<serv>

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

typedef struct sockaddr_in si;
typedef struct sockaddr *  sap;
typedef struct p
{
int data;
char name[20];
}p;

#define BUF_SIZE          32

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

void read_childproc(int sig)
{
   pid_t pid;
   int status;
   pid = waitpid(-1,&status, WNOHANG);
   printf("Romoved proc id: %d\n", pid);
}

int main(int argc,char **argv)
{
   int serv_sock, clnt_sock;
   si serv_addr, clnt_addr;
   pid_t pid;
   struct sigaction act;
   socklen_t addr_size;
   int str_len, state;
   char buf[BUF_SIZE] = {0};

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

```
    act.sa_handler = read_childproc;
    sigemptyset(&act.sa_mask);
    act.sa_flags = 0;
    state = sigaction(SIGCHLD, &act, 0);

    if(serv_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 = 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() error");

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

    for(;;)
    {
        addr_size = sizeof(clnt_addr);
        clnt_sock = accept(serv_sock, (sap)&clnt_addr, &addr_size);
        if(clnt_sock == -1)
           continue;
        else
           puts("New Client Connected ...");
        pid = fork();
        if(pid == -1)
        {
           close(clnt_sock);
           continue;
        }
        if(pid == 0)
        {
           close(serv_sock);

           while((str_len = read(clnt_sock, buf, BUF_SIZE)) != 0)//들어온걸 읽고 보내주는것이 echo
//read 는 blocking 이라 뭔신호가 올때까지 계속 기다리고 있으니 끝나지가않는다.
                write(clnt_sock, buf, str_len);

           close(clnt_sock);
           puts("Client Disconnected ...");
           return 0;
        }
        else
           close(clnt_sock);
    }
    close(serv_sock);
    return 0;
}
```

<cleint>

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

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;


#define BUF_SIZE 32
typedef struct  p
{
int data;
} p;
void err_handler(char *msg){
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
}


void read_routine(int sock, char *buf){

    for(;;){
            int str_len = read(sock, buf, BUF_SIZE);
            if(str_len == 0)
                return;
            buf[str_len] = 0;
            printf("msg from server: %s",buf);
    }
}




void write_routine(int sock, char *buf){

    for(;;){
```

```
        fgets(buf,BUF_SIZE, stdin);

        if(!strcmp(buf,"q\n") || !strcmp(buf,"0\n")){
            shutdown(sock,SHUT_WR);
            return ;
        }
        write(sock,buf,strlen(buf));
    }
}

int main(int argc, char **argv){

    pid_t pid;
    int i, sock;
    si serv_addr;
    char buf[BUF_SIZE] = {0};

    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....");

    pid = fork();

    if(pid == 0)
        write_routine(sock, buf);
    else
        read_routine(sock, buf);

    close(sock);
    return 0;
}
```

<소켓을 이용한 서버 클라이언트 서로 구조체를 보내기>

<client>

```
#include "common.h"

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

void read_proc(int sock, d *buf)
{
for(;;)
{
int len = read(sock, buf, BUF_SIZE);

if(!len)
return;

printf("msg from serv: %d, %f\n", buf->data, buf->fdata);
}
}

void write_proc(int sock, d *buf)
{
char msg[32] = {0};

for(;;)
{
fgets(msg, BUF_SIZE, stdin);

if(!strcmp(msg, "q\n") || !strcmp(msg, "Q\n"))
{
shutdown(sock, SHUT_WR);
return;
}

buf->data = 3;
buf->fdata = 7.7;

write(sock, buf, sizeof(d));
}
}

int main(int argc, char **argv)
{
pid_t pid;
```

```
int i, sock;
si serv_addr;
d struct_data;
char buf[BUF_SIZE] = {0};

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, (sp)&serv_addr, sizeof(serv_addr)) == -1)
err_handler("connect() error");
else
puts("Connected!\n");

pid = fork();

if(!pid)
write_proc(sock, (d *)&struct_data);
else
read_proc(sock, (d *)&struct_data);

close(sock);

return 0;
}



#include "common.h"
#include <signal.h>
#include <sys/wait.h>

typedef struct sockaddr_in si;
typedef struct sockaddr * sp;

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

```
exit(1);
}
void read_cproc(int sig)
{
pid_t pid;
int status;
pid = waitpid(-1, &status, WNOHANG);
printf("Removed proc id: %d\n", pid);
}
int main(int argc, char **argv)
{
int serv_sock, clnt_sock, len, state;
char buf[BUF_SIZE] = {0};
si serv_addr, clnt_addr;
struct sigaction act;
socklen_t addr_size;
d struct_data;
pid_t pid;
if(argc != 2)
{
printf("use: %s <port>\n", argv[0]);
exit(1);
}
act.sa_handler = read_cproc;
sigemptyset(&act.sa_mask);
act.sa_flags = 0;
state = sigaction(SIGCHLD, &act, 0);
serv_sock = socket(PF_INET, SOCK_STREAM, 0);
if(serv_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 = htonl(INADDR_ANY);
serv_addr.sin_port = htons(atoi(argv[1]));
if(bind(serv_sock, (sp)&serv_addr, sizeof(serv_addr)) == -1)
err_handler("bind() error");
if(listen(serv_sock, 5) == -1)
err_handler("listen() error");
for(;;)
{
```

```
addr_size = sizeof(clnt_addr);
clnt_sock = accept(serv_sock, (sp)&clnt_addr, &addr_size);

if(clnt_sock == -1)
continue;
else
puts("New Client Connected!\n");

pid = fork();

if(pid == -1)
{
close(clnt_sock);
continue;
}

if(!pid)
{
close(serv_sock);

while((len = read(clnt_sock, (d *)&struct_data, BUF_SIZE)) != 0)
{
printf("struct.data = %d, struct.fdata = %f\n", struct_data.data, struct_data.fdata);
write(clnt_sock, (d *)&struct_data, len);
}

close(clnt_sock);
puts("Client Disconnected!\n");
return 0;
}
else
close(clnt_sock);
}
close(serv_sock);

return 0;
}
```