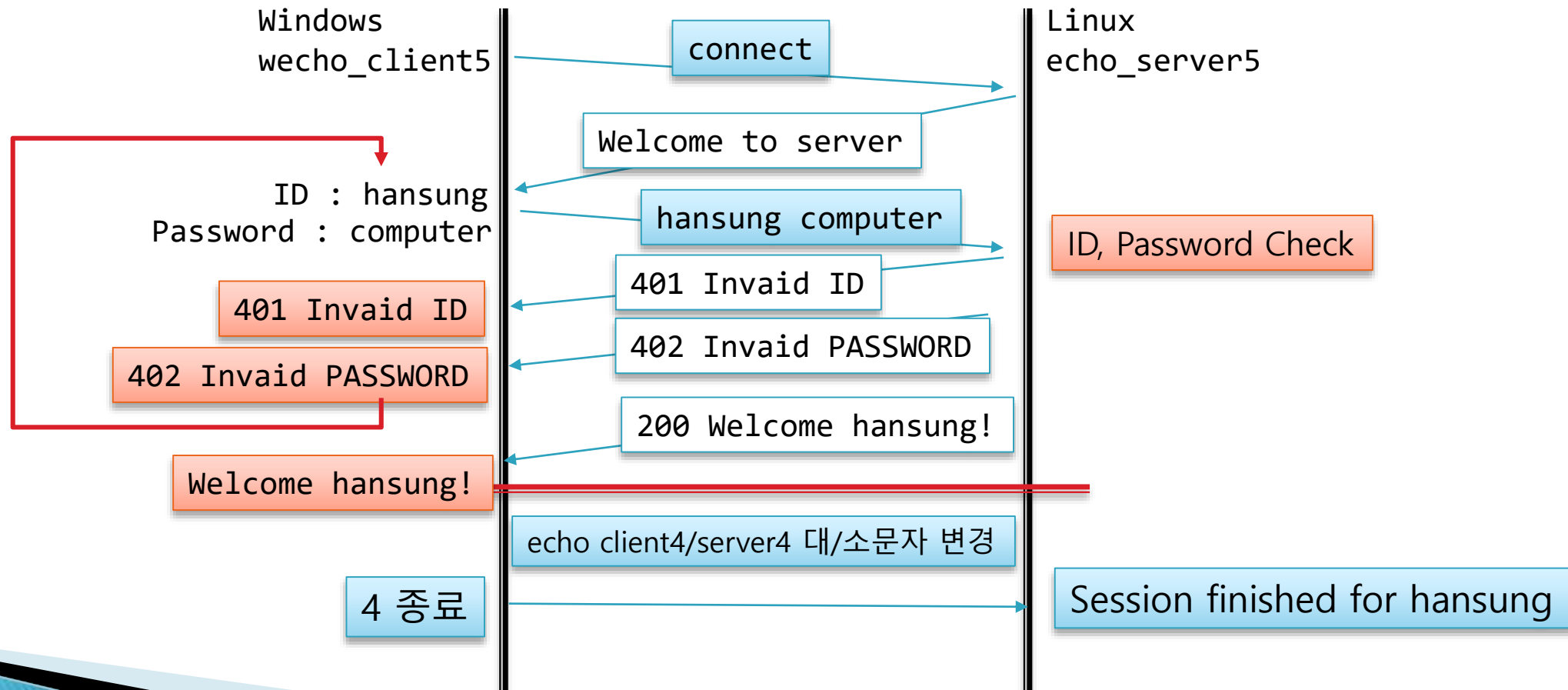
*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 3
Input string : Windows client > Linux Server
Received 128 bytes : wINDOWS CLIENT > LINUX SERVER

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 4
계속하려면 아무 키나 누르십시오 . . .
```

```
user@user-virtual-machine: ~/netprog/NetP04-linux
echo_client      time_client      udp_time_client
echo_client.c    time_client.c    udp_time_client.c
user@user-virtual-machine:~/netprog/NetP04-linux$ cp echo_client.c echo_client4.c
user@user-virtual-machine:~/netprog/NetP04-linux$ cp echo_server.c echo_server4.c
user@user-virtual-machine:~/netprog/NetP04-linux$ vi Makefile
user@user-virtual-machine:~/netprog/NetP04-linux$ make
cc -w echo_client4.c -o echo_client4
cc -w echo_server4.c -o echo_server4
user@user-virtual-machine:~/netprog/NetP04-linux$ ./echo_server4
echo_server waiting connection..
server_fd = 3
Server : waiting connection request.
Client connected from 192.168.126.1:4589
client_fd = 4
Received len=128 : 1 Windows message
Sending len=128 : WINDOWS MESSAGE
Received len=128 : 2 HANSUNG UNIVERSITY
Sending len=128 : hansung university
Received len=128 : 3 Windows client > Linux Server
Sending len=128 : wINDOWS CLIENT > LINUX SERVER
Received len=128 : 4
```


4주 과제3 Login 기능 추가하기

- ▶ echo_client4/server4 활용 >> echo_client5/server5
- ▶ Windows Client / Linux Server 로 구현
- ▶ Login protocol 만들기



echo_client5/server5 실행 화면

```
ca. C:\Windows\system32\cmd.exe
Connecting 127.0.0.1 30000
Received 128 bytes : Welcome to Server!!
ID : hansung1
Password : computer
401 Invalid ID
ID : hansung
Password : computer1
402 Invalid Password
ID : hansung
Password : computer
Welcome hansung!!
*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 1
Input string : Hello
Received 128 bytes : HELLO

*** 대/소문자 변환 메뉴입니다. ***
(1) 모두 대문자 변환
(2) 모두 소문자 변환
(3) 대>소 소>대 변환
(4) 종료
선택하세요 : 4
계속하려면 아무 키나 누르십시오 . . .
```

```
ca. C:\Windows\system32\cmd.exe
echo_server5 waiting connection..
server_fd = 248
Server : waiting connection request.
Client connected from 127.0.0.1:1871
client_fd = 256
Sending len=128 : Welcome to Server!!
Received id=hansung1 pass=computer
Sending len=128 : 401 Invalid ID
Received id=hansung pass=computer1
Sending len=128 : 402 Invalid Password
Received id=hansung pass=computer
Sending len=128 : 200 Welcome hansung!!
Received len=128 : 1 Hello
Sending len=128 : HELLO
Received len=128 : 4
Session finished for hansung.
```