시스템 프로그래밍 실습

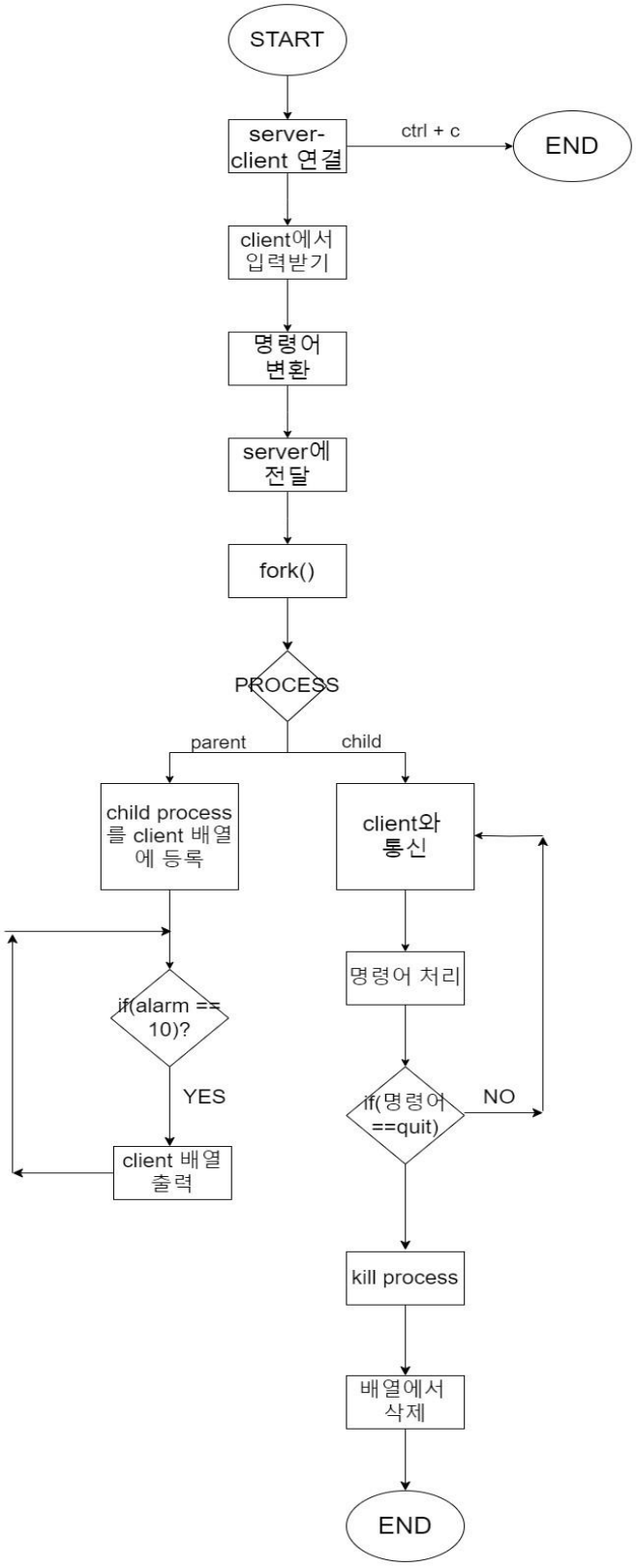
[Assignment2-3]

Class	:	[A]
Professor	:	[김태석 교수님]
Student ID	:	[2019202032]
Name	:	[이상현]

Introduction

해당 과제는 기 구현했던 Assignment1_3 에 socket 프로그래밍을 추가하는 과제이며, 이때 server 는 client 로부터 다중 접속을 허용한다. Server 는 client 와 연결이 될 때마다 연결된 client socket 의 정보를 출력하고, 10 초마다 현재 실행 중인 child process 들의 정보를 출력한다. 이때 Client 가 접속될 때마다 10 초를 다시 카운팅하도록 한다. Client 는 quit 나 ctrl+c 를 통해서 종료할 수 있으며, client 가 종료되면 signal 을 활용하여 해당 child process 를 server 에서 종료하고, child process 를 관리하는 배열에서 삭제한다.

Flow chart


START
server-client 연결
ctrl + c
END
client에서 입력받기
명령어 변환
server에 전달
fork()
PROCESS
parent
child
child process를 client 배열에 등록
client와 통신
if(alarm == 10)?
YES
client 배열 출력
명령어 처리
if(명령어 ==quit)
NO
kill process
배열에서 삭제
END

Pseudo code

<server>

```
typedef struct {
    int pid;
    int port;
    time_t start_time;
} ClientInfo;


ClientInfo clients[MAX_CLIENTS];
int clients_cnt = 0;


int cmd_process(char*buff, char*result_buff){
if(!strcmp(command, "QUIT")){          //case of QUIT
    }
    if(!strcmp(command, "NLST")){          //case of NLST
        DIR *dirp;
        struct dirent *dir;
        struct stat file;
         ////////////////// if the argument is file ////////////////////////
        if(access(directory, F_OK) == -1){
            correct_argument(directory);//check if i can open the directory
        }


        /*                    case of no read permission                    */
```

```
char per[100];
GetPermission(file, directory, per);
if(per[0] == '-')
{
	write(1, "cannot access₩n₩0", 16);
	exit(0);
}

char filetype = GetFiletype(file, directory); //get filetype
/*                              If directory is file                        */
if(filetype == '-' ){
	if(!lflag)
		exit(0);
	write(1, "NLST -l₩n", 9);
	option_l(directory, result_buff);
	chdir(current_directory);
	return 1;
}

char**filenames = (char**)malloc(sizeof(char*)*BUF_SIZE);
for(int i = 0; i<BUF_SIZE; i++){
	filenames[i] = (char*)malloc(sizeof(char)*BUF_SIZE);
}
int filecnt = 0;
```

```
/*                    get the files of directory              */
dirp = opendir(directory);
while((dir=readdir(dirp))!= NULL){
    if(aflag)                                   //if a option
        strcpy(filenames[filecnt++], dir->d_name);
    else{
        if(dir->d_name[0] != '.')     //if not a option
            strcpy(filenames[filecnt++], dir->d_name);
    }
}
closedir(dirp);

/*              save current working directory and change directory                    */
chdir(directory);
////////////////////////////////////////////////////////////////////////////////////////

char**temp_filenames = (char**)malloc(sizeof(char*)*BUF_SIZE);
for(int i = 0; i<BUF_SIZE; i++){
    temp_filenames[i] = (char*)malloc(sizeof(char)*BUF_SIZE);
    strcpy(temp_filenames[i], filenames[i]);
}

ArrangeFilenames(filenames, temp_filenames, 0, filecnt-1);//                    arrange
files
```

```
/*                              start nlst -al                              */
if(aflag && lflag){
        write(1, "NLST -al₩n", 10);
        for(int i = 0; i<filecnt; i++){
                option_l(filenames[i], result_buff);
        }
        chdir(current_directory);
        return 1;
}
//////////                    finish nlst -al                    /////////


/*                              start nlst -a                               */
if(aflag){
        write(1, "NLST -a₩n", 9);
        int c_num = 0;
        for(int i= 0; i<filecnt; i++){
                char filetype = GetFiletype(file, filenames[i]);
                if(filetype == 'd'){            //file type d
                        strcat(result_buff, filenames[i]);
                        strcat(result_buff, "/₩n");
                }
                else{
                        strcat(result_buff, filenames[i]);
```

```
                strcat(result_buff, "₩n");
            }
        }
        chdir(current_directory);
        return 1;
    }
//////////                finish nlst -a                //////////

/*                          start nlst -l                          */
if(lflag){
        write(1, "NLST -l₩n", 9);
        for(int i= 0; i<filecnt; i++){
            option_l(filenames[i], result_buff);      //get file information
        }
        chdir(current_directory);
        return 1;
    }
//////////                finish nlst -l                //////////

/*                          start nlst                          */
else{
        write(1, "NLST₩n", 6);
        for(int i =0; i<filecnt; i++){
            char filetype = GetFiletype(file, filenames[i]);
```

```
                if(filetype == 'd'){                    //filetype d
                    strcat(result_buff, filenames[i]);
                    strcat(result_buff, "/₩n");
                }
                else{
                    strcat(result_buff, filenames[i]);
                    strcat(result_buff, "₩n");
                }
            }
            chdir(current_directory);
            return 1;
        }
        //////////                    finish nlst                    /////////
        return 1;
    }
    return -1;
}
int client_info(struct sockaddr_in client_addr){
    write(1, "==========Client info==========", 32);
    write(STDOUT_FILENO, "₩n₩n", 3);
    /********************* client IP address ************************/
    write(1, "client IP : ", 13);
    char*client_IP = inet_ntoa(client_addr.sin_addr);
    write(STDOUT_FILENO, client_IP, strlen(client_IP));
```

```
    write(STDOUT_FILENO, "₩n₩n₩n", 4);
    ////////////////////////////////////////////////////////////////
char client_port[100];
    sprintf(client_port, "%d", client_addr.sin_port);
    write(1, "client port : ", 15);
    write(STDOUT_FILENO, client_port, strlen(client_port));
}

void process_command(int connfd) {

    char buffer[BUF_SIZE];
    ssize_t n;

    while ((n = read(connfd, buffer, BUF_SIZE - 1)) > 0) {
        buffer[n] = '₩0';
        printf("Received from client: %s₩n", buffer);

        // 클라이언트 명령 처리 로직 추가

        // 클라이언트로 응답 전송
        write(connfd, buffer, strlen(buffer));

        // "quit" 명령 처리
        if (strcmp(buffer, "quit") == 0) {
```

```
                break;
            }
        }

        close(connfd);
}

/***********************************case of QUIT client***********************************
*
*       pid_t pid;
*       int status;
*       while((pid = waitpid(-1, &status, WNOHANG)) > 0) {
*               for(int i = 0; i<clients_cnt; i++){
*                       if(clients[i].pid == pid){
*                               kill(clients[i].pid, SIGTERM);
*                               printf("Client(%d)'s Release₩n", pid);
*                               for(int j = i; j<clients_cnt-1; j++){
*                                       clients[j] = clients[j+1];
*                               }
*                               clients_cnt--;
*                               break;
*                       }
*               }
*       }
```

```
*****************************************************************************************/


void print_child_process() {
    printf("Current number of client: %d₩n", clients_cnt);
    printf("PID₩tPORT₩tTIME₩t₩n");
    for (int i = 0; i < clients_cnt; i++) {
        time_t current_time = time(NULL);
        int process_time = (int)(current_time - clients[i].start_time);
        printf("%d₩t%d₩t%d₩n", clients[i].pid, clients[i].port, process_time);
    }
}


void timer_handler(int sig) {
    print_child_process();
    alarm(10); // 10 초 후에 다시 알람 설정
}

void sigint_handler(int sig) {
```

```
    // 모든 클라이언트 연결 종료 및 자식 프로세스 종료
    for (int i = 0; i < clients_cnt; i++) {
        kill(clients[i].pid, SIGTERM);
    }
    exit(0);
}

int main(int argc, char*argv[]) {
    char buff[BUF_SIZE];
    int n;
    struct sockaddr_in server_addr, client_addr;
    int server_fd, client_fd;
    int len;
    int port;



    pid_t pid;

    /******************** prepare server socket and connect with client socket **********************/
    server_fd = socket(PF_INET, SOCK_STREAM, 0);
    int opt = 1;
    setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
```

```
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);//set address
    server_addr.sin_port = htons(atoi(argv[1]));//set port


    if(bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0){   //bind socket
        printf("Server: Can't bind local address₩n");
        return 0;
    }

    listen(server_fd, 5);     //listen from client

/*****************************************************************************************
**/

    /************************** register signal handling   ******************************/
    signal(SIGALRM, timer_handler);
    signal(SIGINT, sigint_handler);
    alarm(10); // set 10 seconds alarm
    /*******************************************************************************/

    int num = 0;
```

```
    while (1) {
        pid_t pid;
        len = sizeof(client_addr);
        client_fd = accept(server_fd, (struct sockaddr *)&client_addr, &len);     //connect

        if ((pid = fork()) == 0) { // chile process

            close(server_fd); // close server socket which is used in parent process

            if(client_info(client_addr) < 0)
                write(STDERR_FILENO, "client_info() err!!₩n", 21);

            /***************************** communicate between sockets ********************************/
            while(1){
                n = read(client_fd, buff, BUF_SIZE);            //read string from client

                char result_buff[BUF_SIZE];
                cmd_process(buff, result_buff);
                write(client_fd, result_buff, strlen(result_buff));
            }

/*****************************************************************************************************
*/
```

```
            close(client_fd);
            exit(0); // terminate chile process
        } else if (pid > 0) { // parent process
            clients[clients_cnt].pid = pid;
            clients[clients_cnt].port = client_addr.sin_port;
            clients[clients_cnt].start_time = time(NULL);
            clients_cnt++;
            close(client_fd); // close client socket which is used in child process
        } else { // fork failed
            perror("fork");
            exit(1);
        }
    }

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

<client>

```
int conv_cmd(char*buff, char*cmd_buff){
    memset(cmd_buff, 0, sizeof(cmd_buff));

    char command[30];
```

```
int i = 0;
/****************seperate command from buffer***************/
for(i = 0; i<strlen(buff); i++){
        if(buff[i] != ' ' && buff[i] != '₩0' && buff[i] != '₩n')
        {
                command[i] = buff[i];
                command[i+1] = '₩0';
        }
        else{
                break;
        }
}
/*********************************************************/

/************************convert FTP command*********************/
Convert_FTP command()l
/*************************************************************/

/*************** strcat option directory   *****************/
int c = strlen(cmd_buff);
for(i; i<strlen(buff); i++){
        cmd_buff[c++] = buff[i];
        cmd_buff[c] = '₩0';
}
```

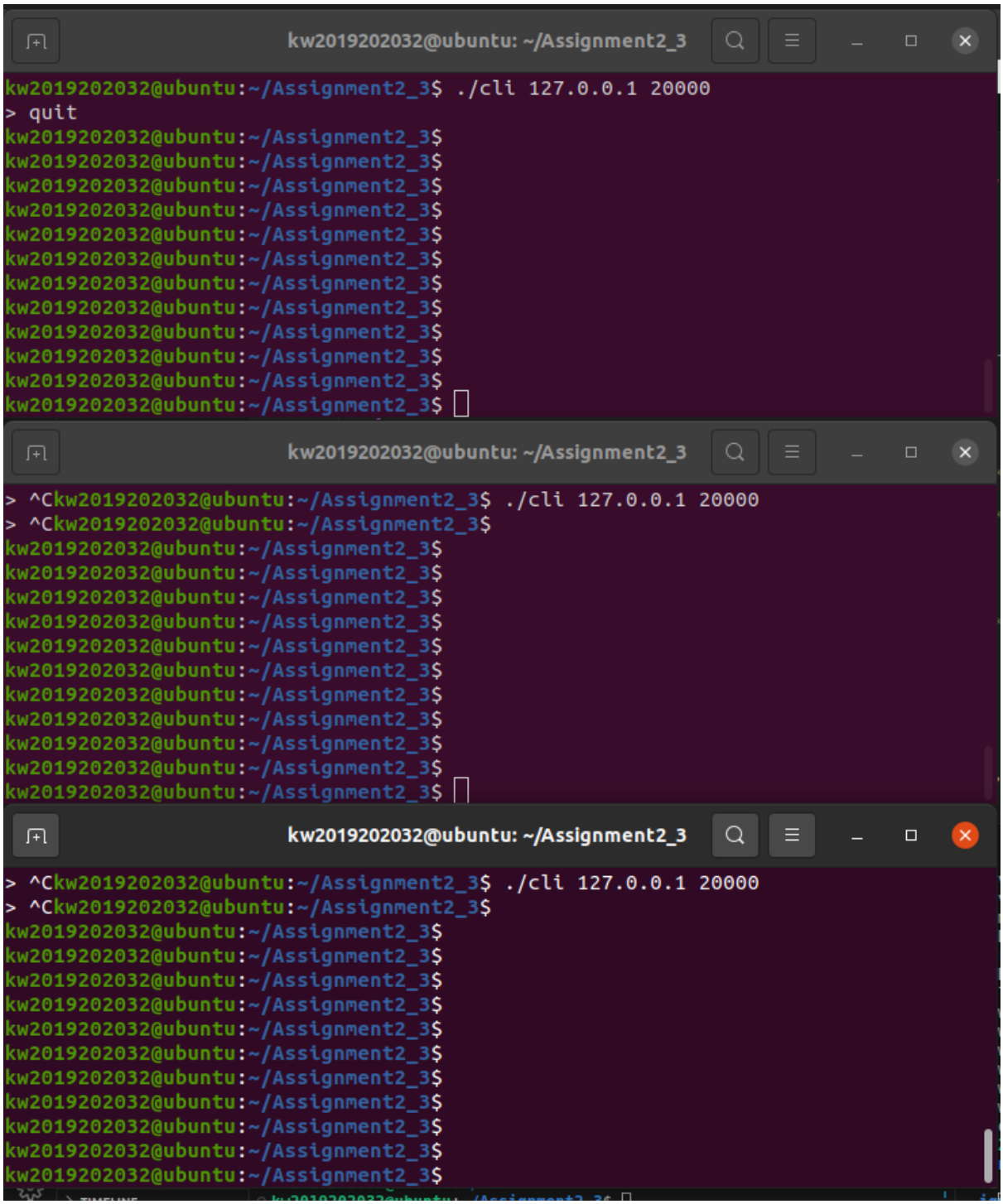
```
    /*******************************************************/
}
int main(int argc, char**argv)
{
    /******************** prepare client socket **********************/
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
/*****************************************************************/

    while(1){
        memset(buff, 0, BUF_SIZE);
        write(STDOUT_FILENO, "> ", 2);
        if(read(STDIN_FILENO, buff, BUF_SIZE) < 0){ /* receive string from user*/
        /*************** read error handling ****************/
            close(sockfd);
            exit(0);
        /*****************************************************/
        }
        buff[strlen(buff)-1] = '₩0';
       convert_command;

        if(write(sockfd, cmd_buff, BUF_SIZE) > 0){   /* send string to server */
            memset(buff, 0, BUF_SIZE);
            if((n = read(sockfd, buff, BUF_SIZE)) > 0) { /* receive string from server */
                buff[strlen(buff)] = '₩0';
```

```
                write(1, buff, strlen(buff));
                memset(buff, 0, BUF_SIZE);
            }
            else      /* error handling for read */
            {
                close(sockfd);
                shutdown(sockfd, SHUT_RDWR);
                exit(0);
            }
        }
        else /* write error handling */
        {
            close(sockfd);
            exit(0);
        }
    }
    close(sockfd);
    return 0;
}
```

결과화면

<다중접속>

<client>



위에서부터 차례대로 server 에 접속하였고, 2->1->3 번 순으로 client 를 종료하였다.

<server>

```
^Ckw2019202032@ubuntu:~/Assignment2_3$ ./srv 20000
==========Client info==========

client IP : 127.0.0.1


client port : 62656

================================

Current number of client: 1
PID     PORT    TIME
70946   62656   1
==========Client info==========

client IP : 127.0.0.1


client port : 65216

================================

Current number of client: 2
PID     PORT    TIME
70946   62656   3
70948   65216   1
==========Client info==========

client IP : 127.0.0.1


client port : 705

================================

Current number of client: 3
PID     PORT    TIME
70946   62656   7
70948   65216   5
70950   705     1
```

세 개의 client 가 각각 접속한 뒤의 화면을 보여주고 있다.

```
Current number of client: 3
PID     PORT    TIME
70946   62656   17
70948   65216   15
70950   705     11
QUIT    [70948]
Client(70948)'s Release
Current number of client: 2
PID     PORT    TIME
70946   62656   27
70950   705     21
QUIT    [70946]
Client(70946)'s Release
Current number of client: 1
PID     PORT    TIME
70950   705     31
QUIT    [70950]
Client(70950)'s Release
Current number of client: 0
PID     PORT    TIME
```

10 초뒤에 client 를 다시 보여주고, 2->1->3 순으로 client 를 종료하였을 때의 결과를 차례대로 보여주고 있다.

<LS>

<client>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
> ls
cli     cli.c   Makefile        srv     srv.c
> ls -l
-rwxrwxr-x 1 kw2019202032 kw2019202032 17424 May 13 07:4907:49 cli
-rw-rw-r-- 1 kw2019202032 kw2019202032 6151 May 13 07:4807:48 cli.c
-rwxrw-rw- 1 kw2019202032 kw2019202032 162 Apr 17 07:2107:21 Makefile
-rwxrwxr-x 1 kw2019202032 kw2019202032 39944 May 13 09:0309:03 srv
-rw-rw-r-- 1 kw2019202032 kw2019202032 38834 May 13 09:0309:03 srv.c
> ls -al
drwxrwxr-x 3 kw2019202032 kw2019202032 4096 May 13 09:0309:03 ./
drwxr-xr-x 29 kw2019202032 kw2019202032 4096 May 11 13:0713:07 ../
drwxrwxr-x 2 kw2019202032 kw2019202032 4096 May 11 11:1311:13 .vscode/
-rwxrwxr-x 1 kw2019202032 kw2019202032 17424 May 13 07:4907:49 cli
-rw-rw-r-- 1 kw2019202032 kw2019202032 6151 May 13 07:4807:48 cli.c
-rwxrw-rw- 1 kw2019202032 kw2019202032 162 Apr 17 07:2107:21 Makefile
-rwxrwxr-x 1 kw2019202032 kw2019202032 39944 May 13 09:0309:03 srv
-rw-rw-r-- 1 kw2019202032 kw2019202032 38834 May 13 09:0309:03 srv.c
> ls -a
./      ../     .vscode/        cli     cli.c
Makefile        srv     srv.c
> quit
```

<server>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./srv 20000
==========Client info==========

client IP : 127.0.0.1


client port : 40672

================================

Current number of client: 1
PID     PORT    TIME
71819   40672   1
NLST    [71819]
NLST -l [71819]
NLST -al        [71819]
Current number of client: 1
PID     PORT    TIME
71819   40672   11
NLST -a [71819]
QUIT    [71819]
Client(71819)'s Release
```

Ls 명령어를 입력한 client 의 정보를 srv 에서 출력하고, cli 에서 결과를 출력하고 있다.

<DIR>

<client>

```
> ^Ckw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
> dir
-rw-rw-r-- 1 kw2019202032 kw2019202032 0 May 13 07:4307:43 b
-rwxrwxr-x 1 kw2019202032 kw2019202032 17424 May 13 07:4907:49 cli
-rw-rw-r-- 1 kw2019202032 kw2019202032 6151 May 13 07:4807:48 cli.c
-rwxrw-rw- 1 kw2019202032 kw2019202032 162 Apr 17 07:2107:21 Makefile
-rwxrwxr-x 1 kw2019202032 kw2019202032 39944 May 13 08:4008:40 srv
-rw-rw-r-- 1 kw2019202032 kw2019202032 38420 May 13 08:4008:40 srv.c
```

<server>

```
^Ckw2019202032@ubuntu:~/Assignment2_3$ ./srv 20000
==========Client info==========

client IP : 127.0.0.1


client port : 59564

================================

Current number of client: 1
PID     PORT    TIME
71001   59564   1
LIST    [71001]
QUIT    [71001]
Client(71001)'s Release
```

Dir 를 통해 파일을 출력한다.

<MKDIR + CD + PWD + RMDIR>

<client1>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
> mkdir new_dir1 new_dir2
MKD new_dir1
MKD new_dir2
```

<client2>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
```

두개의 client 에서 접속한다.

<server>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./srv 20000
==========Client info==========

client IP : 127.0.0.1


client port : 29907

================================

Current number of client: 1
PID      PORT     TIME
71829    29907    1
==========Client info==========

client IP : 127.0.0.1


client port : 31955

================================

Current number of client: 2
PID      PORT     TIME
71829    29907    2
71831    31955    1
MKD      [71829]
```

두개의 client 가 접속되었고, 첫번째 client 에서 mkdir 을 통해 new_dir1 new_dir2 를 생성한다.

<client2>

```
> ls
cli      cli.c    Makefile         new_dir1/        new_dir2/
srv      srv.c
> cd new_dir1
"/home/kw2019202032/Assignment2_3/new_dir1" is current directory
> pwd
"/home/kw2019202032/Assignment2_3/new_dir1" is current directory
> cd ..
"/home/kw2019202032/Assignment2_3" is current directory
> rmdir new_dir2 new_dir2
RMD new_dir2
Error : failed to remove 'new_dir2'
> rmdir new_dir1
RMD new_dir1
```

두번째 client 에서 ls 를 통해서 new_dir1, new_dir2 가 만들어졌음을 확인할 수 있고, cd 를 통해 new_dir1 로 이동한 뒤, pwd 를 실행한다. 이후 cd ..를 통해 이전 directory 로 이동하고, rmdir 명령어를 통해 만들어진 2 개의 new_dir1, new_dir2 를 삭제한다.

<server>

```
NLST    [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   22
71831   31955   21
CWD new_dir1    [71831]
PWD     [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   32
71831   31955   31
CDUP    [71831]
RMD     [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   42
71831   31955   41
RMD     [71831]
```

Server 화면을 통해 두번째 client 에서 명령어를 실행했음을 확인할 수 있다.

<client1>

```
> ls
cli     cli.c   Makefile        srv     srv.c
> quit
kw2019202032@ubuntu:~/Assignment2_3$
```

첫번째 client 에서 ls 를 출력하면 new_dir1, new_dir2 가 삭제되었음을 확인할 수 있다.

<server>

```
NLST    [71829]
Current number of client: 2
PID     PORT    TIME
71829   29907   62
71831   31955   61
QUIT    [71829]
Client(71829)'s Release
QUIT    [71831]
Client(71831)'s Release
^Ckw2019202032@ubuntu:~/Assign
```

server 화면에서 client 의 접속기록을 확인한다.

<전체 실행화면>

<client1>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
> mkdir new_dir1 new_dir2
MKD new_dir1
MKD new_dir2
> ls
cli     cli.c   Makefile        srv     srv.c
> quit
kw2019202032@ubuntu:~/Assignment2_3$
```

<client2>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
> ls
cli     cli.c   Makefile        new_dir1/       new_dir2/
srv     srv.c
> cd new_dir1
"/home/kw2019202032/Assignment2_3/new_dir1" is current directory
> pwd
"/home/kw2019202032/Assignment2_3/new_dir1" is current directory
> cd ..
"/home/kw2019202032/Assignment2_3" is current directory
> rmdir new_dir2 new_dir2
RMD new_dir2
Error : failed to remove 'new_dir2'
> rmdir new_dir1
RMD new_dir1
> quit
```

<server>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./srv 20000
==========Client info==========

client IP : 127.0.0.1


client port : 29907

================================

Current number of client: 1
PID     PORT    TIME
71829   29907   1
==========Client info==========

client IP : 127.0.0.1


client port : 31955

================================

Current number of client: 2
PID     PORT    TIME
71829   29907   2
71831   31955   1
MKD     [71829]
Current number of client: 2
PID     PORT    TIME
71829   29907   12
71831   31955   11
NLST    [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   22
71831   31955   21
CWD new_dir1    [71831]
PWD     [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   32
71831   31955   31
CDUP    [71831]
```

```
Current number of client: 2
PID     PORT    TIME
71829   29907   22
71831   31955   21
CWD new_dir1    [71831]
PWD     [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   32
71831   31955   31
CDUP    [71831]
RMD     [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   42
71831   31955   41
RMD     [71831]
Current number of client: 2
PID     PORT    TIME
71829   29907   52
71831   31955   51
NLST    [71829]
Current number of client: 2
PID     PORT    TIME
71829   29907   62
71831   31955   61
QUIT    [71829]
Client(71829)'s Release
QUIT    [71831]
Client(71831)'s Release
```

<RENAME + DELETE>

<client>

```
kw2019202032@ubuntu:~/Assignment2_3$ touch a
kw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
> ls
a       cli     cli.c   Makefile        srv
srv.c
> rename a b
RNFR a
RNTO b
> ls
b       cli     cli.c   Makefile        srv
srv.c
> delete b
DELE b
> ls
cli     cli.c   Makefile        srv     srv.c
> ^Ckw2019202032@ubuntu:~/Assignment2_3$
```

<server>

```
kw2019202032@ubuntu:~/Assignment2_3$ ./srv 20000
==========Client info==========

client IP : 127.0.0.1


client port : 132

================================

Current number of client: 1
PID     PORT    TIME
71850   132     1
NLST    [71850]
RNFR    [71850]
NLST    [71850]
Current number of client: 1
PID     PORT    TIME
71850   132     11
DELE    [71850]
NLST    [71850]
QUIT    [71850]
Client(71850)'s Release
```

<QUIT>

<server>

```
^Ckw2019202032@ubuntu:~/Assignment2_3$ ./srv 20000
==========Client info==========

client IP : 127.0.0.1


client port : 35537

================================

Current number of client: 1
PID      PORT     TIME
71853    35537    1
QUIT     [71853]
Client(71853)'s Release
```

Client 를 종료하고 server 에서는 해당 process 를 삭제한다.

<client>

```
> ^Ckw2019202032@ubuntu:~/Assignment2_3$ ./cli 127.0.0.1 20000
> quit
```

고찰

해당 과제를 진행하는 과정에서 가장 어려웠던 점은 여러 개의 clients 를 등록하고 관리하는 것과 어떤 clients 가 종료되었을 때, 해당 clients 를 배열에서 삭제하는 것이었다. 이를 해결하기 위해서 child process 들은 구조체 배열을 사용하여 배열에 저장하여 관리하였고, 연결된 socket 이 종료될 때마다 event 를 실행할 수 있도록 해주는 signal(SIGHLD)를 사용하여 어떤 client 가 종료될 때마다 배열에서 삭제하고 process 를 종료하였다. 또한 10 초마다 알람이 오도록 하는 것 또한 어려웠는데, 특히 어떤 client 가 새롭게 연결될 때마다 다시 10 초를 카운팅 해줘야한다는 것을 해결하는데 어려움이 있었다. 이를 해결하기 위해서 alarm(10)을 새로운 client 가 연결될 때마다 다시 등록하였고, 이를 통해 문제없이 실행될 수 있도록 하였다. 이번 과제를 통해 signal 에 대해 자세히 알게되었고, 여러 client 들의 다중 접속을 허용하고 관리하는 방법에 대해 익히게 되었다.

Reference

강의자료만 참고하였습니다.