Assignment1

김성록(KIM SEONGROK)

2016116783

Problem1

Primitive	Meaning
Socket	소켓 생성 (create socket object)
Bind	소켓 주소 할당 (allocate socket's address)
Listen	연결요청 대기상태 (waiting for connect)
Accept	연결허용 (accept connection)
Connect	연결요청 (request connection)
Send	데이터 송신 (send data)
Receive	데이터 수신 (receive data)
Close	연결종료 (shut connection down)

Problem2

hello_server.c

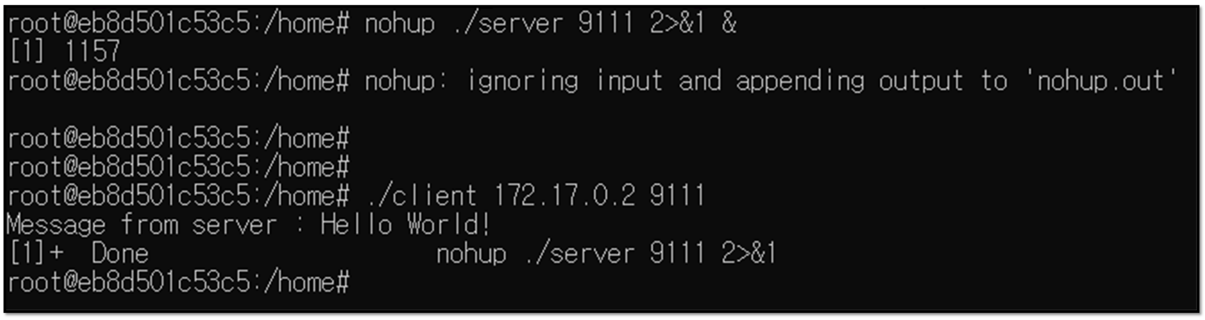
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

void error_handling(char *message);

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

        struct sockaddr_in serv_addr;
        struct sockaddr_in clnt_addr;
        socklen_t clnt_addr_size;

        char message[] = "Hello World!";
```

```
	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_addr, 0, sizeof(serv_addr));
	serv_addr.sin_family = AF_INET;
	serv_addr.sin_addr.s_addr = inet_addr("172.17.0.2");
	serv_addr.sin_port = htons(atoi(argv[1]));
	if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
		error_handling("bind() error");
	if(listen(serv_sock, 5)==-1)
		error_handling("listen() error");
	clnt_addr_size = sizeof(clnt_addr);
	clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
	if(clnt_sock==-1)
		error_handling("accept() error");

	write(clnt_sock, message, sizeof(message));
	close(clnt_sock);
	close(serv_sock);
	return 0;
}

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

hello_client.c

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

void error_handling(char *message);
int main(int argc, char* argv[]){
	int sock;
	struct sockaddr_in serv_addr;
	char message[30];
	int str_len;
	if(argc!=3){
		printf("Usage: %s <IP> <port>\n", argv[0]);
		exit(1);
	}
	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!");
        str_len = read(sock, message, sizeof(message)-1);
        if(str_len==-1)
                error_handling("read() error!");
        printf("Message from server : %s\n", message);
        close(sock);
        return 0;
}

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

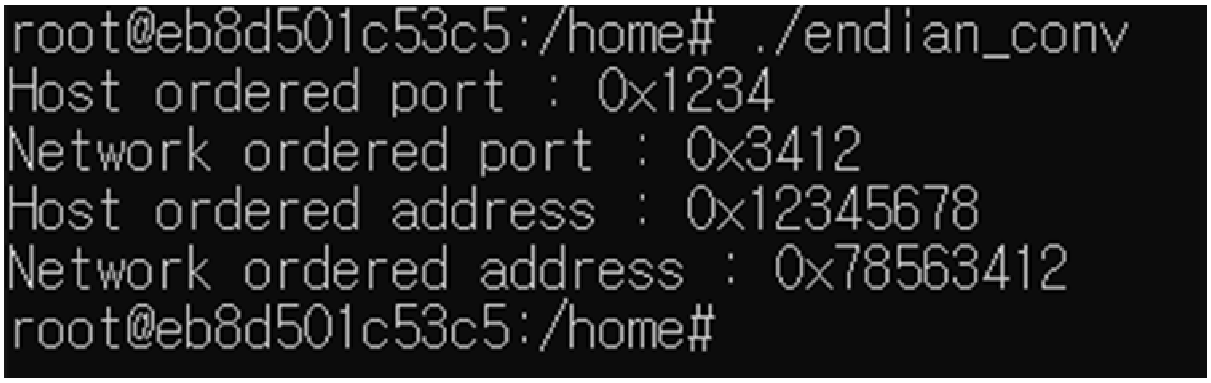


```
root@eb8d501c53c5:/home# nohup ./server 9111 2>&1 &
[1] 1157
root@eb8d501c53c5:/home# nohup: ignoring input and appending output to 'nohup.out'

root@eb8d501c53c5:/home#
root@eb8d501c53c5:/home#
root@eb8d501c53c5:/home# ./client 172.17.0.2 9111
Message from server : Hello World!
[1]+  Done                    nohup ./server 9111 2>&1
root@eb8d501c53c5:/home#
```

Examples

1. Endian_conv.c

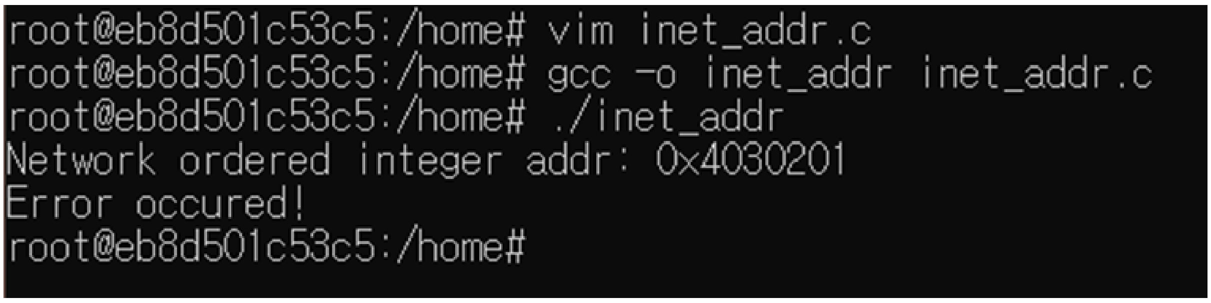
```
#include <stdio.h>
#include <arpa/inet.h>

int main(int argc, char *argv[]){
        unsigned short host_port = 0x1234;
        unsigned short net_port;
        unsigned long host_addr = 0x12345678;
        unsigned long net_addr;

        net_port = htons(host_port);
        net_addr = htonl(host_addr);
        printf("Host ordered port : %#x \n", host_port);
        printf("Network ordered port : %#x \n", net_port);
        printf("Host ordered address : %#lx \n", host_addr);
        printf("Network ordered address : %#lx \n", net_addr);

        return 0;
}
```



```
root@eb8d501c53c5:/home# ./endian_conv
Host ordered port : 0x1234
Network ordered port : 0x3412
Host ordered address : 0x12345678
Network ordered address : 0x78563412
root@eb8d501c53c5:/home#
```

2. inet_addr.c

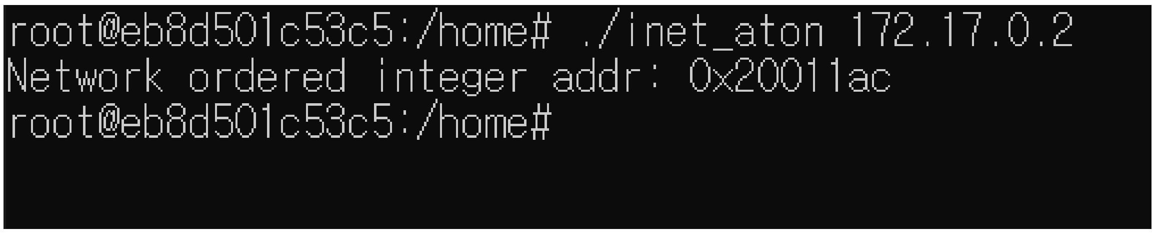
```
#include <stdio.h>
#include <arpa/inet.h>

int main(int argc, char *argv[]){
        char *addr1 = "1.2.3.4";
        char *addr2 = "1.2.3.256";

        unsigned long conv_addr = inet_addr(addr1);
        if(conv_addr == INADDR_NONE)
                printf("Error occured!\n");
        else
                printf("Network ordered integer addr: %#lx\n", conv_addr);
        conv_addr = inet_addr(addr2);
        if(conv_addr == INADDR_NONE)
                printf("Error occured!\n");
        else
                printf("Networkd ordered interger addr: %#lx \n", conv_addr);
        return 0;

}
```



```
root@eb8d501c53c5:/home# vim inet_addr.c
root@eb8d501c53c5:/home# gcc -o inet_addr inet_addr.c
root@eb8d501c53c5:/home# ./inet_addr
Network ordered integer addr: 0x4030201
Error occured!
root@eb8d501c53c5:/home#
```

3. inet_aton.c

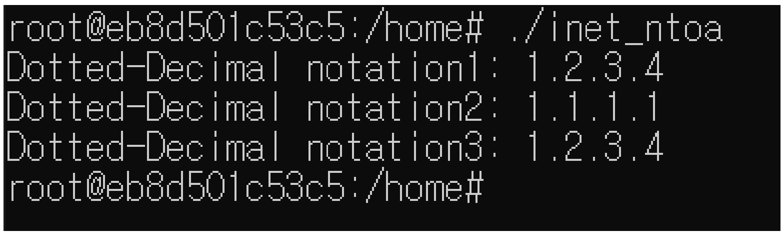
```c
#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>

void error_handling(char *message);

int main(int argc, char *argv[]){
        struct sockaddr_in addr_inet;
        if(argc!=2){
                printf("Usage: %s <IP>\n", argv[0]);
                exit(1);
        }
        if(!inet_aton(argv[1], &addr_inet.sin_addr))
                error_handling("Conversion error");
        else
                printf("Network ordered integer addr: %#x\n", addr_inet.sin_addr.s_addr);
        return 0;
}

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



```
root@eb8d501c53c5:/home# ./inet_aton 172.17.0.2
Network ordered integer addr: 0x20011ac
root@eb8d501c53c5:/home#
```

4. inet_ntoa.c

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

int main(int argc, char *argv[]){
        struct sockaddr_in addr1,addr2 ;
        char *str_ptr;
        char str_arr[20];
        addr1.sin_addr.s_addr = htonl(0x1020304);
        addr2.sin_addr.s_addr = htonl(0x1010101);

        str_ptr = inet_ntoa(addr1.sin_addr);
        strcpy(str_arr, str_ptr);
        printf("Dotted-Decimal notation1: %s \n", str_ptr);

        inet_ntoa(addr2.sin_addr);
        printf("Dotted-Decimal notation2: %s \n", str_ptr);
        printf("Dotted-Decimal notation3: %s \n", str_arr);
        return 0;
}
```



```
root@eb8d501c53c5:/home# ./inet_ntoa
Dotted-Decimal notation1: 1.2.3.4
Dotted-Decimal notation2: 1.1.1.1
Dotted-Decimal notation3: 1.2.3.4
root@eb8d501c53c5:/home#
```

Q: Why it does not display "Dotted-Decimal notation2 : 1. 2. 3. 4"

> **A** : Because return value of `inet_ntoa()` (pointer) is stored in internal static buffer in the function. When we call `inet_ntoa()` function again, the buffer is re-write with new value(pointer). So we have to keep previous return value with variable if we want.

5-1. hello_server.c

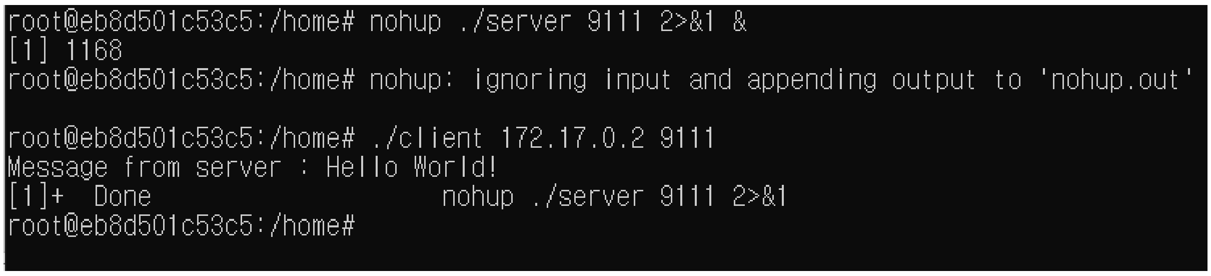
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

void error_handling(char *message);
```

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

        struct sockaddr_in serv_addr;
        struct sockaddr_in clnt_addr;
        socklen_t clnt_addr_size;

        char message[] = "Hello World!";
        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_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, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
                error_handling("bind() error");
        if(listen(serv_sock, 5)==-1)
                error_handling("listen() error");
        clnt_addr_size = sizeof(clnt_addr);
        clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
        if(clnt_sock==-1)
                error_handling("accept() error");

        write(clnt_sock, message, sizeof(message));
        close(clnt_sock);
        close(serv_sock);
        return 0;
}

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

5-2. hello_client.c

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

void error_handling(char *message);
```

```
int main(int argc, char* argv[]){
        int sock;
        struct sockaddr_in serv_addr;
        char message[30];
        int str_len;
        if(argc!=3){
                printf("Usage: %s <IP> <port>\n", argv[0]);
                exit(1);
        }
        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!");
        str_len = read(sock, message, sizeof(message)-1);
        if(str_len==-1)
                error_handling("read() error!");
        printf("Message from server : %s\n", message);
        close(sock);
        return 0;
}

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

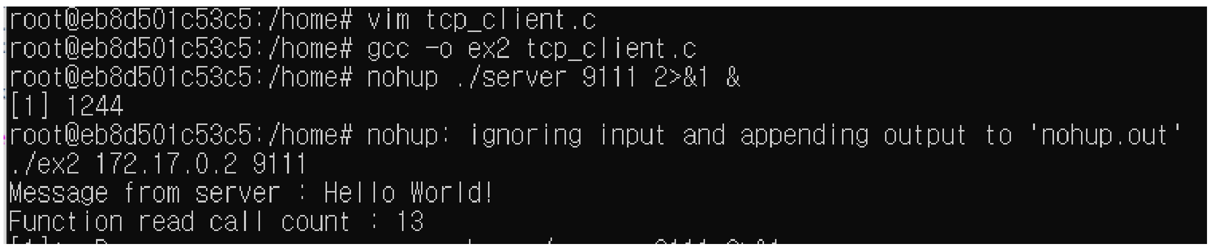


```
root@eb8d501c53c5:/home# nohup ./server 9111 2>&1 &
[1] 1168
root@eb8d501c53c5:/home# nohup: ignoring input and appending output to 'nohup.out'

root@eb8d501c53c5:/home# ./client 172.17.0.2 9111
Message from server : Hello World!
[1]+  Done                    nohup ./server 9111 2>&1
root@eb8d501c53c5:/home#
```

6. tcp_client.c

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

void error_handling(char *message);
```

```c
int main(int argc, char *argv[])
{
    int sock;
    struct sockaddr_in serv_addr;
    char message[30];
    int str_len = 0;

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

    while (read_len = read(sock, &message[idx++], 1))
    {
        if (read_len == -1)
            error_handling("read() error!");
        str_len += read_len;
    }
    printf("Message from server : %s\n", message);
    printf("Function read call count : %d\n", str_len);
    close(sock);
    return 0;
}

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



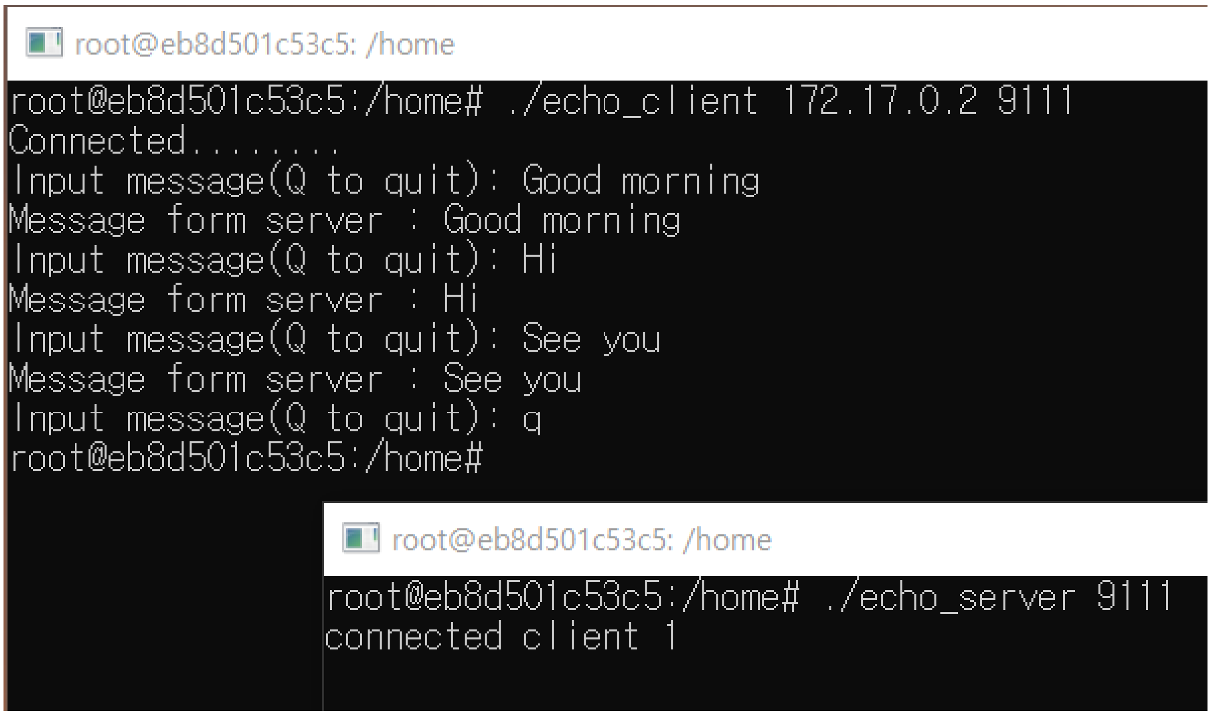
```
root@eb8d501c53c5:/home# vim tcp_client.c
root@eb8d501c53c5:/home# gcc -o ex2 tcp_client.c
root@eb8d501c53c5:/home# nohup ./server 9111 2>&1 &
[1] 1244
root@eb8d501c53c5:/home# nohup: ignoring input and appending output to 'nohup.out'
./ex2 172.17.0.2 9111
Message from server : Hello World!
Function read call count : 13
```

7-1. Echo_server.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 serv_sock;
  int clnt_sock;

  struct sockaddr_in serv_addr, clnt_addr;
  socklen_t clnt_addr_size;
  char message[BUF_SIZE];
  int str_len, i;
  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_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, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error!");
  if(listen(serv_sock, 5)==-1)
    error_handling("listen() error!");
  clnt_addr_size = sizeof(clnt_addr);

  for(i=0; i<5; i++){
    clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
    if(clnt_sock==-1)
      error_handling("accept() error");
    else
      printf("connected client %d \n", i+1);
    while((str_len = read(clnt_sock, message, BUF_SIZE)) != 0)
      write(clnt_sock, message, str_len);
    close(clnt_sock);
  }
  close(serv_sock);
  return 0;
}



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

7-2. 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);
 }
```



```
root@eb8d501c53c5: /home
root@eb8d501c53c5:/home# ./echo_client 172.17.0.2 9111
Connected........
Input message(Q to quit): Good morning
Message form server : Good morning
Input message(Q to quit): Hi
Message form server : Hi
Input message(Q to quit): See you
Message form server : See you
Input message(Q to quit): q
root@eb8d501c53c5:/home#
```

```
root@eb8d501c53c5: /home
root@eb8d501c53c5:/home# ./echo_server 9111
connected client 1
```