Assignment3

김성록(KIM SEONGROK)

2016116783

Example1

`file_server.c`

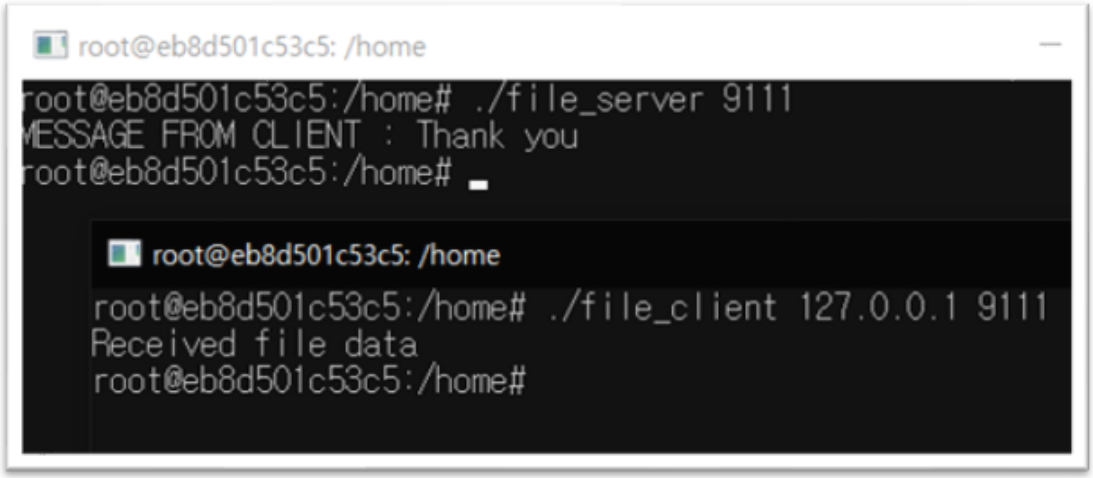
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);

int main(int argc, char *argv[]){
    int serv_sd, clnt_sd;
    FILE * fp;
    char buf[BUF_SIZE];
    int read_cnt;
    struct sockaddr_in serv_adr, clnt_adr;
    socklen_t clnt_adr_sz;

    if(argc!=2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
    }
    fp = fopen("file_server.c","rb");
    serv_sd = socket(PF_INET, SOCK_STREAM, 0);
    if(serv_sd==-1)
        error_handling("socket() error");

    memset(&serv_adr,0,sizeof(serv_adr));
    serv_adr.sin_family = AF_INET;
    serv_adr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_adr.sin_port = htons(atoi(argv[1]));
    if(bind(serv_sd, (struct sockaddr*)&serv_adr, sizeof(serv_adr))==-1)
        error_handling("bind() error");

    if(listen(serv_sd,5)==-1)
        error_handling("listen() error");

    clnt_adr_sz = sizeof(clnt_adr);
    clnt_sd = accept(serv_sd, (struct sockaddr*)&clnt_adr, &clnt_adr_sz);
```

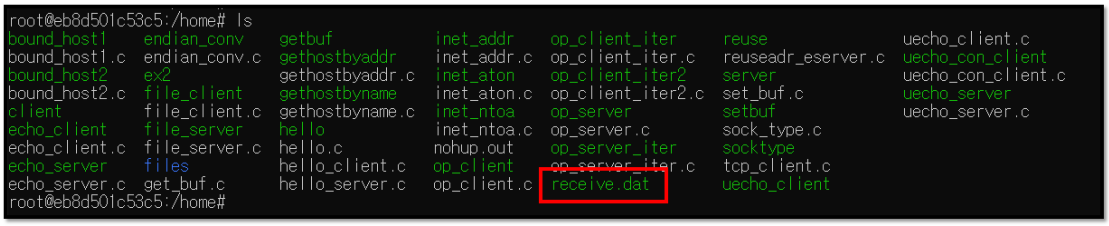
```
    while(1){
        read_cnt = fread((void*)buf, 1, BUF_SIZE, fp);
        if(read_cnt < BUF_SIZE){
            write(clnt_sd, buf, read_cnt);
            break;
        }
        write(clnt_sd, buf, BUF_SIZE);
    }
    shutdown(clnt_sd, SHUT_WR);
    read(clnt_sd, buf, BUF_SIZE);
    printf("MESSAGE FROM CLIENT : %s \n",buf);

    fclose(fp);
    close(clnt_sd);
    close(serv_sd);
    return 0;
}

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

file_client.c

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

#define BUF_SIZE 30

void error_handling(char *message);

int main(int argc, char *argv[]){
    int sd;
    FILE * fp;
    char buf[BUF_SIZE];
    int read_cnt;
    struct sockaddr_in serv_adr;

    if(argc!=3){
        printf("Usage : %s <IP> <port>\n", argv[0]);
        exit(1);
    }
    fp = fopen("receive.dat","wb");
    sd = socket(PF_INET, SOCK_STREAM, 0);

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

```
    connect(sd, (struct sockaddr*)&serv_adr, sizeof(serv_adr));

    while((read_cnt = read(sd, buf, BUF_SIZE))!=0)
        fwrite((void*)buf, 1, read_cnt, fp);

    puts("Received file data");
    write(sd, "Thank you", 10);
    fclose(fp);
    close(sd);
    return 0;
}

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





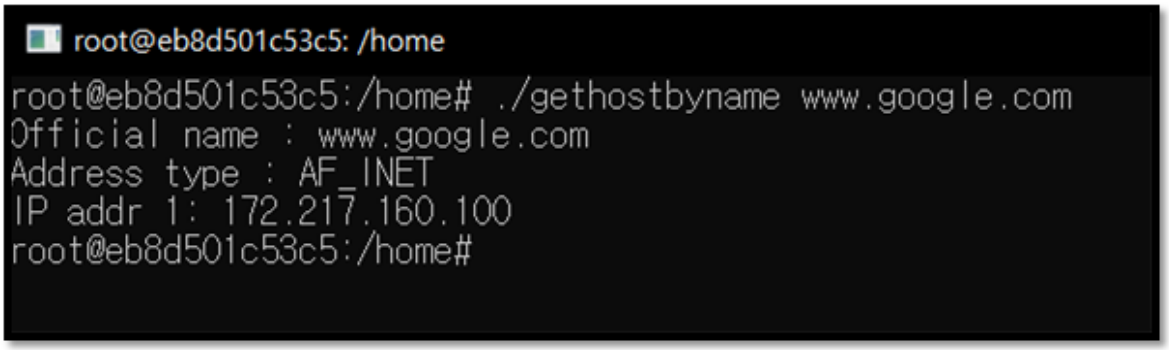
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);

int main(int argc, char *argv[])
{
        int serv_sd, clnt_sd;
        FILE *fp;
        char buf[BUF_SIZE];
        int read_cnt;
        struct sockaddr_in serv_adr, clnt_adr;
        socklen_t clnt_adr_sz;

        if (argc != 2)
        {
                printf("Usage : %s <port>₩n", argv[0]);
                exit(1);
        }
        fp = fopen("file_server.c", "rb");
        serv_sd = socket(PF_INET, SOCK_STREAM, 0);
        if (serv_sd == -1)
                error_handling("socket() error");

        memset(&serv_adr, 0, sizeof(serv_adr));
        serv_adr.sin_family = AF_INET;
        serv_adr.sin_addr.s_addr = htonl(INADDR_ANY);
        serv_adr.sin_port = htons(atoi(argv[1]));
        if (bind(serv_sd, (struct sockaddr *)&serv_adr, sizeof(serv_adr)) == -1)
                error_handling("bind() error");

        if (listen(serv_sd, 5) == -1)
                error_handling("listen() error");

        clnt_adr_sz = sizeof(clnt_adr);
        clnt_sd = accept(serv_sd, (struct sockaddr *)&clnt_adr, &clnt_adr_sz);
        while (1)
        {
                read_cnt = fread((void *)buf, 1, BUF_SIZE, fp);
"receive.dat" 68L, 1454C
```

Example2

`gethostbyname.c`

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

void error_handling(char *message);
```

```
int main(int argc, char *argv[])
{
    int i;
    struct hostent *host;

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

    host = gethostbyname(argv[1]);

    if (!host)
        error_handling("gethost...error");

    printf("Official name : %s\n", host->h_name);
    for (i = 0; host->h_aliases[i]; i++)
        printf("Aliases %d: %s\n", i+1, host->h_aliases[i]);

    printf("Address type : %s\n", host->h_addrtype == AF_INET ? "AF_INET" : "AF_INET6");
    for (i = 0; host->h_addr_list[i]; i++)
        printf("IP addr %d: %s\n", i+1, inet_ntoa(*(struct in_addr *)host->h_addr_list[i]));
}

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



Example3

`gethostbyaddr.c`

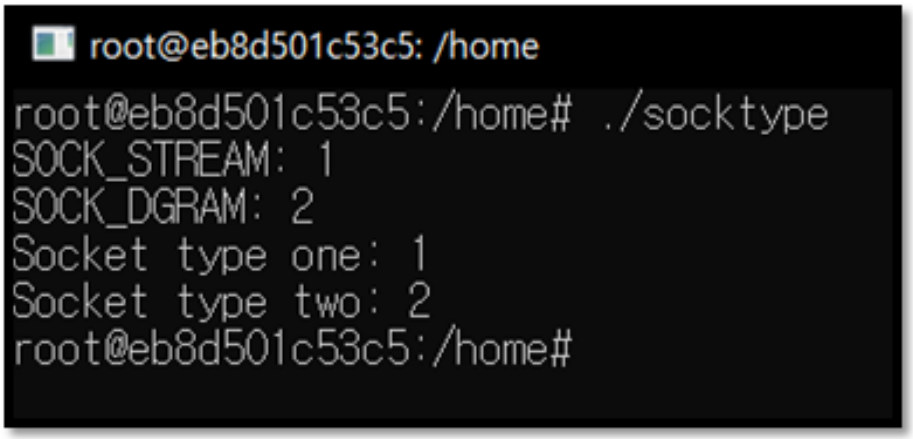
```
#include <stdio.h>
#include <stdlib.h>
```

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

void error_handling(char *message);

int main(int argc, char *argv[])
{
    int i;
    struct hostent *host;
    struct sockaddr_in addr;

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

    memset(&addr, 0, sizeof(addr));
    addr.sin_addr.s_addr = inet_addr(argv[1]);
    host = gethostbyaddr((char*)&addr.sin_addr,4,AF_INET);

    if (!host)
        error_handling("gethost...error");

    printf("Official name : %s\n", host->h_name);
    for (i = 0; host->h_aliases[i]; i++)
        printf("Aliases %d: %s\n", i+1, host->h_aliases[i]);

    printf("Address type : %s\n", host->h_addrtype == AF_INET ? "AF_INET" : "AF_INET6");
    for (i = 0; host->h_addr_list[i]; i++)
        printf("IP addr %d: %s\n", i+1, inet_ntoa(*(struct in_addr *)host->h_addr_list[i]));
}

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

Example4

`sock_type.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>

void error_handling(char *message);

int main(int argc, char* argv[]){
    int tcp_sock, udp_sock;
    int sock_type;
    socklen_t optlen;
    int state;

    optlen = sizeof(sock_type);
    tcp_sock = socket(PF_INET, SOCK_STREAM, 0);
    udp_sock = socket(PF_INET, SOCK_DGRAM, 0);
    printf("SOCK_STREAM: %d\n", SOCK_STREAM);
    printf("SOCK_DGRAM: %d\n", SOCK_DGRAM);

    state = getsockopt(tcp_sock, SOL_SOCKET, SO_TYPE, (void*)&sock_type, &optlen);

    if (state)
        error_handling("getsockopt() error");
    printf("Socket type one: %d\n", sock_type);

    state = getsockopt(udp_sock, SOL_SOCKET, SO_TYPE, (void*)&sock_type, &optlen);

    if (state)
        error_handling("getsockopt() error");
    printf("Socket type two: %d\n", sock_type);

    return 0;

}


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



Example5

`get_buf.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>

void error_handling(char *message);

int main(int argc, char* argv[]){
    int sock;
    int snd_buf, rcv_buf, state;
    socklen_t len;

    sock = socket(PF_INET, SOCK_STREAM, 0);
    len = sizeof(snd_buf);
    state = getsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void*)&snd_buf, &len);

    if (state)
        error_handling("getsockopt() error");

    len = sizeof(rcv_buf);
    state = getsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void*)&rcv_buf, &len);

    if (state)
        error_handling("getsockopt() error");

    printf("Input buffer size: %d\n",rcv_buf);
    printf("Output buffer size: %d\n",snd_buf);
    return 0;
}


void error_handling(char *message)
{
```

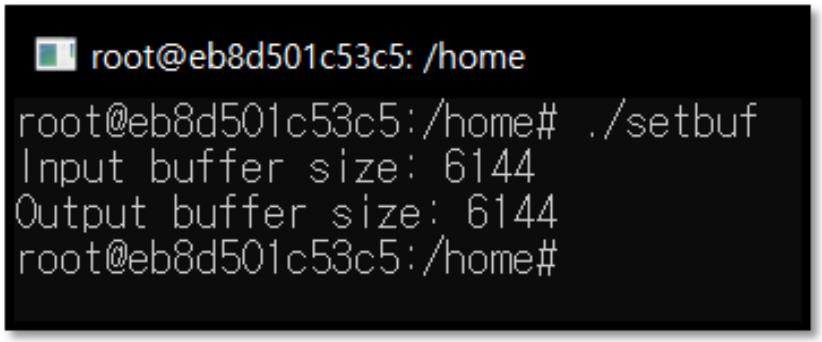
```
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

Example6

`set_buf.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>

void error_handling(char *message);

int main(int argc, char* argv[]){
    int sock;
    int snd_buf = 1024*3;
    int rcv_buf = 1024*3;
    int state;
    socklen_t len;

    sock = socket(PF_INET, SOCK_STREAM, 0);

    state = setsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void*)&rcv_buf , sizeof(rcv_buf));

    if (state)
        error_handling("setsockopt() error");

    state = setsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void*)&snd_buf , sizeof(snd_buf));

    if (state)
        error_handling("setsockopt() error");

    len = sizeof(snd_buf);
    state = getsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void*)&snd_buf , &len);
```

```
    if (state)
        error_handling("getsockopt() error");

    len = sizeof(rcv_buf);
    state = getsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void*)&rcv_buf , &len);

    if (state)
        error_handling("getsockopt() error");

    printf("Input buffer size: %d\n",rcv_buf);
    printf("Output buffer size: %d\n",snd_buf);
    return 0;
}


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



Example7

reuseadr_eserver.c

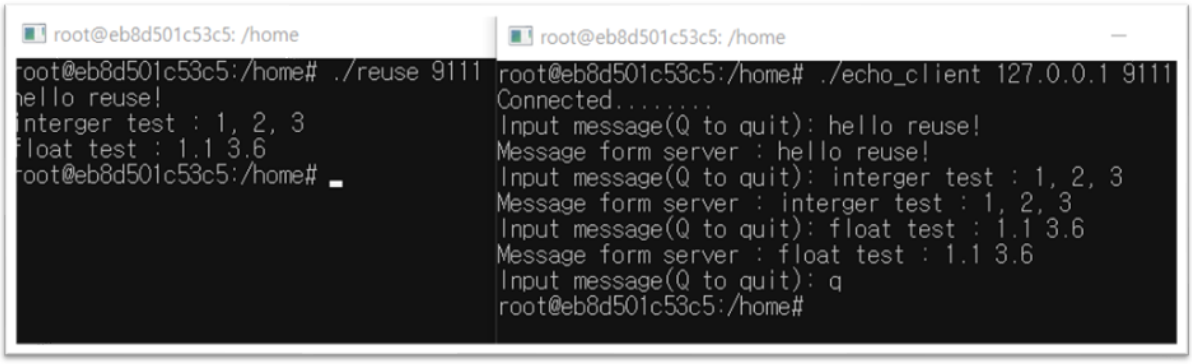
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define TRUE 1
#define False 0

void error_handling(char *message);

int main(int argc, char* argv[]){
```

```
    int serv_sock, clnt_sock;
    char message[30];
    int option, str_len;
    socklen_t optlen, clnt_adr_sz;
    struct sockaddr_in serv_adr, clnt_adr;

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

    serv_sock = socket(PF_INET, SOCK_STREAM, 0);
    if (serv_sock==-1)
        error_handling("socket() Error");

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

    if (bind(serv_sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr)))
        error_handling("bind() error");

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

    clnt_adr_sz = sizeof(clnt_adr);
    clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_adr, &clnt_adr_sz);
    while((str_len=read(clnt_sock, message, sizeof(message)))!=0)
    {
        write(clnt_sock, message,str_len);
        write(1,message, str_len);
    }

    close(clnt_sock);
    close(serv_sock);
}

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

echo_client.c

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

#define BUF_SIZE 1024
```

```
void error_handling(char *message);

int main(int argc, char* argv[]){
        int sock;
        struct sockaddr_in serv_addr;
        char message[BUF_SIZE];
        int str_len;
        if(argc != 3){
                printf("Usage: %s <IP> <port>\n", argv[0]);
        }
        sock = socket(PF_INET, SOCK_STREAM, 0);
        if(sock==-1)
                error_handling("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, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("connect() error!");
        else
                puts("Connected........");

        while(1){
                fputs("Input message(Q to quit): ", stdout);
                fgets(message, BUF_SIZE, stdin);

                if(!strcmp(message, "q\n") || !strcmp(message, "Q\n"))
                        break;
                write(sock, message, strlen(message));
                str_len = read(sock, message, BUF_SIZE-1);
                message[str_len] = 0;
                printf("Message form server : %s", message);
        }
        close(sock);
        return 0;
}



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



Problem1

`Server.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netdb.h>

void error_handling(char *message);

int main(int argc, char *argv[])
{
    int i;
    struct hostent *host;

    int serv_sock, clnt_sock;
    char message[30];
    int option, str_len;
    socklen_t optlen, clnt_adr_sz;
    struct sockaddr_in serv_adr, clnt_adr;
    char* ipaddr;
    char* temp;

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

    serv_sock = socket(PF_INET, SOCK_STREAM, 0);
    if (serv_sock==-1)
        error_handling("socket() Error");

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

```
    if (bind(serv_sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr)))
        error_handling("bind() error");

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

    clnt_adr_sz = sizeof(clnt_adr);
    clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_adr, &clnt_adr_sz);

    while((str_len=read(clnt_sock, message, sizeof(message)))!=0)
    {
        printf("from client : %s\n",message);
        for(int ii=0;ii<sizeof(message);ii++){
            if (message[ii]=='\n')
                message[ii]='\0';
        }
        host = gethostbyname(message);
        if (!host)
            error_handling("gethost...error");

        ipaddr = inet_ntoa(*(struct in_addr *)host->h_addr_list[0]);

        write(clnt_sock, ipaddr,str_len);
        write(1,ipaddr, str_len);
    }

    close(clnt_sock);
    close(serv_sock);
}

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

Client.c

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

#define BUF_SIZE 1024

void error_handling(char *message);

int main(int argc, char* argv[]){
        int sock;
        struct sockaddr_in serv_addr;
        char message[BUF_SIZE];
        int str_len;
        if(argc != 3){
```

```
                printf("Usage: %s <IP> <port>\n", argv[0]);
        }
        sock = socket(PF_INET, SOCK_STREAM, 0);
        if(sock==-1)
                error_handling("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, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("connect() error!");
        else
                puts("Connected........");

        while(1){
                fputs("Input message(Q to quit): ", stdout);
                fgets(message, BUF_SIZE, stdin);

                if(!strcmp(message, "q\n") || !strcmp(message, "Q\n"))
                        break;
                write(sock, message, strlen(message));
                str_len = read(sock, message, BUF_SIZE-1);
                message[str_len] = 0;
                printf("%s\n", message);
        }
        close(sock);
        return 0;
}



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