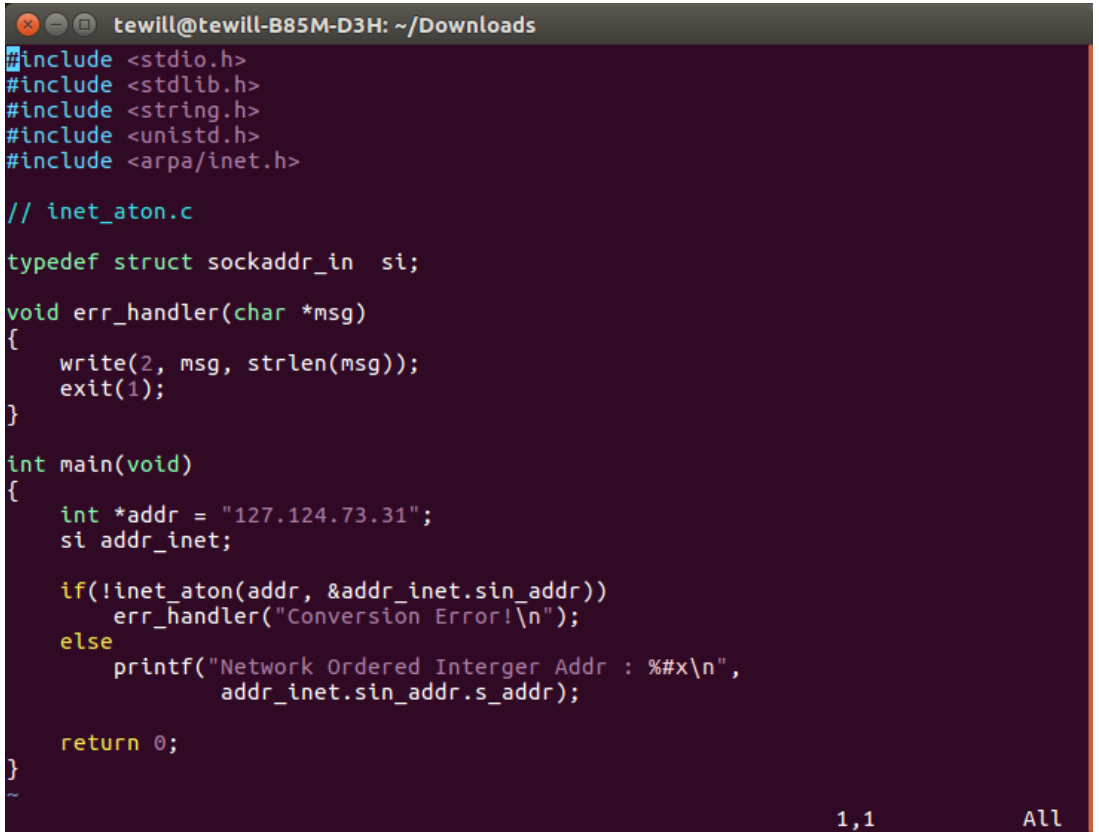
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

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

Network addr



- ❖ Network addr 를 확인하는 코드이다.
- ❖ addr 을 Network 로 변환 시킨 후 결과를 출력해 보면, Big-endian 로 변환된 것을 볼 수 있다.

- ❖ 네트워크는 항상 Big-endian 이다.
- ❖ 인텔은 항상 Little-endian 이며, ARM 은 둘 중 하나로 설정할 수 있다.
- ❖ 따라서, 호스트에서 네트워크 주소로 변환할 때는 자신이 Big-endian 인지 Little-endian 인지 확인 후Big-endian 으로 변환한다.
- ❖ 네트워크에서 다시 호스트로 이동 시, 위 동작을 반대로 진행한다.

Network addr

tewill@tewill-B85M-D3H: ~/Downloads

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

//inet_ntoa.c

typedef struct sockaddr_in  si;

int main(int argc, char **srgv)
{
    si addr1, addr2;
    char *str;
    char str_arr[32] = {0};

    addr1.sin_addr.s_addr = htonl(0x10203040);
    addr2.sin_addr.s_addr = htonl(0x12345678);

    str = inet_ntoa(addr1.sin_addr);
    strcpy(str_arr, str);
    printf("Not 1: %s\n", str);

    inet_ntoa(addr2.sin_addr);
    printf("Not 2: %s\n", str);
    printf("Not 3: %s\n", str_arr);

    return 0;
}
```

1,1 All

- htonl() : Host 를 Network addr 로 변환한다.
- inet_ntoa() : Network addr 를 corss matching 해서 반환한다.

- Not 2 에서 우리가 원하는 결과 값으로 나오는 이유는 컴파일러가 알아서 처리했기 때문이다. 명확하게 하기 위해서는 ‘ str = ’ 처리를 해야 한다.

echo_server

tewill@tewill-B85M-D3H: ~/Downloads

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

// echo_server.c

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE    1024

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

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

    char msg[BUF_SIZE];

    si serv_addr, clnt_addr;
    socklen_t clnt_addr_size;
```

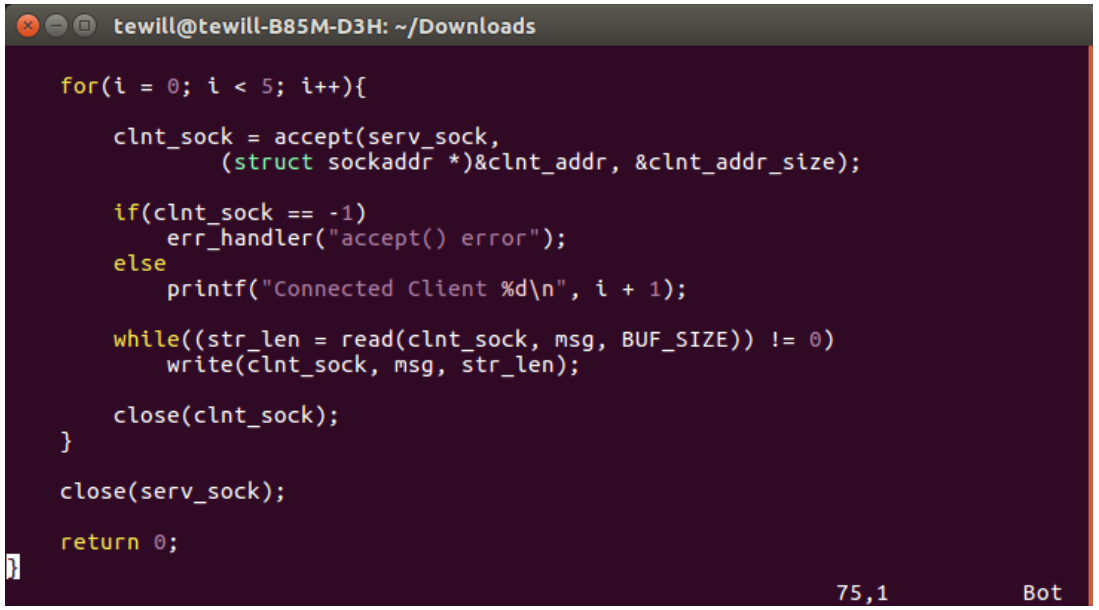
1,1 Top

tewill@tewill-B85M-D3H: ~/Downloads

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

    serv_sock = socket(PF_INET, SOCK_STREAM, 0);

    if(serv_sock == -1)
        err_handler("socket() errer");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));

    if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("bind() errer");

    if(listen(serv_sock, 5) == -1)
        err_handler("listen() error");

    clnt_addr_size = sizeof(clnt_addr);
```

31,1 60%

echo_server



- 기본적으로 server 형식과 동일하다.
- listen 으로 5명을 받기 때문에, for 문을 통해 5번 accept 를 실시한다.
- 원하는 방향으로 작동하지 않는 이유는 read 를 통하여 블럭 되어 있기 때문이다.
- 따라서, 클라이언트가 종료될 때, read 의 리턴 값이 0 이 되면서 한번에 출력되는 것을 볼 수 있다.

echo_client

tewill@tewill-B85M-D3H: ~/Downloads

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

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE    1024

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

int main(int argc, char **argv)
{
    int sock, str_len;
    si serv_addr;
    char msg[32];
    char *m = "Input Message(q to quit): ";
```

1,1 Top

tewill@tewill-B85M-D3H: ~/Downloads

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

    sock = socket(PF_INET, SOCK_STREAM, 0);

    if(sock == -1)
        err_handler("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("connect() error");
    else
        puts("Connected ......");
```

26,0-1 50%

echo_client

- Client 역시 기본적인 형식은 동일하다.
- fputs 와 fgets 는 write 나 read 와 동일하다.

- 1 입력 받은 것을 확인 후, q 나 Q 면 종료한다.
- sock 의 내용을 msg 에 저장한다.
- msg 내용을 sock 에 저장한다. (echo)
- msg 를 출력한다.

OP_server

tewill@tewill-B85M-D3H: ~/Downloads

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

// op_server.c

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE    1024
// size 크기 4
#define OPSZ        4


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

1,1 Top

tewill@tewill-B85M-D3H: ~/Downloads

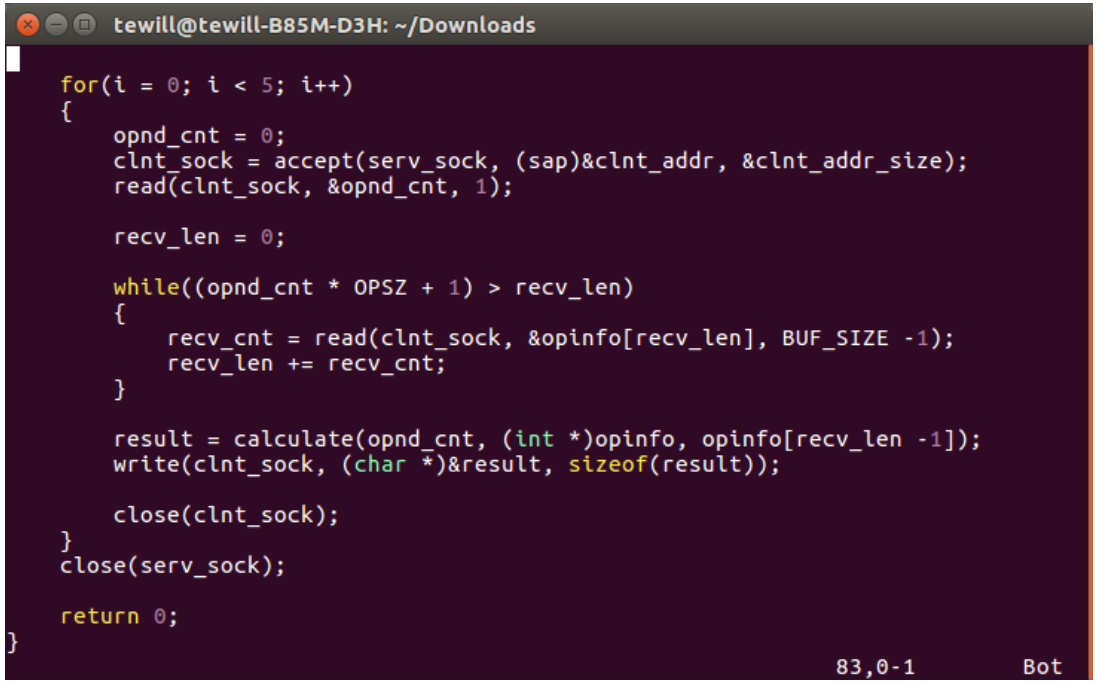
```c
int main(int argc, char **argv)
{
    int serv_sock, clnt_sock;
    char opinfo[BUF_SIZE];

    int result, opnd_cnt, i;
    int recv_cnt, recv_len;

    si serv_addr, clnt_addr;
    socklen_t clnt_addr_size;

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

    serv_sock = socket(PF_INET, SOCK_STREAM, 0);

    if(serv_sock == -1)
        err_handler("socket() error");
```

tewill@tewill-B85M-D3H: ~/Downloads

```c
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));

    if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("hind() error");


    if(listen(serv_sock, 5) == -1)
        err_handler("listen() error");

    clnt_addr_size = sizeof(clnt_addr);
```

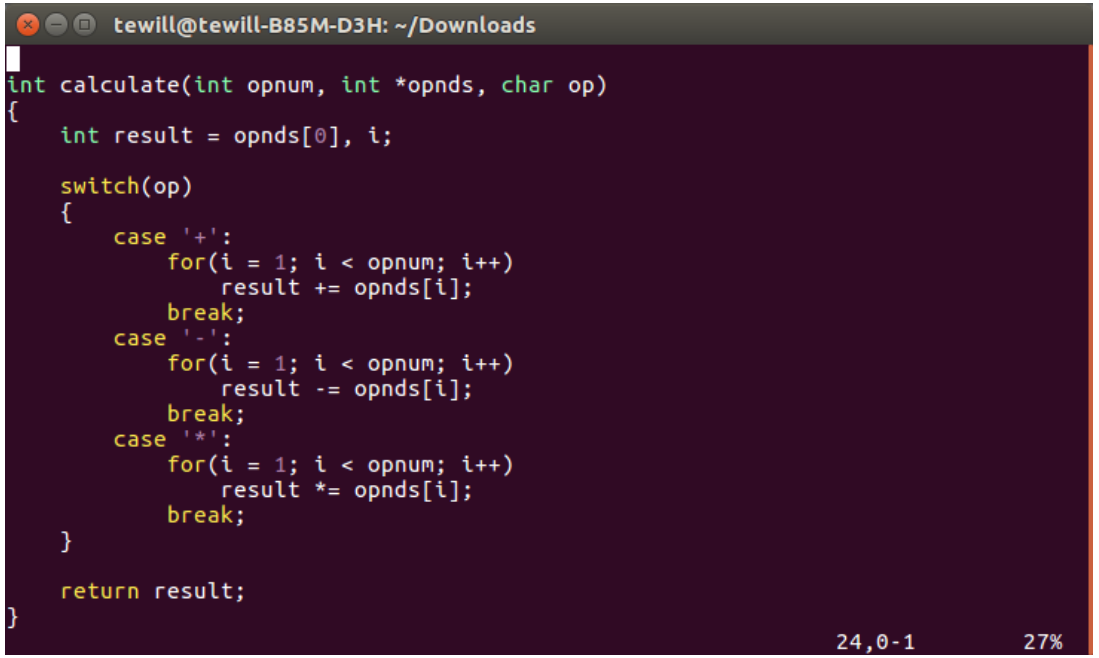
69,0-1 73%

OP_server



- server 의 기본 형식은 동일하다.
- listen 으로 받은 것을 반복문을 통해 accept 한다.
- 반복문을 통해 sock 데이터를 읽는다.
- 만든 함수 calculate 에 각각의 인자를 넣는다.
- 처리 결과를 clnt_sock 에 저장한다.

OP_server



- 첫번째 인자는 숫자들의 갯수를 나타낸다.
- 두번째 인자는 계산하는 숫자들의 배열이다.
- 세번째 인자는 하고자 하는 연산을 나타낸다.

OP_client

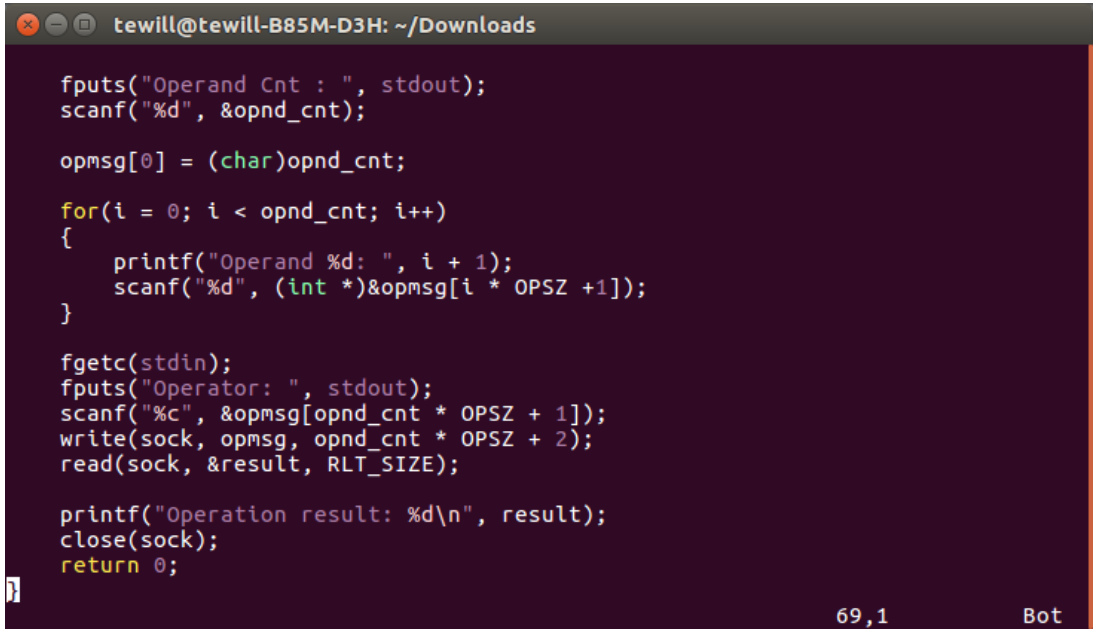
tewill@tewill-B85M-D3H: ~/Downloads

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

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;

#define BUF_SIZE 1024
#define RLT_SIZE    4
#define OPSZ        4

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

int main(int argc, char **argv)
{
    int i, sock, result, opnd_cnt;
    char opmsg[BUF_SIZE] = {0};
    si serv_addr;
                                                    1,1           Top
```

tewill@tewill-B85M-D3H: ~/Downloads

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

    sock = socket(PF_INET, SOCK_STREAM, 0);

    if(sock == -1)
        err_handler("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("connect() error");
    else
        puts("Connected ......");
                                                    27,0-1        54%
```

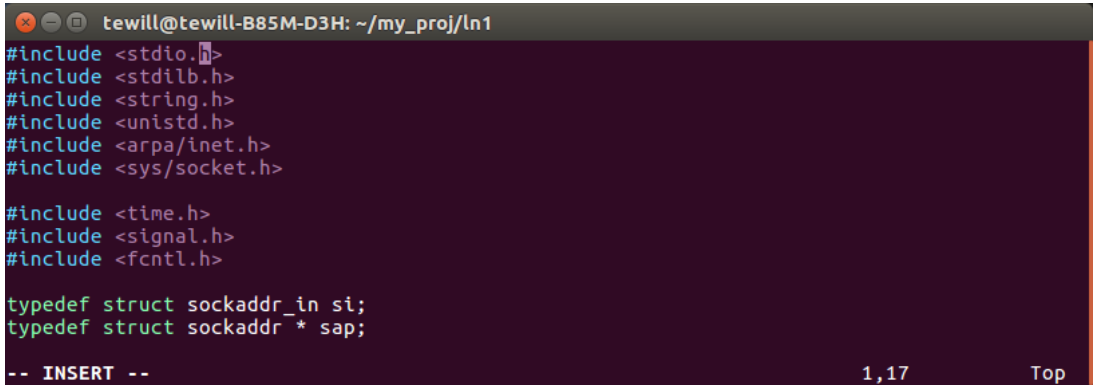
OP_client



- ❖ 기존 client 와 동일한 형식이다.
- ❖ 사용할 숫자의 갯수를 받아온다.
- ❖ 그 숫자만큼 반복문을 실행하며, 해당 데이터를 opmsg 의 배열에 저장한다.
- ❖ 마지막으로 연산하고자 하는 부호를 입력한다.
- ❖ write 를 통해 server 에 데이터를 전송한다.
- ❖ 그 후 결과를 입력 받는다.

- ❖ 출력을 통해 결과를 확인한다.

Up and Down


tewill@tewill-B85M-D3H: ~/my_proj/ln1
#include <stdio.h>
#include <stdilb.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <time.h>
#include <signal.h>
#include <fcntl.h>
typedef struct sockaddr_in si;
typedef struct sockaddr * sap;
-- INSERT -- 1,17 Top


tewill@tewill-B85M-D3H: ~/my_proj/ln1
void err_handler(char *msg)
{
fputs(msg, stderr);
fputc('\n', stderr);
exit(1);
}
int main(int argc, char **argv)
{
int i, pid[5] = {0};
int serv_sock, clnt_sock;
si serv_addr, clnt_addr;
socklen_t clnt_addr_size;
srand(time(NULL));
char buf[1024];
if(argc != 3)
{
printf("use : %s <IP> <port>\n", argv[0]);
exit(1);
}
-- INSERT -- 50,1 56%

Up and Down

tewill@tewill-B85M-D3H: ~/my_proj/ln1

```
        serv_sock = socket(PF_INET, SOCK_STREAM, 0);

        if(sock = -1)
                err_handler("socket() error");

        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_addr.sin_port = htons(atoi(argv[2]));

        if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
                err_handler("bind() error");

        if(listen(serv_sock, 5) == -1)
                err_handler("listen() error");

        clnt_addr_size = sizeof(clnt_addr);

        for(i = 0; i < 5 ; i++) {

                pid[i] = fork();

                if(pid[i] > 0)

                else {
                        clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);

                        if(clnt_sock == -1)
                                err_handler("accept() err");
                        //

                }
                close(clnt_sock);
        }

        close(serv_sock);
        return 0;
}
-- INSERT --                                                    74,1          Bot
```

Up and Down

tewill@tewill-B85M-D3H: ~/my_proj/ln1

```
void sig_handler(int signo)
{
        printf("Time over\n");
        //alarm(3);
}

int make_game(int max)
{
        int ref;
        ref = (rand() % max) + 1;
        return ref;
}

void play_game(int ref)
{
        char buf[32] = {0};
        int num;

        signal(SIGALRM, sig_handler);
        alarm(3);
        read(0, buf, sizeof(buf));
        alarm(0);
        num = atoi(buf);

        if(num == ref)
        {
                // 정답
        } else {
                if(num > ref) {
                        // 크다 출력
                } else {
                        // 작다 출력
                }
        }
}
-- INSERT --                                                    14,1          17%
```