

Assignment5

2016116783 김성록

Example1

pipe1.c

```
#include <stdio.h>
#include <unistd.h>
#define BUF_SIZE 30

int main(int argc, char *argv[]){
    int fds[2];
    char str[] = "Who are you?";
    char buf[BUF_SIZE];
    pid_t pid;

    pipe(fds);
    pid = fork();
    if(pid==0){
        write(fds[1], str, sizeof(str));
    }
    else{
        read(fds[0], buf, BUF_SIZE);
        puts(buf);
    }
    return 0;
}
```

```
root@eb8d501c53c5:/home/assignment5# ./pipe1
Who are you?
root@eb8d501c53c5:/home/assignment5#
```

Example2

pipe2

```

#include <stdio.h>
#include <unistd.h>
#define BUF_SIZE 30

int main(int argc, char *argv[]){
    int fds[2];
    char str1[] = "Who are you?";
    char str2[] = "Thank you for your message";
    char buf[BUF_SIZE];
    pid_t pid;

    pipe(fds);
    pid = fork();
    if(pid==0){
        write(fds[1], str1, sizeof(str1));
        sleep(2);
        read(fds[0], buf, BUF_SIZE);
        printf("Child proc output: %s\n",buf);
    }
    else{
        read(fds[0], buf, BUF_SIZE);
        printf("Parent proc output: %s\n",buf);
        write(fds[1], str2, sizeof(str2));
        sleep(3);
    }
    return 0;
}

```

```

root@eb8d501c53c5:/home/assignment5# ./pipe2
Parent proc output: Who are you?
Child proc output: Thank you for your message
root@eb8d501c53c5:/home/assignment5#

```

Example3

pipe3

```

#include <stdio.h>
#include <unistd.h>
#define BUF_SIZE 30

int main(int argc, char *argv[]){
    int fds1[2], fds2[2];

```

```

char str1[] = "Who are you?";
char str2[] = "Thank you for your message";
char buf[BUF_SIZE];
pid_t pid;

pipe(fds1), pipe(fds2);
pid = fork();
if(pid==0){
    write(fds1[1], str1, sizeof(str1));
    read(fds2[0], buf, BUF_SIZE);
    printf("Child proc output: %s\n",buf);
}
else{
    read(fds1[0], buf, BUF_SIZE);
    printf("Parent proc output: %s\n",buf);
    write(fds2[1], str2, sizeof(str2));
    sleep(3);
}
return 0;
}

```

```

root@eb8d501c53c5:/home/assignment5# ./pipe3
Parent proc output: Who are you?
Child proc output: Thank you for your message
root@eb8d501c53c5:/home/assignment5#

```

Example4

eacho_storeserv.c

```

#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>

#define BUF_SIZE 100
void error_handling(char *message);
void read_childproc(int sig);

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

```

```

struct sockaddr_in serv_adr, clnt_adr;
int fds[2];

pid_t pid;
struct sigaction act;
socklen_t adr_sz;
int str_len, state;
char buf[BUF_SIZE];

if(argc!=2){
    printf("Usage: %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);

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

pipe(fds);
pid=fork();
if(pid==0){
    FILE * fp = fopen("echomsg.txt", "wt");
    char msgbuf[BUF_SIZE];
    int i, len;
    for(i=0; i<10; i++){
        len=read(fds[0], msgbuf, BUF_SIZE);
        fwrite((void*)msgbuf, 1, len, fp);
    }
    fclose(fp);
    return 0;
}

while(1){
    adr_sz = sizeof(clnt_adr);
    clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_adr, &adr_sz);
    if(clnt_sock== -1) continue;
    else puts("New client connected...");
    pid=fork();
    if(pid==0){
        close(serv_sock);
        while((str_len==read(clnt_sock, buf, BUF_SIZE))!=0){
            write(clnt_sock, buf, str_len);
            write(fds[1], buf, str_len);
        }
        close(clnt_sock);
        puts("client disconnected...");
    }
}

```

```

        return 0;
    }
    else close(clnt_sock);
}
close(serv_sock);
return 0;
}

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

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

```

The image shows two terminal windows side-by-side, illustrating a client-server interaction. The left window is the client's terminal, and the right window is the server's terminal.

Left Terminal (Client):

```

root@eb8d501c53c5: /home/assignment5
root@eb8d501c53c5: /home/assignment5# ./server 27.0.0.1 1234
New client connected...
New client connected...
client disconnected...
removed proc id : 184
client disconnected...
removed proc id : 181

```

Right Terminal (Server):

```

root@eb8d501c53c5: /home/assignment5
One
Message from server : One
Three
Message from server : Three
Five
Message from server : Five
Q
root@eb8d501c53c5: /home/assignment5#
Two
Message from server : Two
Four
Message from server : Four
Six
Message from server : Six
Q
root@eb8d501c53c5: /home/assignment5#

```

```
root@eb8d501c53c5:/home/assignment5# cat echomsg.txt
One
Two
Three
Four
Five
Six
```

Example 5

select.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/select.h>
#define BUF_SIZE 30
int main(int argc, char *argv[]){
    fd_set reads, temps;
    int result, str_len;
    char buf[BUF_SIZE];
    struct timeval timeout;
    FD_ZERO(&reads);
    FD_SET(0, &reads);

    timeout.tv_sec = 5;
    timeout.tv_usec = 5000;
    while(1){
        temps = reads;
        timeout.tv_sec = 5;
        timeout.tv_usec = 0;
        result = select(1, &temps, 0, 0, &timeout);
        if(result==-1){
            puts("select() error!");
            break;
        }
        else if(result==0){
            puts("time out!");
        }
        else{
            if(FD_ISSET(0, &temps)){
                str_len = read(0, buf, Buf_SIZE);
                buf[str_len] = 0;
                printf("message from console:%s",buf);
            }
        }
    }
}
```

```
    return 0;
}
```

```
root@eb8d501c53c5:/home/assignment5# ./select
time out!
time out!
time out!
time out!
time out!
time out!
time out!
time out!
Hi
message from console:Hi
Hello
time out!
Hello~
message from console:Hello~
Good
time out!
Bye-
message from console:Good Bye-
^C
```

Example 6

Echo_selectserv.c

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

#define BUF_SIZE 100
void error_handling(char *message);
int main(int argc, char *argv[]){
    int serv_sock, clnt_sock;
    struct sockaddr_in serv_adr, clnt_adr;
    struct timeval timeout;
    fd_set reads, cpy_reads;

    socklen_t adr_sz;
```

```

int fd_max, str_len, fd_num, i;
char buf[BUF_SIZE];
if(argc!=2){
    printf("Usage : %s <port>\n",argv[0]);
    exit(1);
}

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

FD_ZERO(&reads);
FD_SET(serv_sock, &reads);
fd_max = serv_sock;
while(1){
    cpy_reads = reads;
    timeout.tv_sec = 5;
    timeout.tv_usec = 5000;
    if((fd_num = select(fd_max+1, &cpy_reads, 0,0, &timeout))== -1)
        break;
    if(fd_num==0)
        continue;
    for(i=0;i<fd_max+1;i++){
        if(FD_ISSET(i, &cpy_reads)){
            if(i==serv_sock){
                adr_sz = sizeof(clnt_adr);
                clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_adr, &adr_sz);
                FD_SET(clnt_sock, &reads);
                if(fd_max<clnt_sock)
                    fd_max = clnt_sock;
                printf("connected client: %d\n",clnt_sock);
            }
            else{
                str_len = read(i, buf, BUF_SIZE);
                if(str_len==0){
                    FD_CLR(i, &reads);
                    close(i);
                    printf("closed client: %d\n",i);
                }
                else{
                    write(i, buf, str_len);
                }
            }
        }
    }
}
close(serv_sock);
return 0;
}

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

```



```

    exit(1);
}

```

```

root@eb8d501c53c5: /home/assignment5
root@eb8d501c53c5:/home/assignment5# ./echo_selectserv 1234
connected client: 4
connected client: 5
closed client: 4
closed client: 5
^C
root@eb8d501c53c5:/home/assignment5# clear
root@eb8d501c53c5:/home/assignment5# ./echo_client 127.0.0.1 1234
Connected.....
Input message(Q to quit): Hi~
Message form server : Hi~
Input message(Q to quit): Goot bye
Message form server : Goot bye
Input message(Q to quit): Q
root@eb8d501c53c5:/home/assignment5#

root@eb8d501c53c5: /home/assignment5
echo_storeserv echo_client echo_mpclient echo_selectserv
echo_storeserv.c echo_client.c echo_mpclient.c echo_selectserv.c
root@eb8d501c53c5:/home/assignment5# clear
root@eb8d501c53c5:/home/assignment5# ./echo_client 127.0.0.1 1234
Connected.....
Input message(Q to quit): Nice to meet you~
Message form server : Nice to meet you~
Input message(Q to quit): Bye~
Message form server : Bye~
Input message(Q to quit): Q
root@eb8d501c53c5:/home/assignment5#

```

Problem 1

항상 개행(\n)을 잘 살피자

source.c

```

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

#define BUF_SIZE 100

void error_handling(char *message);
int main(int argc, char *argv[]){

```

```

int serv_sock, clnt_sock;
struct sockaddr_in serv_adr, clnt_adr;
struct timeval timeout;
fd_set reads, cpy_reads;

int fd[10];
int fd_size=0;

socklen_t adr_sz;
int fd_max, str_len, fd_num, i;
char buf[BUF_SIZE];

char welcome[10] = "Welcome~\n";
char client_num[60];
char CLIENT_HELLO = '0';
char CLIENT_CHAT = '1';
char CLIENT_BYE = '2';
char client_id[3];
char temp[BUF_SIZE];

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

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

FD_ZERO(&reads);
FD_SET(serv_sock, &reads);
fd_max = serv_sock;
while(1){
    cpy_reads = reads;
    timeout.tv_sec = 5;
    timeout.tv_usec = 5000;
    if((fd_num = select(fd_max+1, &cpy_reads, 0,0, &timeout))==-1)
        break;
    if(fd_num==0)
        continue;
    for(i=0;i<fd_max+1;i++){
        if(FD_ISSET(i, &cpy_reads)){
            if(i==serv_sock){
                adr_sz = sizeof(clnt_adr);
                clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_adr, &adr_sz);
                FD_SET(clnt_sock, &reads);
                if(fd_max<clnt_sock)
                    fd_max = clnt_sock;
                fd[fd_size++]=clnt_sock;
                printf("connected client: %d\n",clnt_sock);
            }
        }
    }
}

```

```

        // send welcome message
        write(clnt_sock, welcome, strlen(welcome));
        sleep(1);

        // send the number of clients
        sprintf(buf, "The number of Clients is %d now.\n", fd_size);
        write(clnt_sock, buf, strlen(buf));

        sprintf(client_id, "%02d", clnt_sock);
        buf[0] = CLIENT_HELLO;
        strcpy(buf+1, client_id);
        strcpy(buf+3, welcome);

        for(int j=0 ;j<fd_size-1;j++){
            write(fd[j], buf, strlen(buf));
        }
    }
    else{
        str_len = read(i, buf, BUF_SIZE);
        buf[str_len]='\0';
        if(str_len==0){

            // remove i from fd[]
            for(int k=0;k<fd_size;k++){
                if(fd[k]==i){
                    for(int m=k;m<fd_size-1;m++){
                        fd[m]=fd[m+1];
                    }
                    break;
                }
            }

            FD_CLR(i, &reads);
            close(i);
            printf("closed client: %d\n", i);

            fd_size--;
            fd[fd_size]=-1;

            sprintf(client_id, "%02d", i);
            buf[0] = CLIENT_BYE;
            strcpy(buf+1, client_id);
            strcpy(buf+3, welcome);

            for(int j=0 ;j<fd_size;j++){
                write(fd[j], buf, strlen(buf));
            }
        }
        else{
            strcpy(temp, buf);
            sprintf(client_id, "%02d", i);
            buf[0] = CLIENT_CHAT;
            strcpy(buf+1, client_id);
            strcpy(buf+3, temp);

            for(int j=0 ;j<fd_size;j++){
                if (fd[j]!=i){
                    write(fd[j], buf, strlen(buf));
                }
            }
        }
    }
}

```

```

    }
    }
    }
    }
    }
    close(serv_sock);
    return 0;
}
void error_handling(char *buf){
    fputs(buf, 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 100
#define CLIENT_HELLO 0
#define CLIENT_CHAT 1
#define CLIENT_BYE 2

void error_handling(char *message);
void read_routine(int sock, char *buf);
void write_routine(int sock, char *buf);

int main(int argc, char *argv[]){
    int sock;
    pid_t pid;
    char buf[BUF_SIZE];
    char buf2[BUF_SIZE];
    struct sockaddr_in serv_adr;

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

    sock = 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]));
}

```

```

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

    // read socket form : [message] # Welcome~
    int str_len;

    str_len = read(sock, buf, BUF_SIZE-1);
    buf[str_len]=0;
    printf("Server : %s", buf);

    // read socket form : [message] # The number of Clients is 00 now.
    str_len = read(sock, buf, BUF_SIZE-1);
    buf[str_len]=0;
    printf("Server : %s", buf);

    pid = fork();
    if(pid==0)
        write_routine(sock, buf);
    else
        read_routine(sock, buf);
    close(sock);
    return 0;
}

void read_routine(int sock, char *buf){
    while(1){
        // read socket form : [state, clientID, message]
        int str_len = read(sock, buf, BUF_SIZE-1);
        if(str_len==0)
            return ;
        buf[str_len]=0;

        int state = buf[0]-'0';
        int clientID = (buf[1]-'0')*10+(buf[2]-'0');
        char *message = buf+3;

        if(state==CLIENT_HELLO){
            printf("Client %d has joined this chatting room\n", clientID);
        }
        else if(state==CLIENT_CHAT){
            printf("Client %d : %s", clientID, message);
        }
        else if(state==CLIENT_BYE){
            printf("Client %d has left this chatting room\n", clientID);
        }
    }
}

void write_routine(int sock, char *buf){
    while(1){
        fgets(buf, BUF_SIZE, stdin);
        if(!strcmp(buf, "q\n") || !strcmp(buf, "Q\n")){
            shutdown(sock, SHUT_WR);
            return ;
        }
        write(sock, buf, strlen(buf));
    }
}

```

```

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

```

The image displays four terminal windows from a Linux environment, showing the execution of a chat server and three clients (Client4, Client5, and Client6). The windows are arranged in a 2x2 grid.

- Top-Left Window (Server):** Shows the server's log of client connections and disconnections.


```

root@eb8d501c53c5: /home/assignment5# ./source 9111
connected client: 4
connected client: 5
connected client: 6
closed client: 6
closed client: 5
closed client: 4

```
- Top-Right Window (Client4):** Shows Client4's interactions with the server.


```

root@eb8d501c53c5: /home/assignment5# ./client 127.0.0.1 9111
Server : Welcome~
Server : The number of Clients is 1 now.
Client 5 has joined this chatting room
Hello~
Client 5 : Hi~
Client 6 has joined this chatting room
Client 6 : Hello everyone!
Nice to meet you
Client 5 : See you~
Client 6 : Bye~
Client 6 has left this chatting room
bye bye!
Client 5 has left this chatting room
q
root@eb8d501c53c5: /home/assignment5#

```
- Bottom-Left Window (Client5):** Shows Client5's interactions with the server.


```

root@eb8d501c53c5: /home/assignment5# ./client 127.0.0.1 9111
Server : Welcome~
Server : The number of Clients is 2 now.
Client 4 : Hello~
Hi~
Client 6 has joined this chatting room
Client 6 : Hello everyone!
Client 4 : Nice to meet you
See you~
Client 6 : Bye~
Client 6 has left this chatting room
Client 4 : bye bye!
q
root@eb8d501c53c5: /home/assignment5#

```
- Bottom-Right Window (Client6):** Shows Client6's interactions with the server.


```

root@eb8d501c53c5: /home/assignment5# ./client 127.0.0.1 9111
Server : Welcome~
Server : The number of Clients is 3 now.
Hello everyone!
Client 4 : Nice to meet you
Client 5 : See you~
Bye~
q
root@eb8d501c53c5: /home/assignment5#

```