
TCP 소켓 프로그래밍 1

컴퓨터 통신 실습
홍석준

1. TCP 소켓 프로그램 작성

TCP 소켓 프로그램 작성(서버 프로그램)

- 아래와 같이 TCP소켓 echo 서버 프로그램 작성
- tcp_server1.c 파일로 작성(gedit tcp_server1.c명령 사용)

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

#define BUFSIZE 1024

void error_handling(char * message);

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

    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    int clnt_addr_size;

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

(계속)

1. TCP 소켓 프로그램 작성

TCP 소켓 프로그램 작성(서버 프로그램)

□ 아래와 같이 TCP소켓 echo 서버 프로그램 작성

```
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(4000);

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");

while((str_len = read(clnt_sock, message, BUFSIZE)) != 0) {
    write(clnt_sock, message, str_len);
    write(1, message, str_len);
}
close(clnt_sock);

return 0;
}

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

(끝)

1. TCP 소켓 프로그램 작성

TCP 소켓 프로그램 작성(클라이언트 프로그램)

□ 아래와 같이 TCP소켓 echo 클라이언트 프로그램 작성

□ tcp_client1.c 파일로 작성

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

#define BUFSIZE 1024

void error_handling(char * message);

int main(int argc, char ** argv)
{
    int sock;
    char message[BUFSIZE];
    int str_len;
    struct sockaddr_in serv_addr;

    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("127.0.0.1");
    serv_addr.sin_port=htons(4000);
```

(계속)

1. TCP 소켓 프로그램 작성

TCP 소켓 프로그램 작성(클라이언트 프로그램)

❑ 아래와 같이 TCP소켓 echo 클라이언트 프로그램 작성

❑ tcp_client1.c 파일로 작성

```
if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
    error_handling("connect() error");

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

    if(!strcmp(message, "q\n")) break;
    write(sock, message, strlen(message));

    str_len=read(sock, message, BUFSIZE -1);
    message[str_len]=0;
    printf("Message from server :%s \n", message);
}

close(sock);

return 0;
}

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

2. TCP 소켓 프로그램 컴파일

TCP 소켓 프로그램 컴파일

- ❑ gcc -o tcp_server1 tcp_server1.c와
- ❑ gcc -o tcp_client1 tcp_client1.c 명령어로 tcp프로그램 컴파일
- ❑ 실행 파일 생성 확인

```
sjhong@ubuntu: ~/program
sjhong@ubuntu:~/program$ gcc -o tcp_server1 tcp_server1.c
sjhong@ubuntu:~/program$ gcc -o tcp_client1 tcp_client1.c
sjhong@ubuntu:~/program$ ls
tcp_client1  tcp_server.c  udp_echo_client.c  udp_test2
tcp_client1.c  techo_client.c  udp_echo_server  udp_test2.c
tcp_server1  techo_server.c  udp_echo_server.c  udp_test.c
tcp_server1.c  udp_echo_client  udp_test
sjhong@ubuntu:~/program$
```

3. 소켓 프로그램(TCP) 실행

TCP 소켓 프로그램 실행

- 2개의 프로그램을 각각 터미널에 띄워서 아래와 같이 동작하는지 확인
- TCP 서버를 먼저 실행해야함

```
sjhong@ubuntu: ~/program
sjhong@ubuntu:~/program$ ./tcp_server1
asdf
asd
c
e
█
```

```
sjhong@ubuntu: ~/program
sjhong@ubuntu:~/program$ ./tcp_client1
Input message (q to quit) : asdf
Message from server :asdf

Input message (q to quit) : asd
Message from server :asd

Input message (q to quit) : c
Message from server :c

Input message (q to quit) : e
Message from server :e

Input message (q to quit) : █
```

4. 커널 메시지 출력 확인

커널 메시지 출력 확인

- 위의 TCP프로그램 실행후 dmesg로 커널 메시지가 출력되는지 확인하기

```
sjhong@ubuntu: ~  
[ 1871.047467] net_tx_action  
[ 1871.432324] net_rx_action ^^  
[ 1873.448693] net_rx_action ^^  
[ 1875.465070] net_rx_action ^^  
[ 1876.073626] tcp_sendmsg_locked  
[ 1876.073634] tcp_push  
[ 1876.073634] tcp_write_xmit  
[ 1876.073635] tcp_transmit_skb  
[ 1876.073637] ip_queue_xmit  
[ 1876.073639] __ip_local_out  
[ 1876.073639] ip_output  
[ 1876.073639] ip_finish_output  
[ 1876.073640] ip_finish_output2  
[ 1876.073641] __dev_queue_xmit  
[ 1876.073645] net_rx_action ^^  
[ 1876.073646] process_backlog  
[ 1876.073646] __netif_receive_skb_core  
[ 1876.073648] ip_rcv  
[ 1876.073649] ip_rcv_finish  
[ 1876.073649] ip_local_deliver  
[ 1876.073649] ip_local_deliver_finish  
[ 1876.073650] tcp_v4_rcv  
[ 1876.073653] tcp_v4_do_rcv  
[ 1876.073653] tcp_rcv_established
```

Thank you for your attention !!
