Computer Network Socket Programming

21800603 이현승/22000546 이예은

1.

Code

#client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define BUFSIZE 1024

void error_handling(char *message);

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

	char MSG1[] = "Hello\nWorld";
	char MSG2[] = "HI\nhi";
	char MSG3[] = "HH\njj";

	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(1) {
		fputs("Enter a message to send(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("A message from the server : %s \n", message);
        }
        */

        int len_m1 = strlen(MSG1);
        write(sock, &len_m1, sizeof(len_m1));
        write(sock, MSG1, strlen(MSG1));

        int len_m2 = strlen(MSG2);
        write(sock, &len_m2, sizeof(len_m2));
        write(sock, MSG2, strlen(MSG2));

        int len_m3 = strlen(MSG3);
        write(sock, &len_m3, sizeof(len_m3));
        write(sock, MSG3, strlen(MSG3));

        for(i = 0; i < 3; i ++) {
            str_len = read(sock, message, BUFSIZE-1);
            message[str_len] = '\0';
            printf("Message %d : \n%s\n", i+1, message);
        }

        close(sock);
        return 0;
}


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

#server.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.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, num = 0;

        struct sockaddr_in serv_addr;
        struct sockaddr_in clnt_addr;
        int clnt_addr_size;
```

```
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");

sleep(5);
clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);

if(clnt_sock == -1)
	error_handling("accept() error");

while(1) {
	clnt_addr_size = sizeof(clnt_addr);

	sleep(1);
	int m_len;
	read(clnt_sock, &m_len, sizeof(m_len));
	str_len = read(clnt_sock, message, m_len);
	if(str_len == 0) break;
	printf("The number of messages: %d \n", num++);
	write(clnt_sock, message, str_len);
	write(1, message, str_len);
}

/*
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);
close(serv_sock);
return 0;
```

```
}

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

결과 화면

```
[s21800603@localhost ~]$ ./client 203.252.112.26 60000
Message 1 :
Hello
World
Message 2 :
HI
hi
Message 3 :
HH
jj
```

메세지를 보내기 전 메세지의 길이를 보내 해당 길이만큼 읽도록 한다. 해당 길이를 기준으로 Message Boundary를 구분하여 메세지를 구분할 수 있도록 하였다.

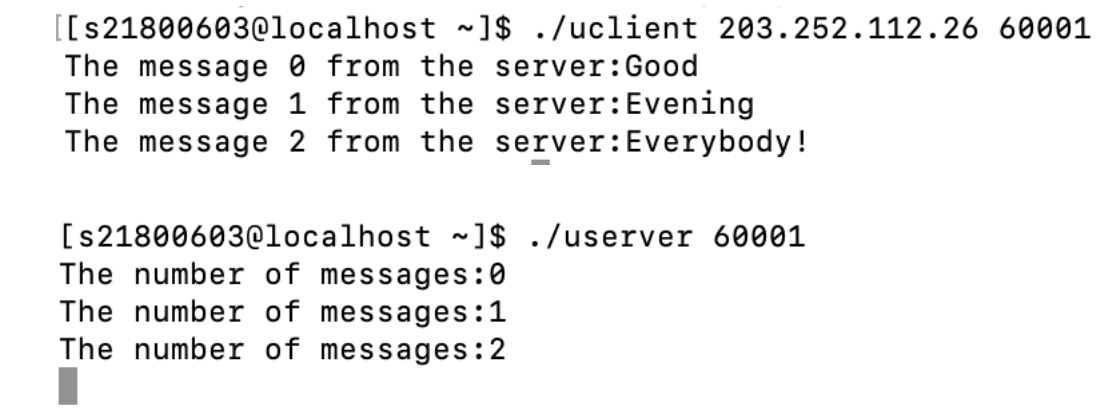
2.
Code
#uclient.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.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 sock,i;
   char message[BUF_SIZE];
   int str_len;
   socklen_t adr_sz;

   char MSG1[]="Good";
   char MSG2[]="Evening";
   char MSG3[]="Everybody!";

   struct sockaddr_in serv_adr, from_adr;
   if(argc!=3){
```

```
        printf("Usage : %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock=socket(PF_INET, SOCK_DGRAM, 0);
    if(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=inet_addr(argv[1]);
    serv_adr.sin_port=htons(atoi(argv[2]));
    sendto(sock,MSG1,strlen(MSG1),0,(struct sockaddr*)&serv_adr,sizeof(serv_adr));
    sendto(sock,MSG2,strlen(MSG2),0,(struct sockaddr*)&serv_adr,sizeof(serv_adr));
    sendto(sock,MSG3,strlen(MSG3),0,(struct sockaddr*)&serv_adr,sizeof(serv_adr));

    for(i=0;i<3;i++){
        adr_sz=sizeof(from_adr);
        str_len=recvfrom(sock,message,BUF_SIZE,0,(struct sockaddr*)&from_adr,&adr_sz);
        message[str_len]='\0';
        printf("The message %d from the server:%s \n",i,message);
    }
    // while(1)
    // {
    //  fputs("Insert message(q to quit): ", stdout);
    //  fgets(message, sizeof(message), stdin);
    //  if(!strcmp(message,"q\n") || !strcmp(message,"Q\n"))
    //      break;

    //  sendto(sock, message, strlen(message), 0,
    //          (struct sockaddr*)&serv_adr, sizeof(serv_adr));
    //  adr_sz=sizeof(from_adr);
    //  str_len=recvfrom(sock, message, BUF_SIZE, 0,
    //          (struct sockaddr*)&from_adr, &adr_sz);

    //  message[str_len]=0;
    //  printf("Message from server: %s", message);
    // }
    close(sock);
    return 0;
}

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

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

```c
#include <sys/socket.h>

#define BUF_SIZE 30
void error_handling(char *message);

int main(int argc, char *argv[])
{
    int serv_sock;
    char message[BUF_SIZE];
    int str_len,num=0;
    socklen_t 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_DGRAM, 0);
    if(serv_sock==-1)
        error_handling("UDP socket creation 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))==-1)
        error_handling("bind() error");

    sleep(5);
    while(1)
    {
        clnt_adr_sz=sizeof(clnt_adr);
        sleep(1);
        str_len=recvfrom(serv_sock, message, BUF_SIZE, 0,
                         (struct sockaddr*)&clnt_adr, &clnt_adr_sz);
        printf("The number of messages:%d\n",num++);
        sendto(serv_sock, message, str_len, 0,
                         (struct sockaddr*)&clnt_adr, clnt_adr_sz);
    }
    close(serv_sock);
    return 0;
}

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

(a)큰 버퍼를 이용하여 수신한 경우 아무런 변화 없이 받은 순서대로 출력된다.



(b)보내진 메세지보다 작은 버퍼 사이즈로 메세지를 읽는 경우엔 아래 사진과 같이 버퍼 사이즈만큼만 출력되고 그 이후의 메세지는 출력되지 않는다.