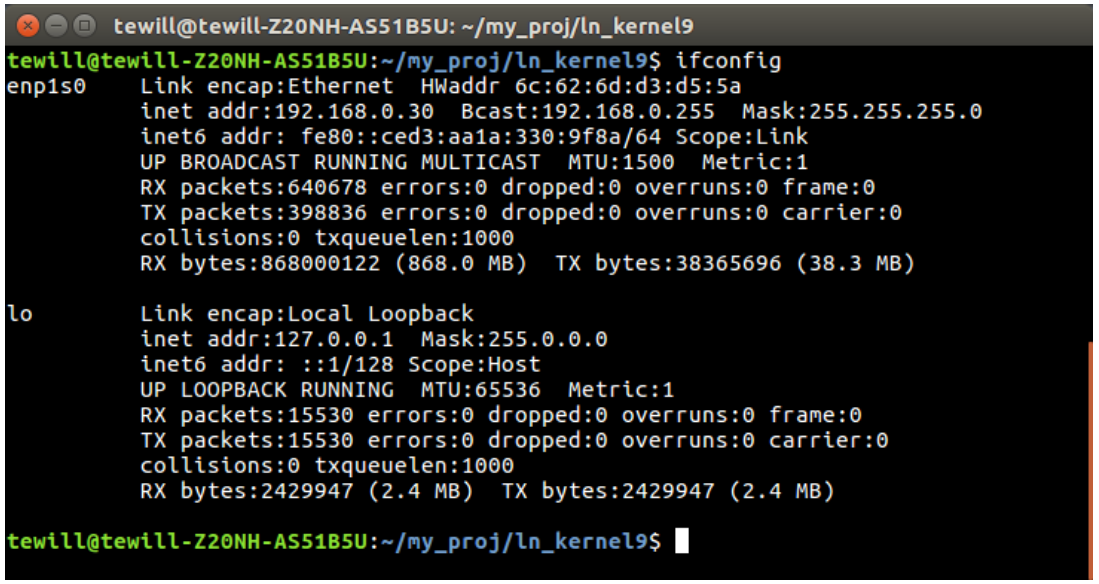
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 은태영
zero_bird@naver.com

명령어_기반 지식



- ifconfig : 자신의 IP 를 확인 할 수 있다.
- inet addr(ipv4) : 기존 IP 로서, 1 byte 4개로 구성되어 있다.
- 1 byte 이기 때문에 255 까지 존재한다.
- 예약어들이 존재한다.
- (0 : 게이트 웨이, 255 : 브로드 캐스트)
- 해당 문제를 해결하기 위해 ipv6 가 만들어졌다.
- ipv6 : 센서 네트워크를 구축하기 위해 만들어짐.
- HWaddr : MAC(하드웨어 장치) 고유 번호.
- PC의 경우 LAN 카드의 식별자이다.

Network 프로그래밍

- CS(client, server)
- 토폴로지 : 네트워크 구성도.(그래프 알고리즘)
- TCP / IP 프로토콜 : OSI 7Layer 를 기반으로 버클리 대학교에서 4계층으로 만들었다.
- 7Layer 보다 적은 계층을 사용하기에 더 가볍다(속도가 빠르다)(라우터, 스위치, OS 등 사용된다)
- 리눅스를 기반으로 만들어졌기 때문에, 리눅스 유닉스에 최적화 되어있다.(윈도우가 느린 이유)

- IP 의 종류
- 공인 IP(WAN 통신) 와 사설 IP(공유기) 가 존재한다.
- 사설 IP 는 NAT 에서 처리한다.

- Networking : 원격 IPC. 느리지만 외부와 통신을 할 수 있다.

sigaction

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```c
#include <stdio.h>
#include <signal.h>

struct sigaction act_new;
struct sigaction act_old;

void sigint_handler(int signo)
{
    printf("Ctrl + C\n");
    printf("IF you push it one more time then exit\n");
    sigaction(SIGINT, &act_old, NULL);
}

int main(void)
{
    act_new.sa_handler = sigint_handler;
    sigemptyset(&act_new.sa_mask);
    sigaction(SIGINT, &act_new, &act_old);
    while(1)
    {
        printf("sigaction test\n");
        sleep(1);
    }
    return 0;
}
```

1,1 All

- sigaction 은 signal 의 행동을 갖고 있는 구조체이다.
- .sa_handler : signal 에서 실행될 함수를 저장하는 위치이다.
- sigemptyset() : 시그널 셋을 비우는 함수이다.
- 사용 방법에 따라 SIG_IGN 과 같은 역할을 한다.
- &.sa_mask : 시그널 전부를 나타내는 주소이다.
- sigaction(, ,) : signal 과 동일하다.
- 1 번 인자는 받을 signal을 나타낸다.
- 2 번째 인자는 실행할 행동을 나타낸다.
- 3 번째 인자는 signal 의 리턴 값을 나타낸다.

Signal

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```c
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>

void gogogo(int voidv)
{
    printf("SIGINT Accur!\n");
    exit(0);
}

int main(void)
{
    signal(SIGINT, gogogo);

    for(;;)
    {
        printf("kill Test\n");
        sleep(2);
    }

    return 0;
}
```

1,1 All

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

int main(int argc, char *argv[])
{
    if(argv < 2)
        printf("Usage : ./exe pid\n");
    else
        kill(atoi(argv[1]), SIGINT);

    return 0;
}
```

1,1 All

- SIGINT 를 받을 때 gogogo 함수가 실행된다.
- 반복문을 통해 외부에서 종료해 보도록 한다.

- kill 을 통해 1번째 인자로 받은 PID 로 SIGINT 를 보낸다.
- PID 를 인자로 받는 해당 프로세스를 실행하면, 해당 PID 에 SIGINT 를 실행한다.
- 문자열 출력 후, 프로세스를 종료한다.

pthread_t

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```c
#include <stdio.h>
#include <pthread.h>

void *task1(void *X)
{
    printf("Thread A Complete\n");
}

void *task2(void *X)
{
    printf("Thread B Complete\n");
}

int main(void)
{
    pthread_t ThreadA, ThreadB;

    pthread_create(&ThreadA, NULL, task1, NULL);
    pthread_create(&ThreadB, NULL, task2, NULL);

    pthread_join(ThreadA, NULL);
    pthread_join(ThreadB, NULL);

    return 0;
}
```

1,1 All

- task 함수 : void * 를 통하여, 무엇이든 인자로 받고, 무엇이든 리턴한다.
- pthread_t : Thread 를 생성한다.
- pthread_create : Thread 의 형태를 만든다.
- 1번째 인자로 Thread 의 주소를 가리킨다.
- Thread 생성 시, 실행될 함수를 나타낸다.
- join : Thread 가 실행된다.

server

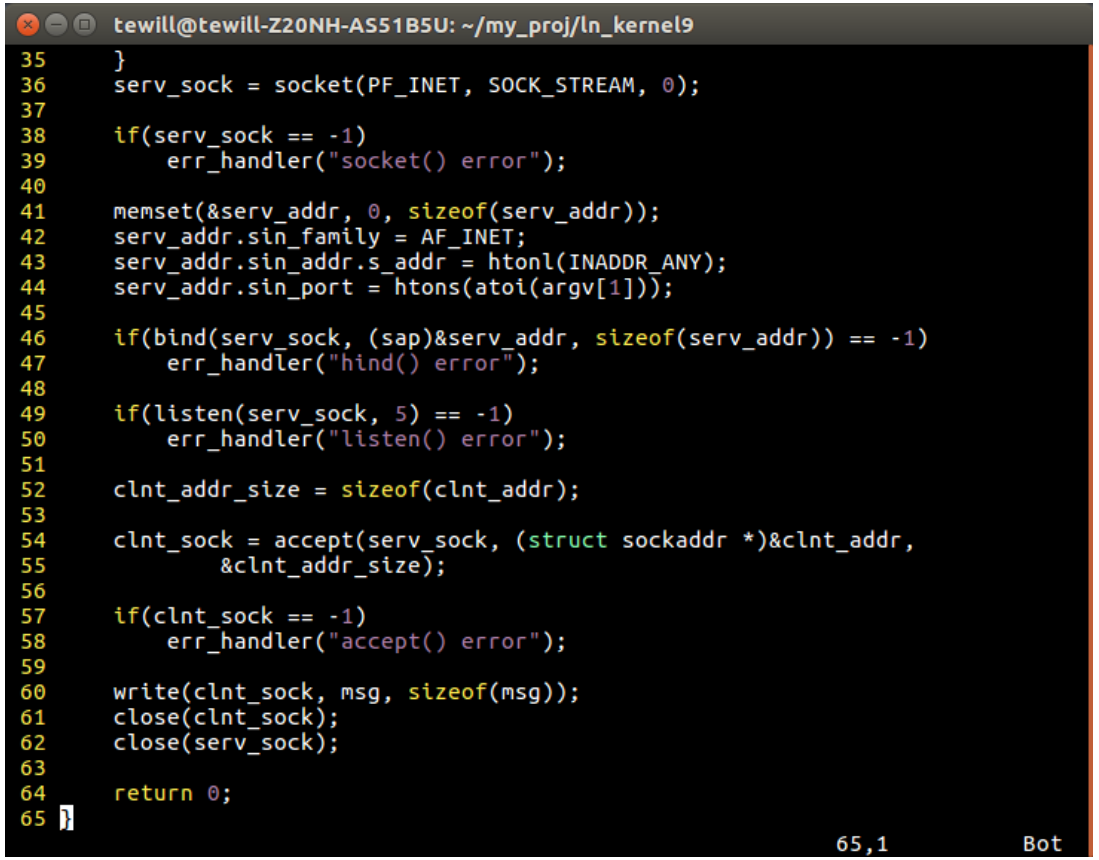
tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

// basic_server.c

typedef struct sockaddr_in  si;
typedef struct sockaddr *   sap;

void err_handler(char *msg)
{
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
}

int main(int argc, char **argv)
{
    int serv_sock;
    int clnt_sock;

    si serv_addr;
    si clnt_addr;
    socklen_t clnt_addr_size;

    char msg[] = "Hello Network Programming";

    if(argc != 2)
    {
        printf("use : %s <port>\n", argv[0]);
        exit(1);
    }
```

1,1 Top

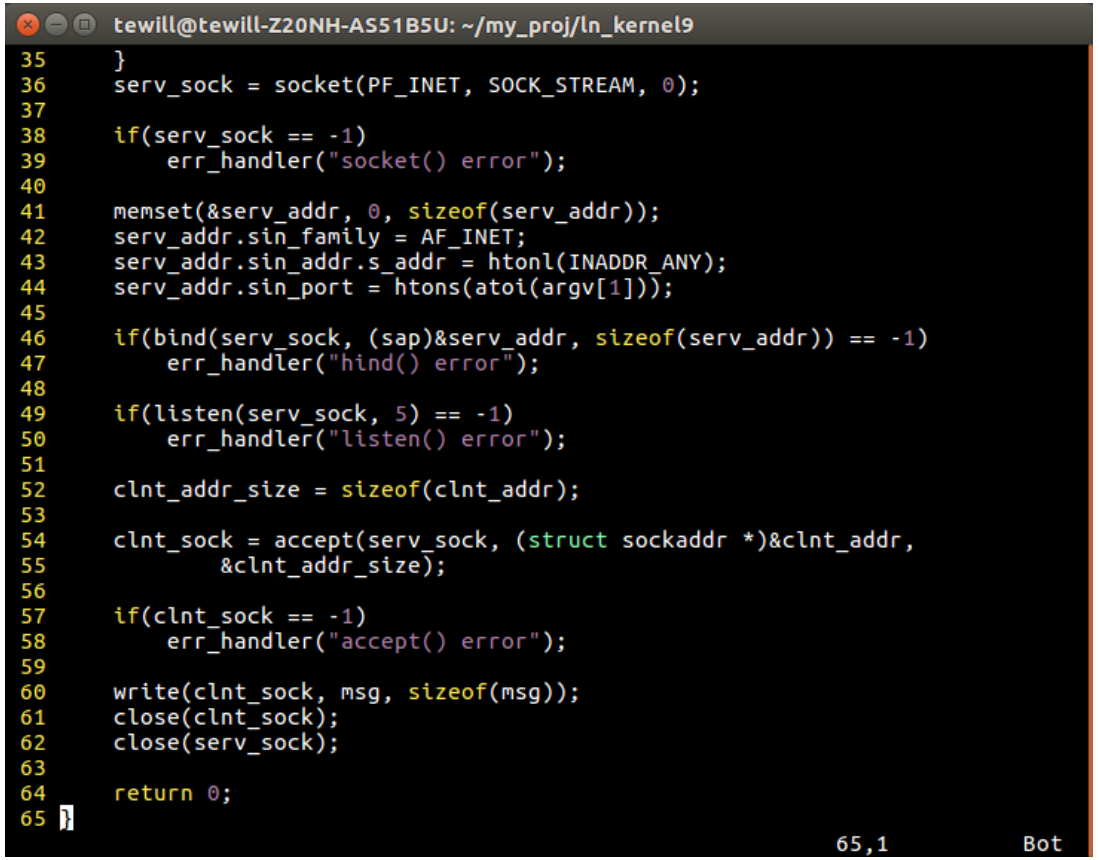
- ❖ sockaddr : 다양한 소켓 주소를 넣을 수 있는 구조체 이다.
- ❖ sockaddr_in : IPv4 주소체계에서 사용되는 구조체이다.
- ❖ err_handler : error 발생 시 확인하기 위해 존재.
- ❖ 서버와 클라이언트 소켓과 주소를 생성한다.
- ❖ socklen_t : 길이 및 크기에 대한 값이다.
- ❖ 확인용 문자열 msg 를 만든다.
- ❖ 인자로 port 를 받아오는지 체크한다.
- ❖ port : 통로 이다. 80(www), 22(ssh)
- ❖ 20(업로드) / 21(다운로드) : ftp 등
- ❖ 다른 숫자는 전용 커스텀 port 이다.(서비스 번호)

server



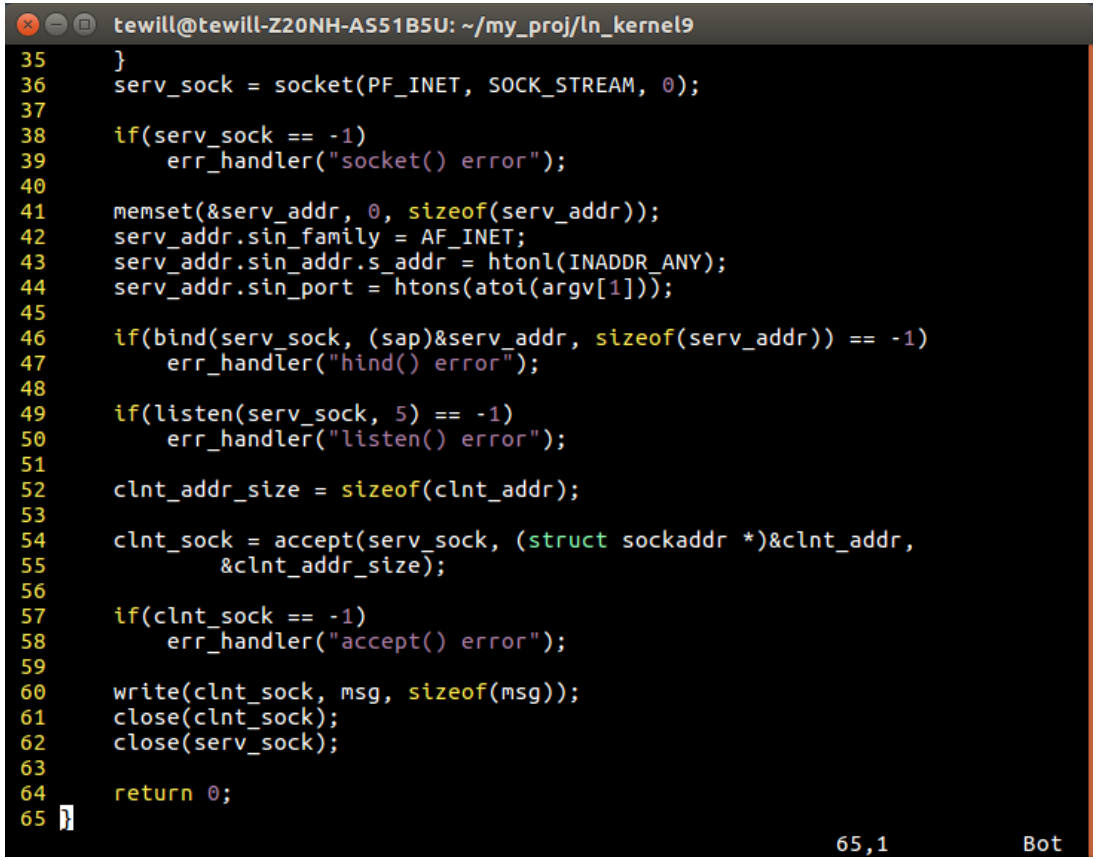
- socket : open 과 동일하게 File Descriptor 을 리턴 한다.
- 첫번째 인자는 프로토콜을 받는다.
- PF_INET : IPv4 프로토콜을 나타낸다.
- 두번째 인자는 전송방식을 설정한다.
- SOCK_STREAM : 느리지만 신뢰성이 높다.
- 세번째 인자는 0을 입력한다. (일종의 패턴)
- 에러를 확인한다.

server



- memset : 일종의 초기화이다.
- 1번 인자에 3번 인자만큼 2번 인자를 입력한다.
- family : 어떤 서비스 인지 셋팅한다.
- AF_INET : IPv4 의 주소 체계를 나타낸다.
- addr.s_addr : IP 를 받는다.
- INADDR_ANY : 사용 가능한 랜 카드의 IP 주소.
- 값은 #define 을 통하여 0으로 설정되어 있다.
- htons : short형 바이트 순서를 네트워크 바이트 순서로 변경한다.
- 자신이 입력한 port 를 입력한다.
- 일종의 초기화 패턴이다.

server

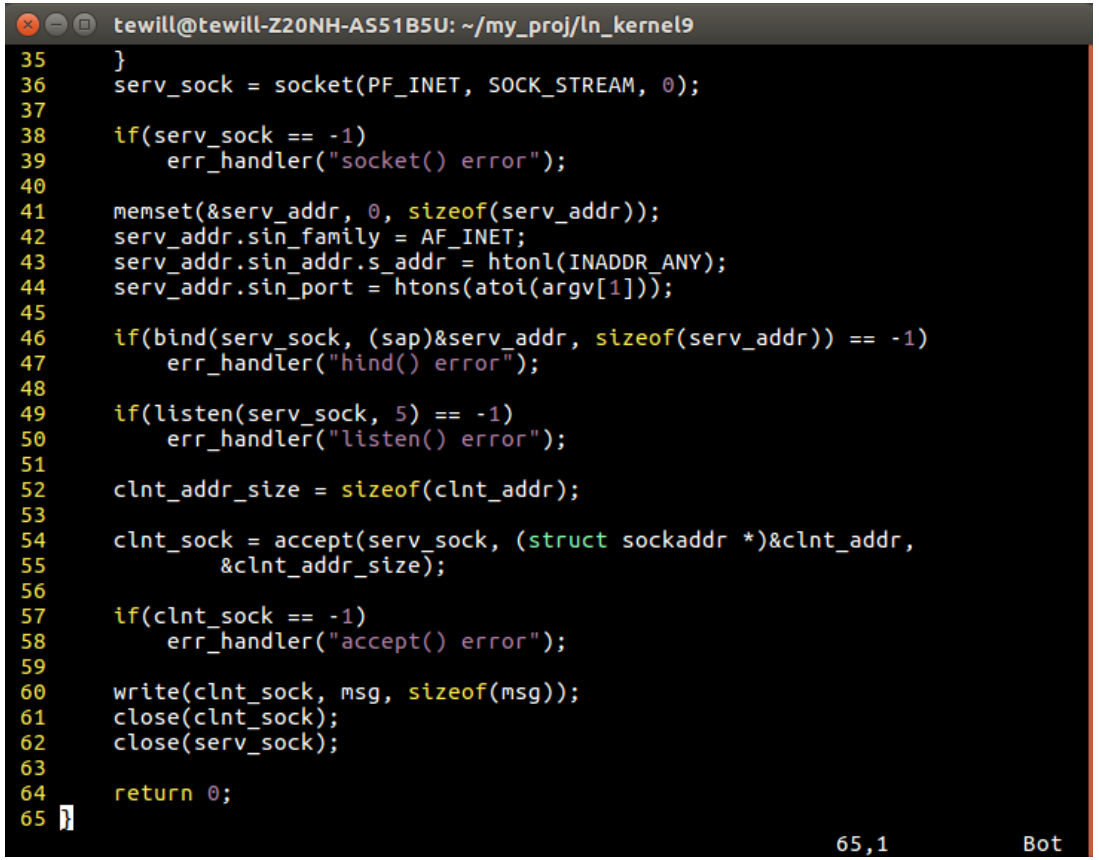


- ❖ bind : 서버의 IP 주소를 셋팅한다.
- ❖ 첫번째 인자로 주소를 입력할 파일 디스크립터다.
- ❖ sockaddr 를 통하여 정보를 받아온다.
- ❖ 구조체의 크기를 나타낸다.

- ❖ listen : 소켓을 연결 요청 대기상태로 둔다.
- ❖ 연결하고자 하는 파일 디스크립터를 주고, 연결 요청하는 대기 큐의 크기를 전달한다.
- ❖ 일종의 문지기 역할을 한다.
- ❖ 실제 접속을 기다리는 위치이다.

- ❖ size 를 클라이언트 사이즈로 저장한다.

server



- ❖ accept : 클라이언트의 접속을 허용한다.
- ❖ listen의 대기 큐에 위치한 연결요청을 수락한다.
- ❖ 1번 인자는 전달받을 파일 디스크립터 이다.
- ❖ 2번 인자는 연결요청 한 클라이언트의 정보이다.
- ❖ 3번 인자는 전달받을 클라이언트 정보 크기이다.
- ❖ listen 과 짝을 이룬다.
- ❖ 보낼 msg 를 연결된 파일 디스크립터에 쓴다.

client

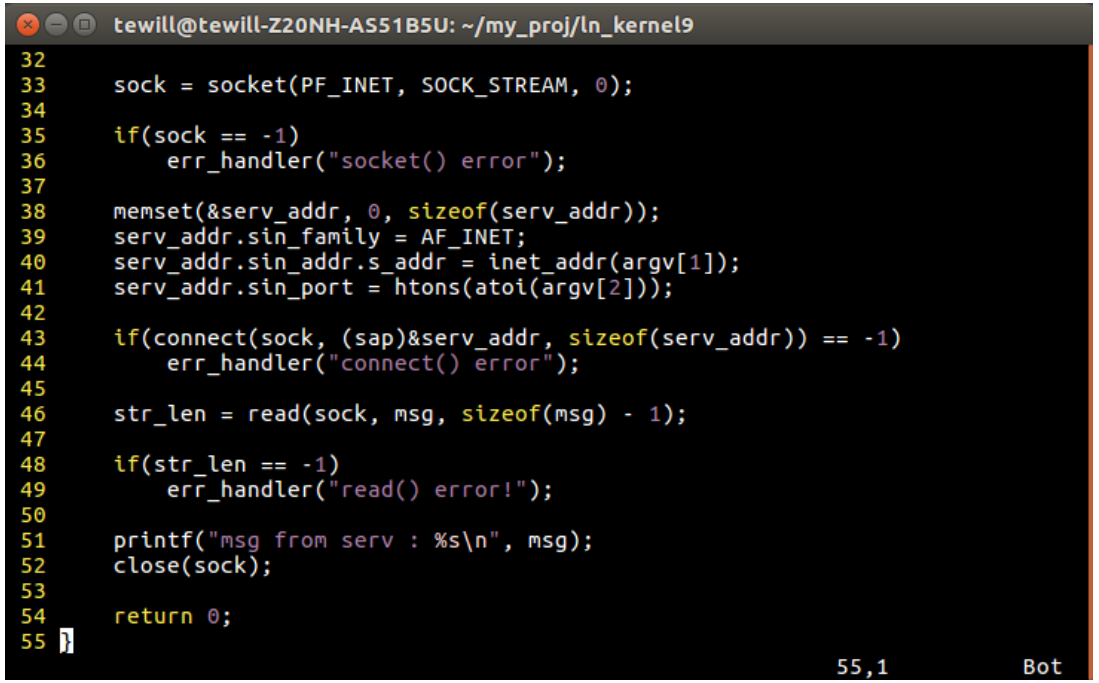
tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

//basic_client.c

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

void err_handler(char *msg)
{
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
}

int main(int argc, char **argv)
{
    int sock;
    int str_len;
    si serv_addr;
    char msg[32];

    if(argc != 3)
    {
        printf("use : %s <IP> <port>\n", argv[0]);
        exit(1);
    }
```

1,1 Top

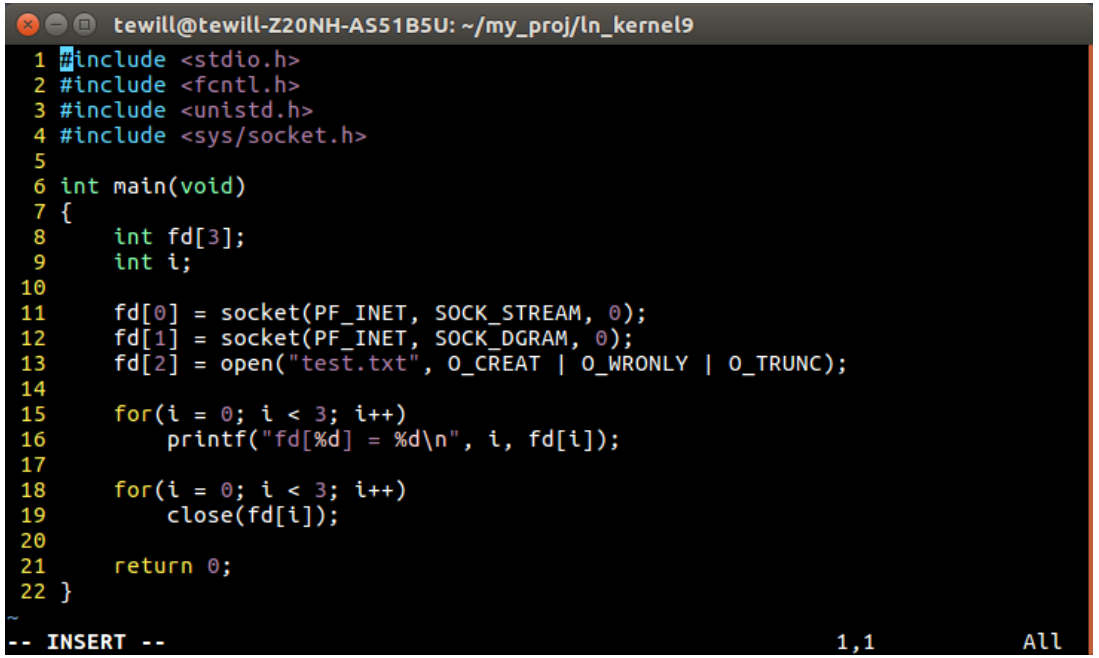
- ❖ 서버와 동일한 구조로 이루어져 있다.
- ❖ 인자를 2개 받아온다.
- ❖ 각각 IP 와 port 를 받는다.
- ❖ 인자 숫자를 체크한 후, 오류 체크를 한다.

client



- ❖ 동일한 방식으로 socket 을 만들고, 초기화 한다.
- ❖ connect : 연결 대기 중인 서버를 연결하는 함수이다.
- ❖ 첫번째 인자는 연결할 파일 디스트립터를 받는다.
- ❖ 두번째 인자는 셋팅한 주소 정보이다.
- ❖ 세번째 인자는 해당 정보의 크기이다.
- ❖ 읽기를 통해 받은 문자열을 msg 에 저장한다.
- ❖ 그 후, 출력한다.

File descriptor



- ❖ socket 의 return 값이 file descriptor 인 것을 확인하는 프로세스이다.
- ❖ TCP 와 UDP, 그리고 일반 파일 open 을 한다.
- ❖ 결과를 확인해 보면 각각 3, 4, 5 로 file descriptor 가 들어가는 것을 확인할 수 있다.

- ❖ socket 은 file 구조체 내부에 파일로서 존재한다.

client

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include <unistd.h>
 5 #include <arpa/inet.h>
 6 #include <sys/socket.h>
 7
 8 typedef struct sockaddr_in si;
 9 typedef struct sockaddr * sap;
10
11 void err_handler(char *msg)
12 {
13     fputs(msg, stderr);
14     fputc('\n', stderr);
15     exit(1);
16 }
17
18 int main(int argc, char **argv)
19 {
20     int sock;
21     int str_len = 0;
22     si serv_addr;
23     char msg[32] = {0};
24     int idx = 0, read_len = 0;
25
26     if(argc != 3)
27     {
28         printf("use : %s <IP> <port>\n", argv[0]);
29         exit(1);
30     }
                                                    1,1           Top
```

- 위에서 만들었던 클라이언트와 동일한 방식으로 만들어 진다.

client

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```
31
32      sock = socket(PF_INET, SOCK_STREAM, 0);
33
34      if(sock == -1)
35          err_handler("socket() error");
36
37      memset(&serv_addr, 0, sizeof(serv_addr));
38      serv_addr.sin_family = AF_INET;
39      serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
40      serv_addr.sin_port = htons(atoi(argv[2]));
41
42      if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
43          err_handler("connect() error");
44
45      while(read_len = read(sock, &msg[idx++], 1))
46      {
47          if(read_len == -1)
48              err_handler("read() error!");
49
50          str_len += read_len;
51      }
52
53      printf("msg from serv : %s\n", msg);
54      printf("read count : %d\n", str_len);
55      close(sock);
56
57      return 0;
58  }
                                                    58,1          Bot
```

- ❖ socket 을 만들고, 초기화를 해 준다.
- ❖ connect 를 통해 server 와 연결한다.
- ❖ 기존과 다르게, 문자 1개씩 읽어온다.
- ❖ 해당 방식을 통해 데이터 전송 중 깨지게 되는 것을 방지할 수 있다.

- ❖ read_len == -1 일 경우, idx-- 해야 정상작동을 할 것 같다.

cross matching

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```
 1 #include <stdio.h>
 2 #include <arpa/inet.h>
 3
 4 int main(void)
 5 {
 6     unsigned short host_port = 0x5678;
 7     unsigned short net_port;
 8     unsigned long host_addr = 0x87654321;
 9     unsigned long net_addr;
10
11     net_port = htons(host_port);
12     net_addr = htonl(host_addr);
13
14     printf("Host Ordered Port : %#x\n", host_port);
15     printf("Network Ordered Port : %#x\n", net_port);
16     printf("Host Ordered Address : %#lx\n", host_addr);
17     printf("Network Ordered Address : %#lx\n", net_addr);
18
19     return 0;
20 }
~
-- INSERT --                                    1,1           All
```

- ❖ host 주소 값이 network 주소로 넘어갈 때 cross matching 되는 것을 확인하는 코드이다.
- ❖ 각각 port (2 byte) 와 addr (4 byte) 의 결과를 확인할 수 있다.
- ❖ network 주소로 넘어갈 때, cross matching 이 되는 것을 확인할 수 있다.

cross matching

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/ln_kernel9

```c
#include <stdio.h>
#include <arpa/inet.h>

int main(int argc, char **argv)
{
    char *addr1 = "3.7.5.9";
    char *addr2 = "1.3.5.7";

    unsigned long conv_addr = inet_addr(addr1);
    if(conv_addr == INADDR_NONE)
        printf("Error!\n");
    else
        printf("Network Ordered Integer Addr : %#lx\n", conv_addr);

    conv_addr = inet_addr(addr2);
    if(conv_addr == INADDR_NONE)
        printf("Error!\n");
    else
        printf("Network Ordered Integer Addr : %#lx\n", conv_addr);

    return 0;
}
```

1,1 All

- cross matching 은 2 가지 형태가 존재한다.
- big-endian : 데이터를 순서대로 받는다.
- little-endian : 데이터 순서를 반대로 받는다.