시스템 프로그래밍 실습

10차 과제

실습 일시 : 화 1,2
담당 교수님 : 김태석 교수님
학번 : 2013722095
이름 : 최재은
실습 번호 : Assignment #3 FTP3

FTP3

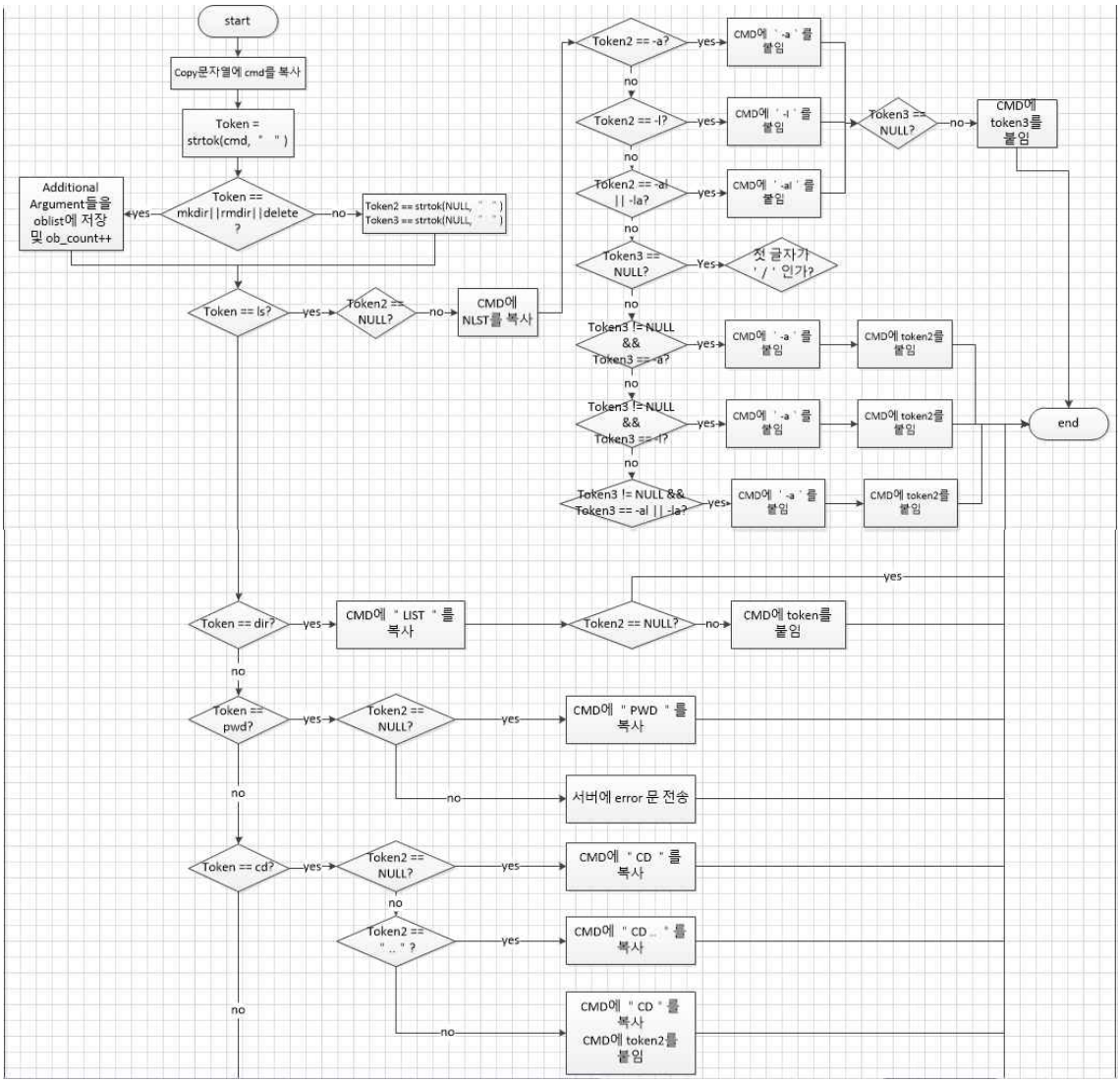
■ Introduction

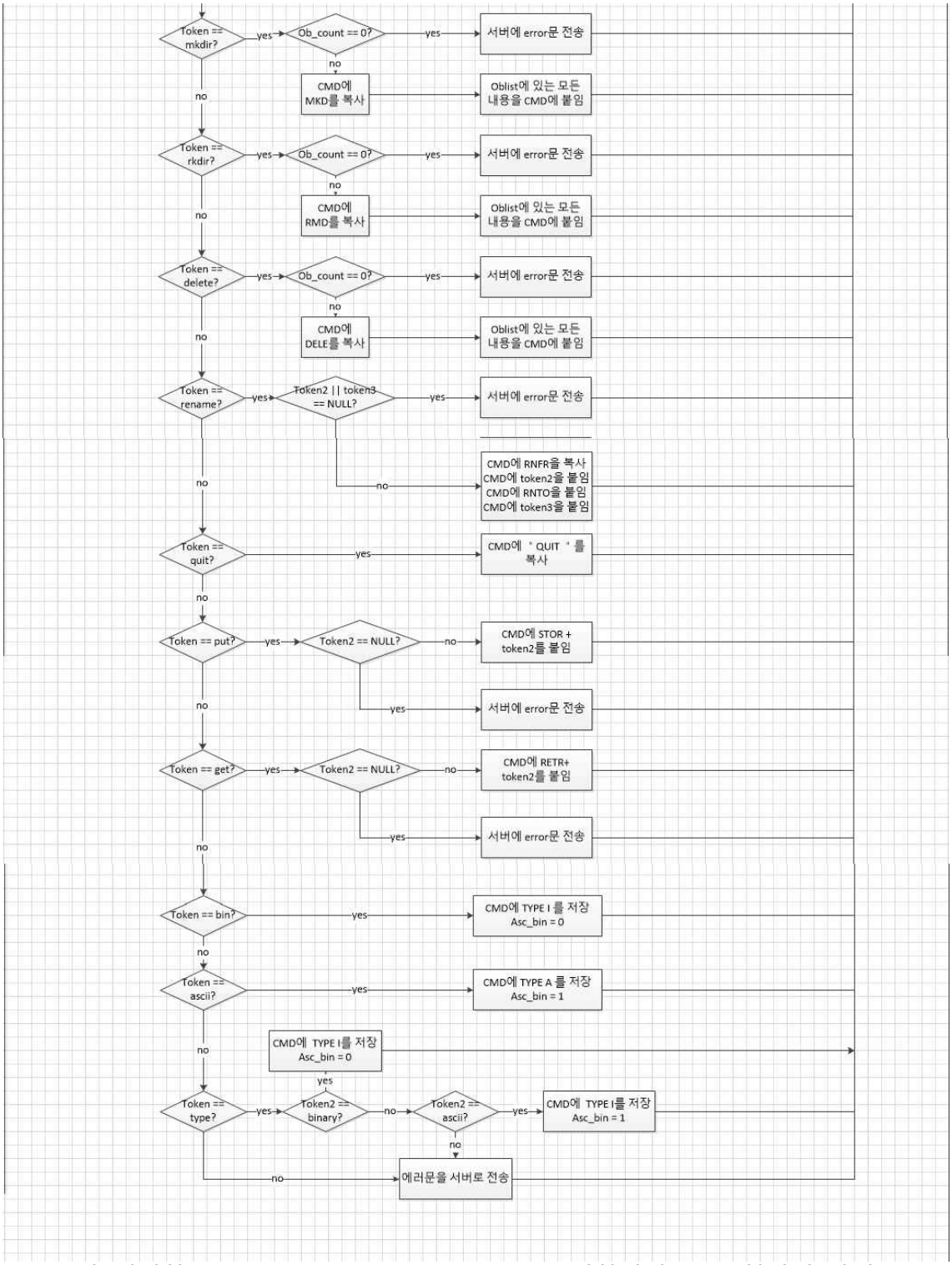
Assignment #2에서 구현했던 내용에 덧붙여, 기존의 1개 link를 통해 데이터를 주고 받았던 방식에서 벗어나 Data 전송 전용의 포트를 새로 만들어 연결한다. 각각의 명령어는 그에 대한 server의 reply를 받을 수 있으며, Data link를 통해서 파일을 주고 받을 수 있도록 구현한다.

■ Flow Chart

1. cli.c

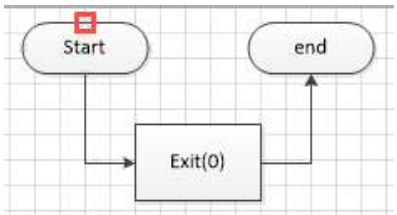
1) convcomm





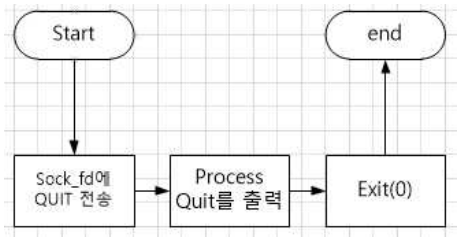
- user가 입력한 command를 FTP용 command로 변환시켜 주고 형식에 맞게 재정렬 해주는 함수

2) sh_term



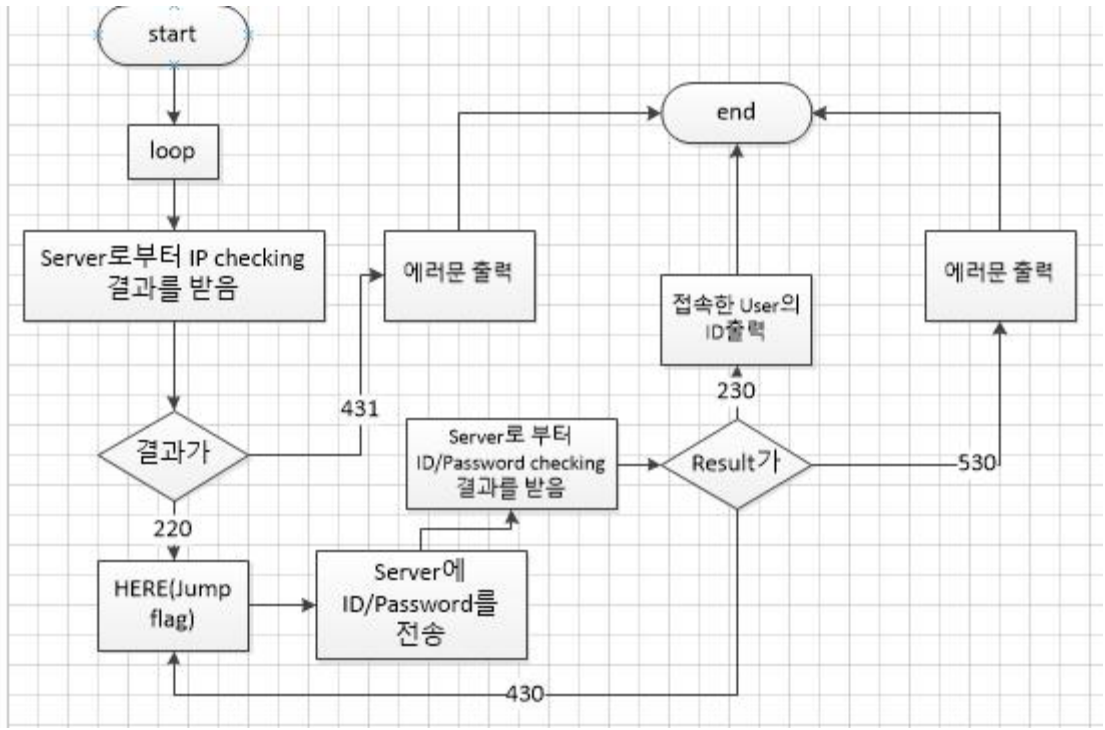
- SIGTERM signal handler

3) sh_int

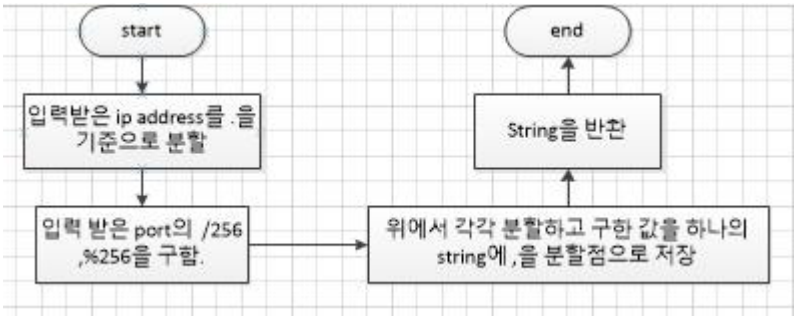


- SIGINT signal handler

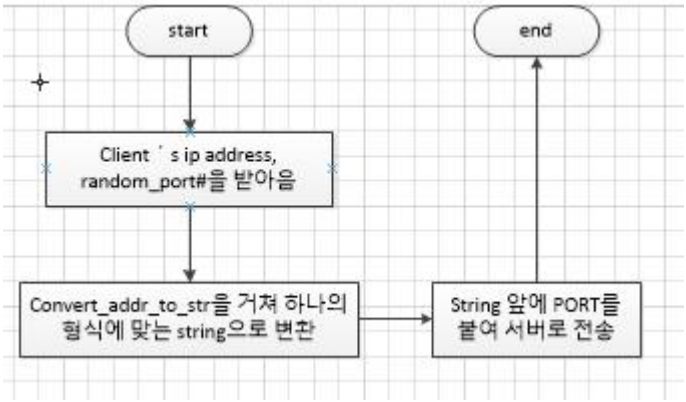
4)log_in



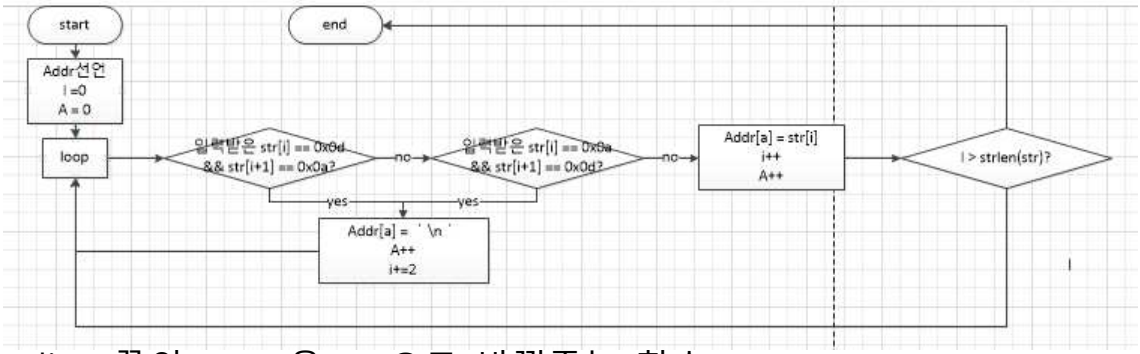
5) convert_addr_to_str



6) port_func

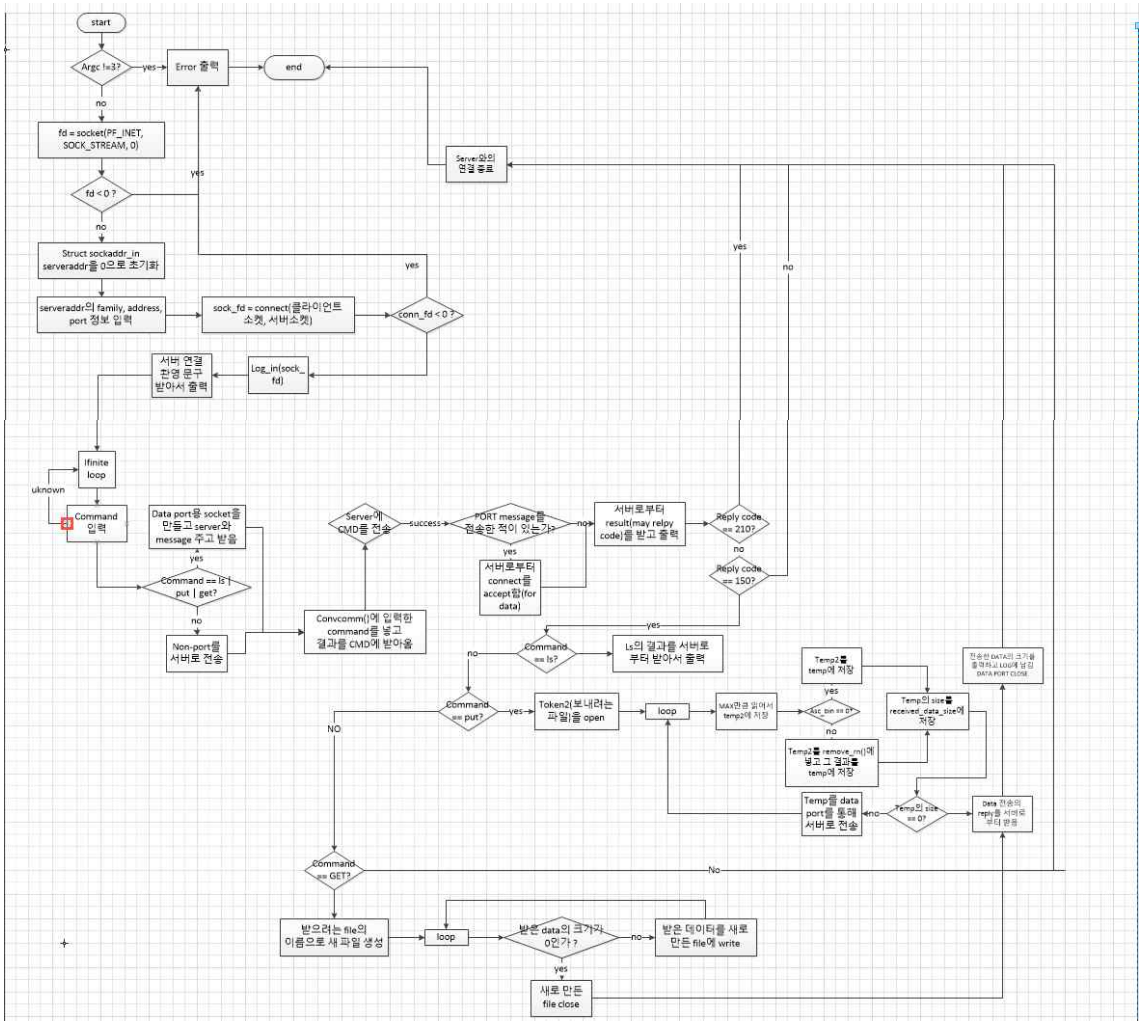


7) remove_rn



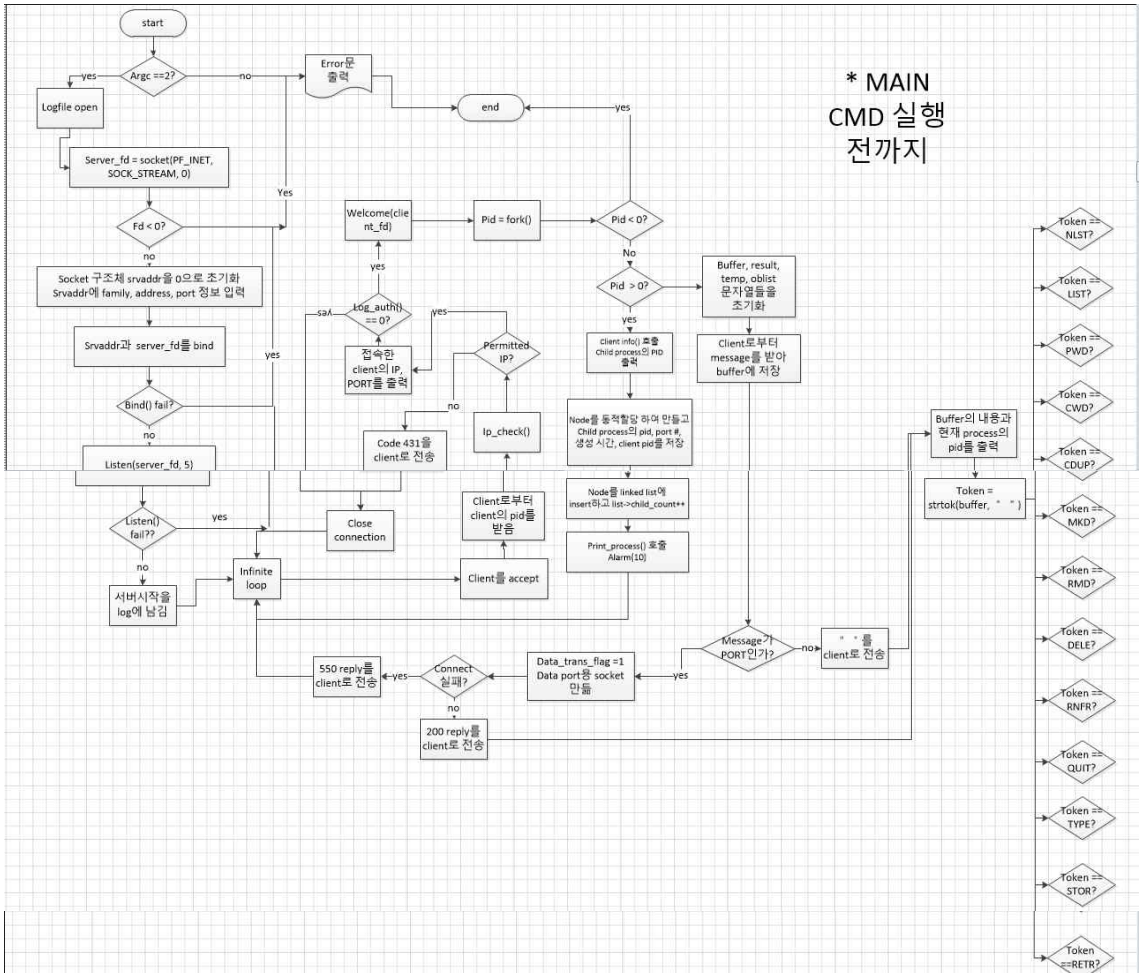
- line 끝의 ₩r₩n을 ₩n으로 바꿔주는 함수

4) client 전체 동작

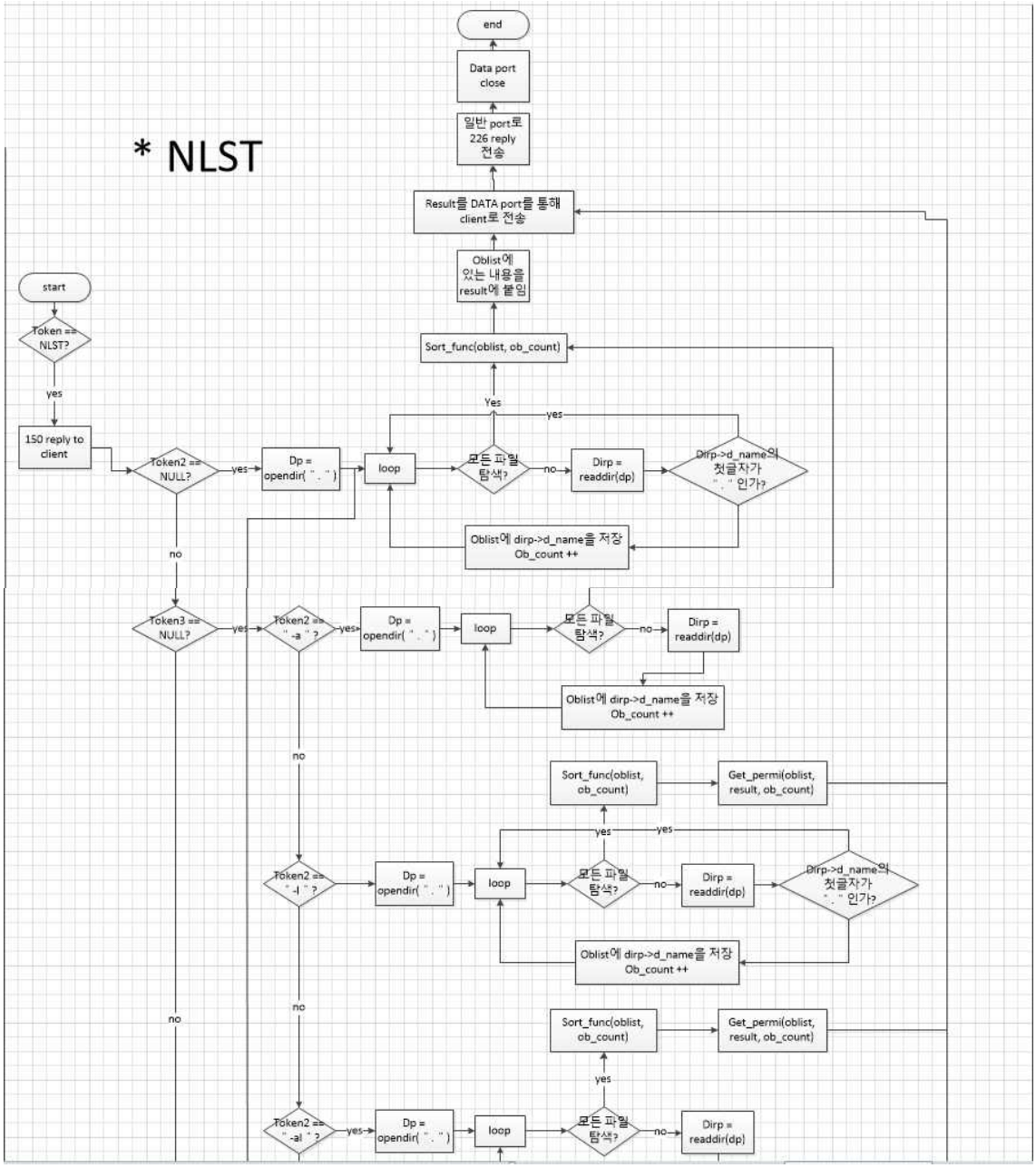


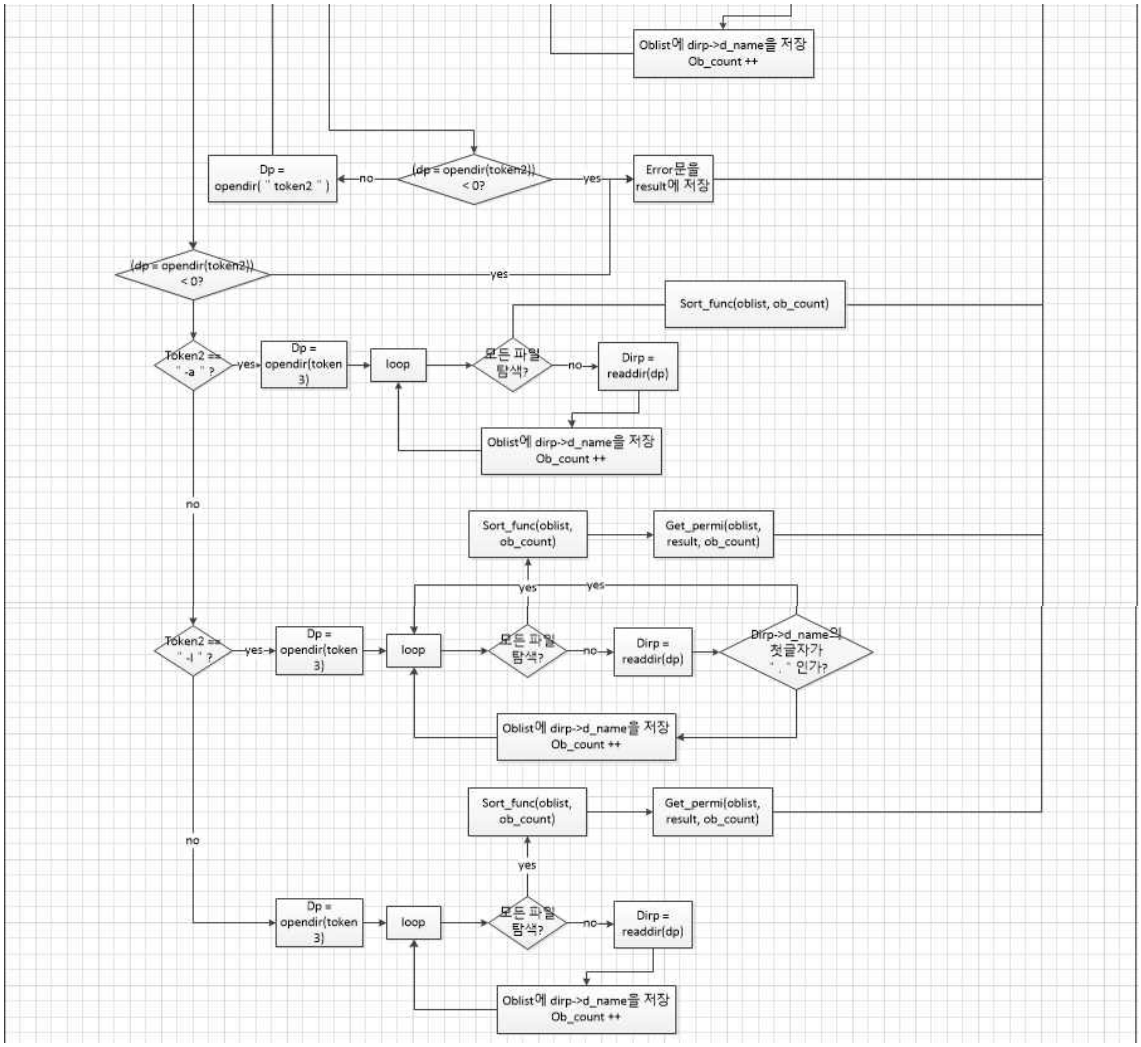
2. srv.c

1) 각 command진입 전까지

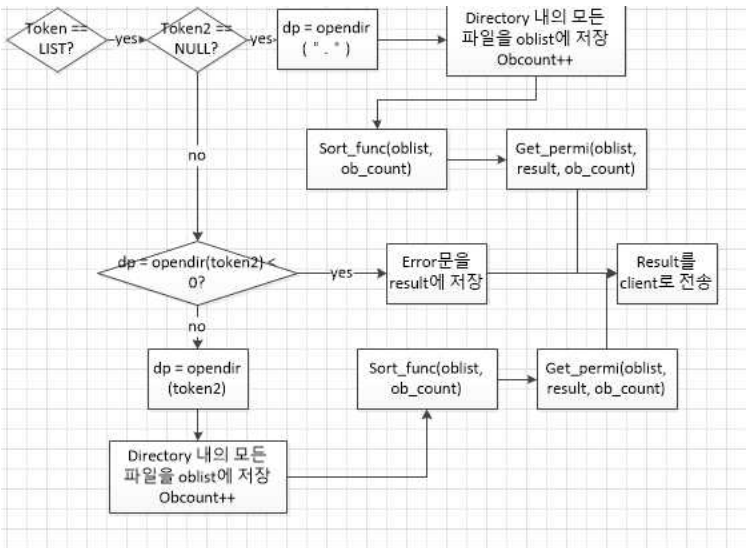


*NLST

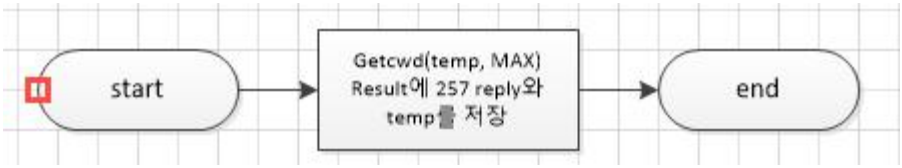

* NLST
end
Data port close
일반 port로 226 reply 전송
Result를 DATA port를 통해 client로 전송
Oblist에 있는 내용을 result에 붙임
Sort_func(oblist, ob_count)
start
Token == NLST?
yes
150 reply to client
Token2 == NULL?
yes
Dp = opendir(" . ")
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Dirp->d_name의 첫글자가 " . " 인가?
Yes
yes
Oblist에 dirp->d_name을 저장 Ob_count ++
no
Token3 == NULL?
yes
Token2 == " -a " ?
yes
Dp = opendir(" . ")
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Oblist에 dirp->d_name을 저장 Ob_count ++
no
Sort_func(oblist, ob_count)
Get_permi(oblist, result, ob_count)
yes
yes
Token2 == " -l " ?
Dp = opendir(" . ")
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Dirp->d_name의 첫글자가 " . " 인가?
Oblist에 dirp->d_name을 저장 Ob_count ++
no
no
Sort_func(oblist, ob_count)
Get_permi(oblist, result, ob_count)
yes
Token2 == " -al " ?
yes
Dp = opendir(" . ")
loop
모든 파일 탐색?
no
Dirp = readdir(dp)


Oblist에 dirp->d_name을 저장
Ob_count ++
Dp = opendir(" token2 ")
no
(dp = opendir(token2)) < 0?
yes
Error문을 result에 저장
(dp = opendir(token2)) < 0?
yes
Sort_func(oblist, ob_count)
Token2 == " -a " ?
yes
Dp = opendir(token 3)
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Oblist에 dirp->d_name을 저장
Ob_count ++
no
Sort_func(oblist, ob_count)
Get_permi(oblist, result, ob_count)
yes
yes
Token2 == " -l " ?
yes
Dp = opendir(token 3)
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Dirp->d_name의 첫글자가 " . " 인가?
Oblist에 dirp->d_name을 저장
Ob_count ++
Sort_func(oblist, ob_count)
Get_permi(oblist, result, ob_count)
no
yes
Dp = opendir(token 3)
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Oblist에 dirp->d_name을 저장
Ob_count ++

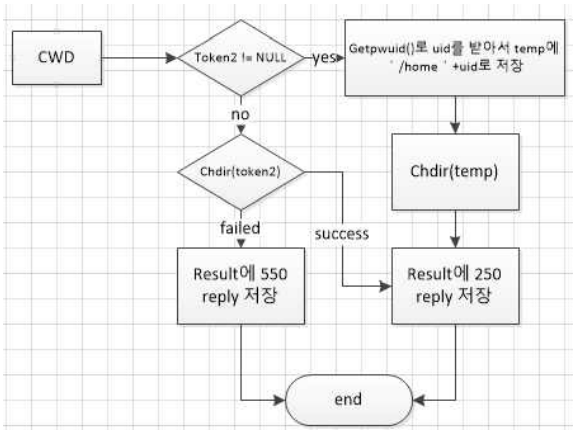
*LIST


Token == LIST?
yes
Token2 == NULL?
yes
dp = opendir (" . ")
Directory 내의 모든 파일을 oblist에 저장 Obcount++
Sort_func(oblist, ob_count)
Get_permi(oblist, result, ob_count)
no
dp = opendir(token2) < 0?
yes
Error문을 result에 저장
Result를 client로 전송
no
dp = opendir (token2)
Sort_func(oblist, ob_count)
Get_permi(oblist, result, ob_count)
Directory 내의 모든 파일을 oblist에 저장 Obcount++

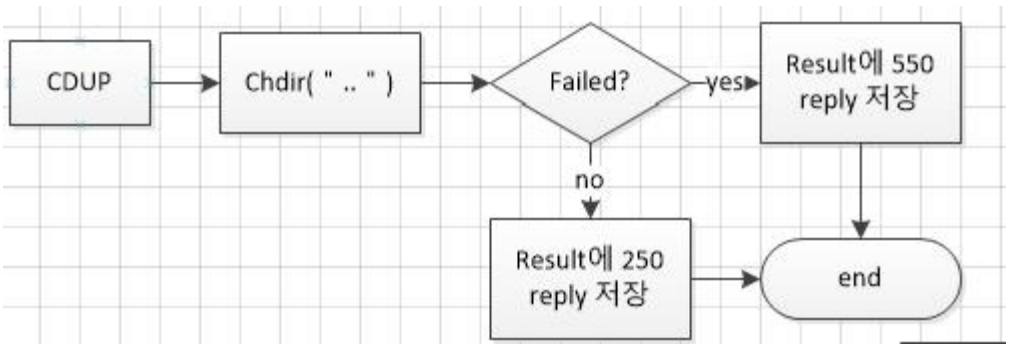
*PWD



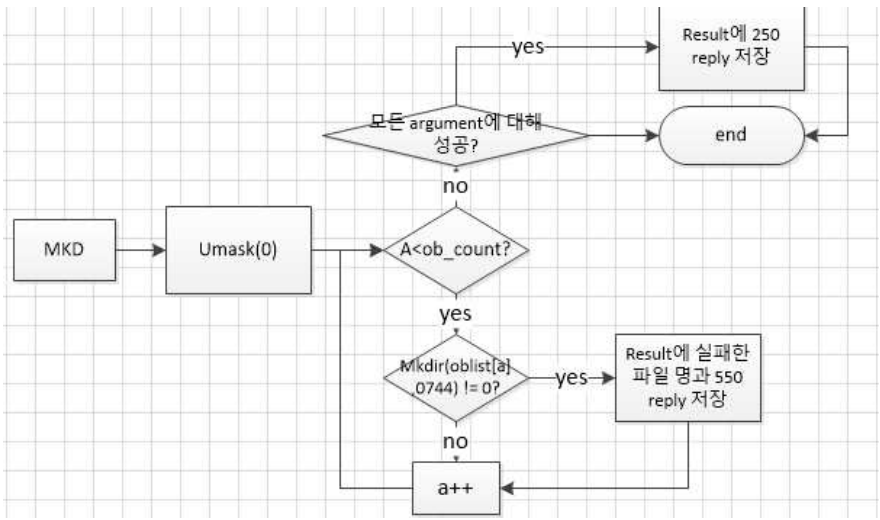
*CWD



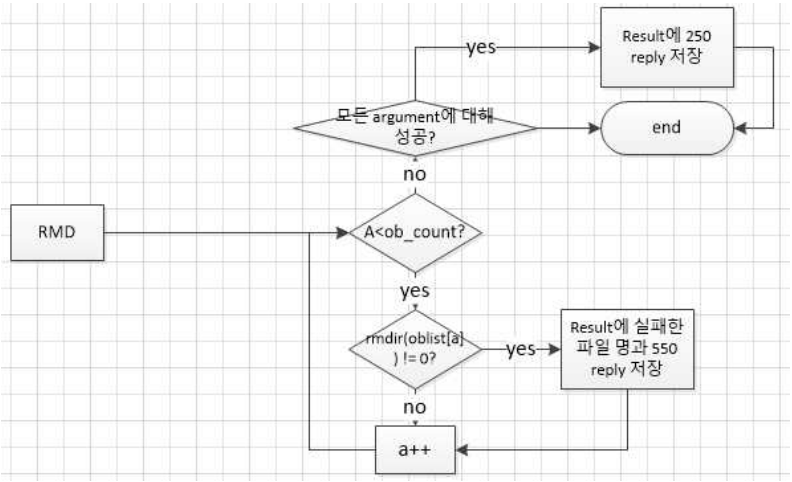
*CDUP



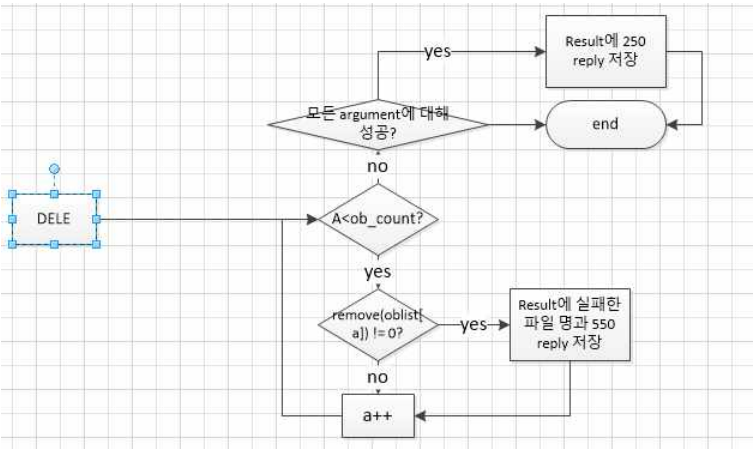
*MKD



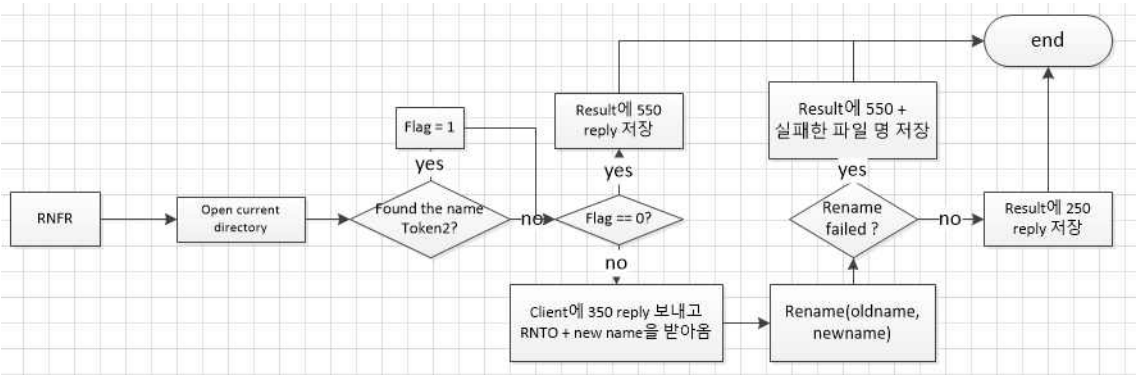
*RMD


yes
Result에 250
reply 저장
모든 argument에 대해
성공?
end
no
RMD
A<ob_count?
yes
rmdir(oblist[a]
) != 0?
yes
Result에 실패한
파일 명과 550
reply 저장
no
a++

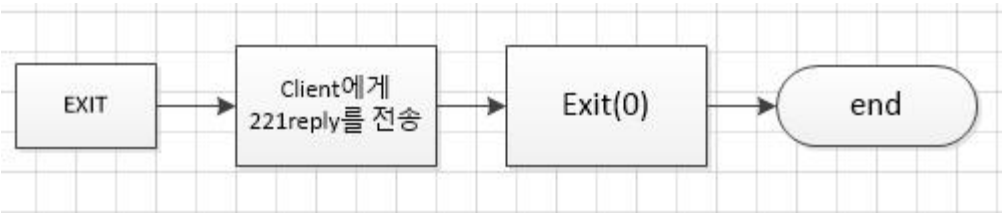
*DELE


yes
Result에 250
reply 저장
모든 argument에 대해
성공?
end
no
DELE
A<ob_count?
yes
remove(oblist[
a]) != 0?
yes
Result에 실패한
파일 명과 550
reply 저장
no
a++

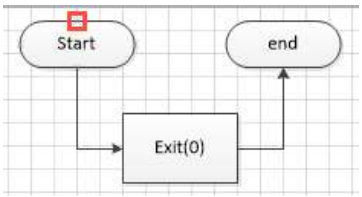
*RNFR & RNTO



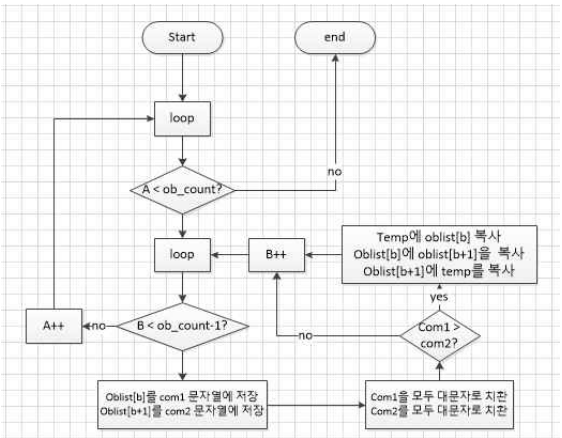
*QUIT



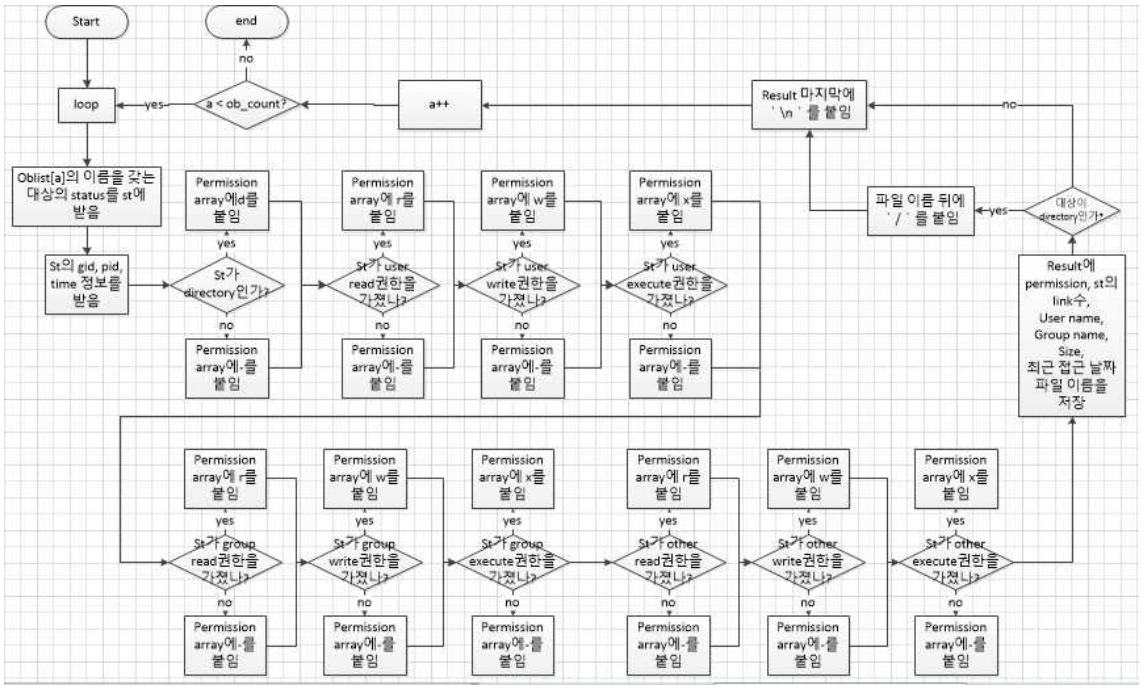
2) sh_term



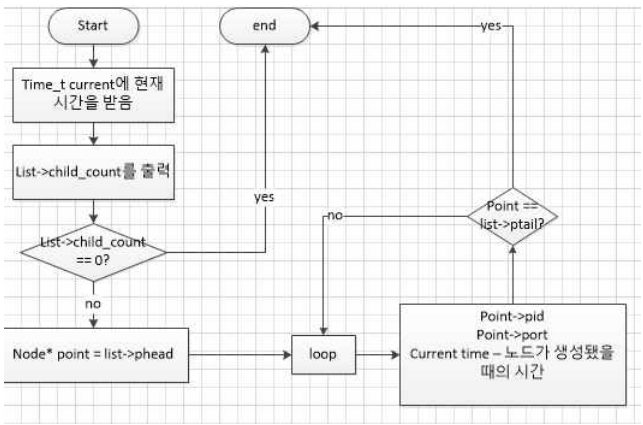
3) sort_func



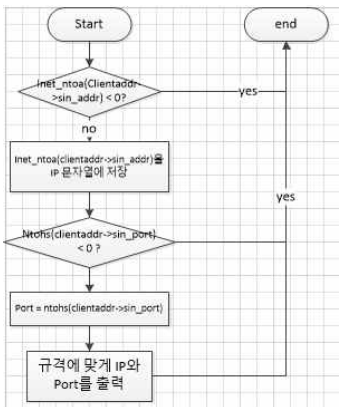
4) get_permi



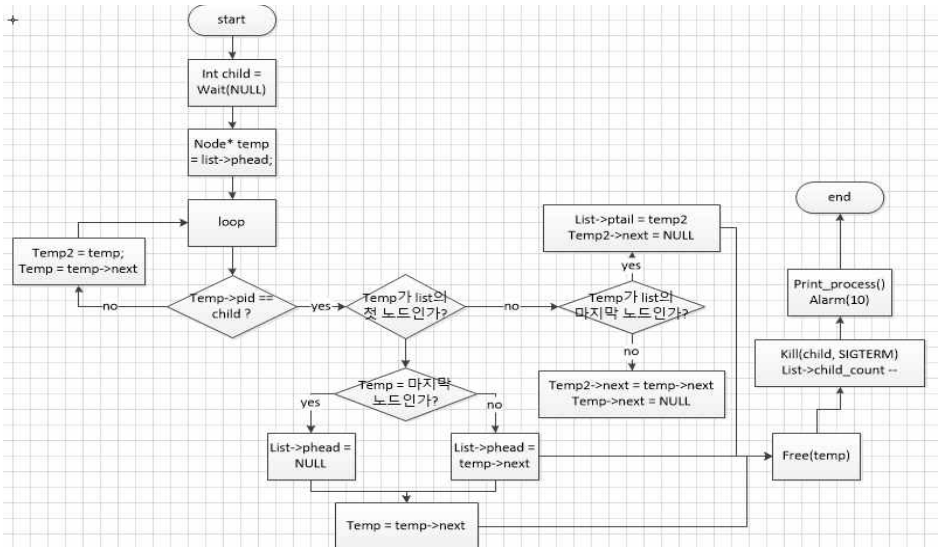
5) printf_process



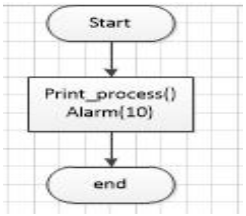
6) client_info



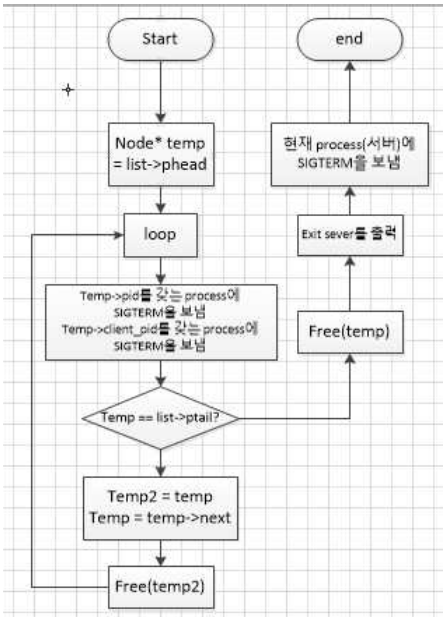
7)sh_chld



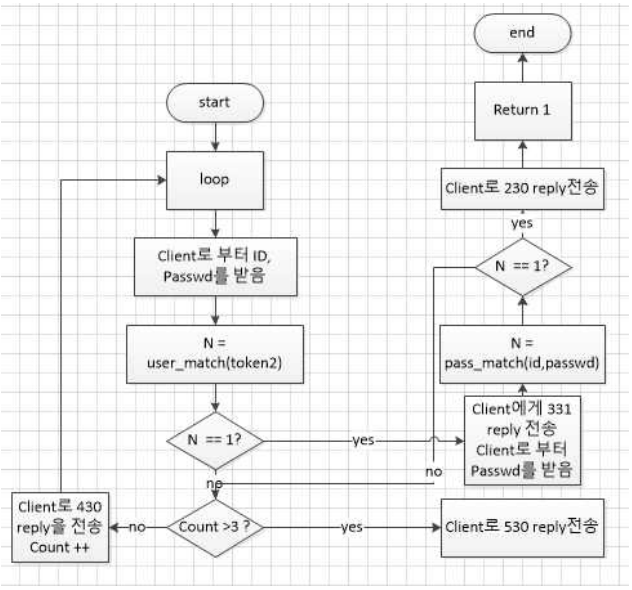
8) sh_alrm



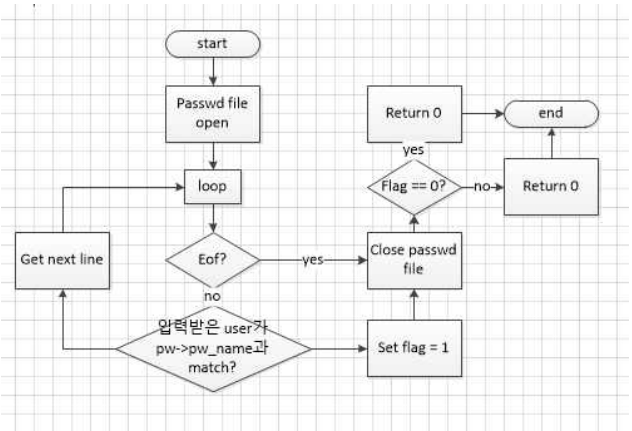
9)sh_int



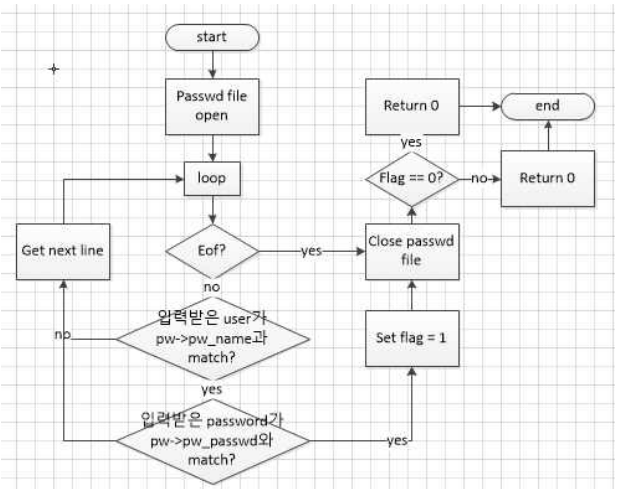
10) log_auth



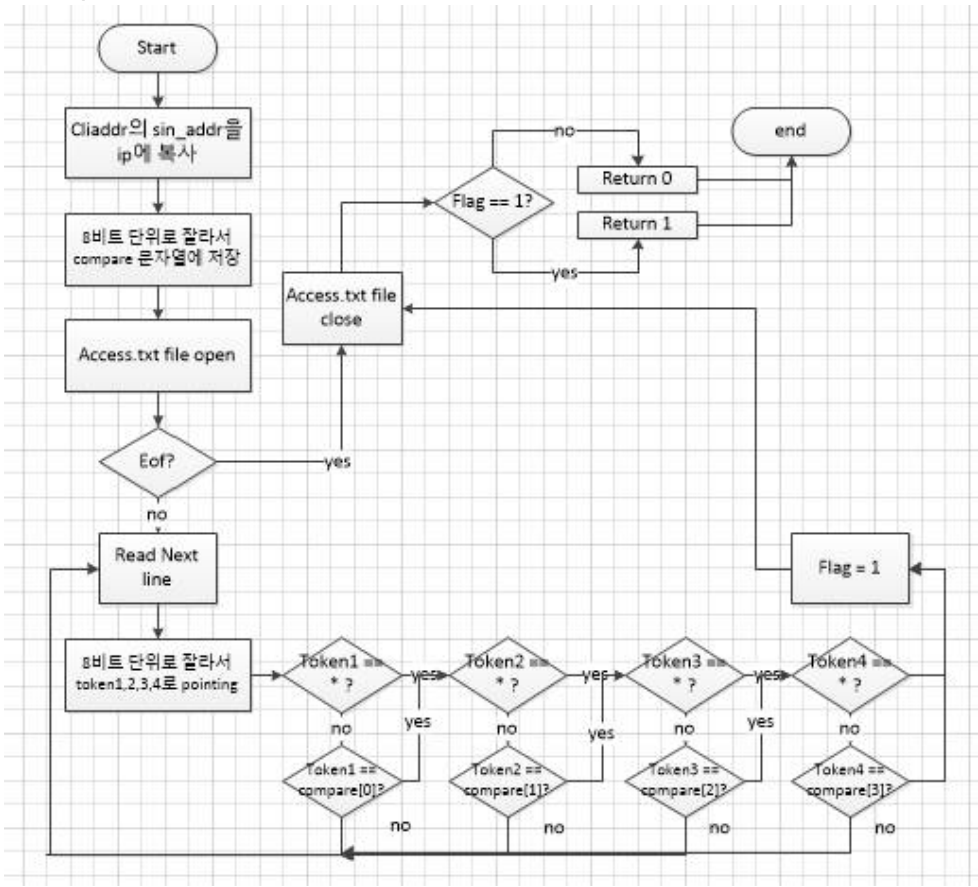
11)user_match



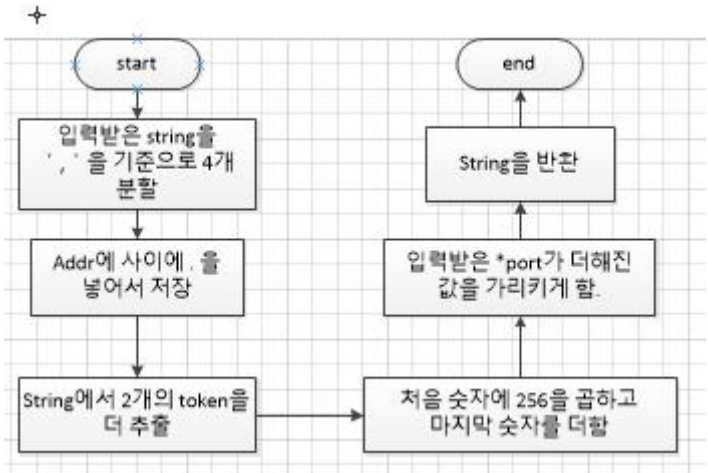
12) pass_match



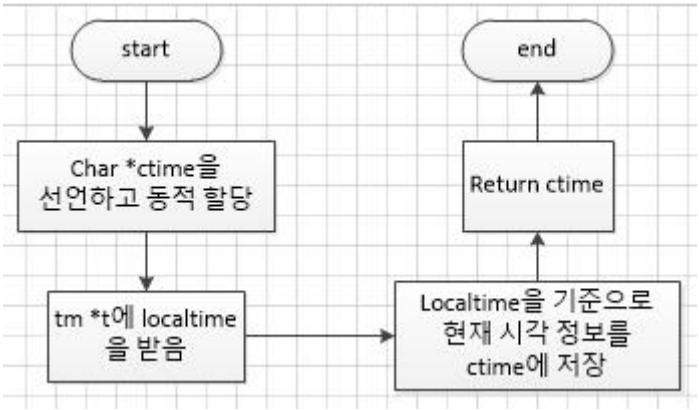
13) ip_check



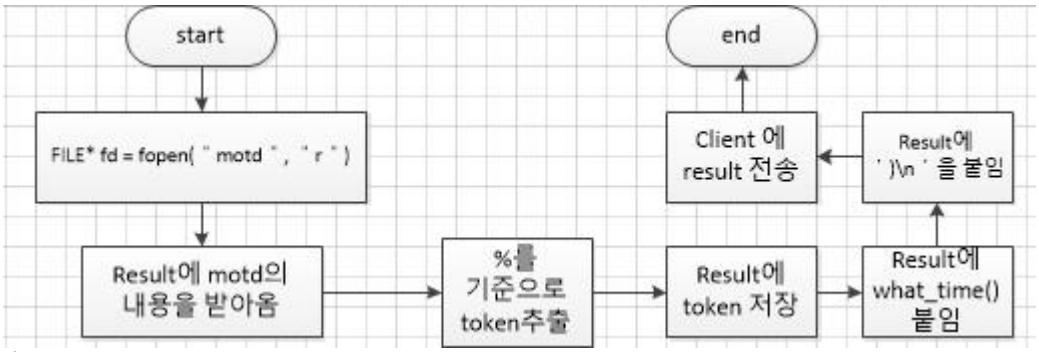
14) convert_str_to_addr



15) what_time



16) welcome



17) remove_rn

- client의 것과 동일

■ Pseudo code

1. cli.c

```
void sh_term(int signal)
{
    프로세스 종료
}

void sh_int(int signal)
{
    QUIT 문자열을 서버로 전송.
    Process quit....을 cmd창에 출력하고
    프로세스 종료
}

int convcomm(char* cmd, char* CMD)
{
    if(mkdir, rmdir, delete command 일 때)
    {
        명령어 뒤의 argument들을 oblist에 저장
        그 개수를 ob_count에 저장
    }

    else
    {
        strtok를 통해 token2, token3를 얻음.
    }

    if(명령어가 ls)
    {
        CMD에 NLST를 저장함.
        if(token2 != NULL)
        {
            if(token2 == -a)
            {
                CMD에 –a를 붙임
                if(token3 != NULL) token3도 CMD에 붙임,
            }
```

```
else if(token2 == -l)
{
    CMD에 –l를 붙임
    if(token3 != NULL) token3도 CMD에 붙임,
}
else if(token2 == -al || -la)
{
    CMD에 –al를 붙임
    if(token3 != NULL) token3도 CMD에 붙임,
}
else if(token3 == NULL)
{
    if(token3의 첫글자가 /가 아니면)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    CMD에 token2를 붙임.
}
else if(token3 != NULL && token2 == -a)
{
    CMD에 –a를 붙임
    token3도 CMD에 붙임,
}
else if(token3 != NULL && token2 == -l)
{
    CMD에 –l를 붙임
    token3도 CMD에 붙임,
}
else if(token3 != NULL && token2 == -al)
{
    CMD에 –al를 붙임
    token3도 CMD에 붙임,
}
else
{ 현재 cmd창에 에러문 출력
 서버로 에러문 전송 return –1
}
```

```
    }
}

else if(명령어가 dir)
{
    if(token3 != NULL)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    CMD에 LIST를 저장
    if(token2 != NULL)
        CMD에 token2를 붙임

}
else if(명령어가 cd)
{
    if(token3 != NULL)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    if(token2 == NULL)
        CMD에 'CWD'를 저장
    else if(token2 == "..")
        CMD에 'CDUP .. "를 저장
    else
        CMD에 'CWD'+ token2를 저장
}
else if(명령어가 pwd)
{
    if(token2 == NULL)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    CMD에 'PWD'를 저장
}
else if(명령어가 mkdir)
{
```

```
    if(ob_count == 0)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    CMD문자열에 'MKD'를 저장
    oblist에 저장돼있는 모든 argument들을 CMD 뒤에 붙임
}
else if(명령어가 rmdir)
{
    if(ob_count == 0)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    CMD문자열에 'RMD'를 저장
    oblist에 저장돼있는 모든 argument들을 CMD 뒤에 붙임
}
else if(명령어가 delete)
{
    if(ob_count == 0)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    CMD문자열에 'DELE'를 저장
    oblist에 저장돼있는 모든 argument들을 CMD 뒤에 붙임
}
else if(명령어가 rename)
{
    if(token2 || token3 가 NULL이면)
    { 현재 cmd창에 에러문 출력
     서버로 에러문 전송 return –1
    }
    CMD 문자열에 'RNFR' token2 'RNTO' token3를 저장}
}
else if(명령어가 quit)
{
    CMD 문자열에 'QUIT'를 저장}
```

```
else if( put )
{
    if(token2 == NULL)
        에러문을 출력하고 서버로 전송 return -1
    CMD에 'STOR 'filename''을 저장
}
else if( get )
{
    if(token2 == NULL)
        에러문을 출력하고 서버로 전송 return -1
    CMD에 'RETR 'filename''을 저장
}
else if( bin )
{
    CMD에 'TYPE I'를 저장
    asc_bin을 0으로 셋
}
else if( ascii )
{
    CMD에 'TYPE A'를 저장
    asc_bin을 1로 셋
}
else if( type )
{
    if( token2 == NULL)
        에러문을 출력하고 서버로 전송 return -1
    if( token2 == binary)
    {
        CMD에 'TYPE I'를 저장
        asc_bin을 0으로 셋
    }
    else if( token2 == ascii )
    {
        CMD에 'TYPE A'를 저장
        asc_bin을 1로 셋
    }
```

```
        else
            에러문을 출력하고 서버로 전송; return -1;
    }
    else
    {
        현재 cmd창에 에러문 출력
        서버로 에러문 전송 return -1
    }

    return 0
}
void log_in(int sockfd)
{
    for(;;)
    {
        server로부터 IP check result를 받음
        if( 승인된 IP)
        {
            user에게서 ID를 입력 받음
            서버에게 ID를 보내고 그 결과를 받음
            if(530 code ) connect close하고 종료
            if(430 code ) ID입력으로 돌아감
            else
            {
                user에게서 passwd를 입력 받음
                서버에게 passwd를 보내고 그 결과를 받음
                if(530 code ) connect close하고 종료
                if(430 code ) ID입력으로 돌아감
                else break;
            }

        }
        else
        {
            client와의 connect를 close
            exit(1)
        }
```

```
    }
}

char* convert_addr_to_str(unsigned long ip_addr, unsigned int port)
{
    char *addr을 동적 할당하여 저장 공간 마련
    ip_addr을 .을 기준으로 4분할 하여 addr에 저장
    port를 256으로 나눠서 head값 준비
    port를 256으로 나눈 나머지를 tail에 저장
    addr에 head, tail을 붙임
}
void port_func(char *client_ip, int random_port)
{
    convert_addr_to_str에 client_ip, random_port를 넣은 결과 앞에
    "PORT"를 붙여서 서버로 전송
}
void port_func(char *client_ip, int random_port)
{
    char *addr에 동적 할당 하여 저장 공간 마련
    for(;;)
    {
        if(str의 내용 중에 0D 0A의 값을 찾으면)
            해당 부분을 0A로 치환해서 addr에 저장
        else
            addr에 str의 현재 char를 저장
        if(str 전체를 탐색)
            break;
    }
    return addr
}
void main(int argc, char** argv)
{
    if(argc가 3이 아니면) // 형식에 맞지 않는 것으로 간주하고
        에러문을 화면에 출력하고 return –1
    if(client socket생성하는데 실패하면)
        에러문을 출력하고 return -1
```

```
server socket구조체의 info를 설정

if(client와 server간의 연결에 실패하면)
    에러문을 출력하고 return –1

client의 pid를 받아서 서버로 전송
client의 IPadress를 서버로부터 받음

log_in(sock_fd) // 현재 client의 IP check and 로그인 승인 받기

welcome()내용을 받음

while(1)
{
    사용자로부터 command를 입력받음
    if(그냥 엔터만 쳤을 때)
        continue;
    else
    {
        random_port #을 받음
        if(command가 ls 나 put, get이면)
        {
            각각에 맞는 값으로 put_get을 setting
            data 전송을 위한 소켓 생성
            port_func()
            연결 요청에 대한 reply 받고 출력
            PORT_flag =1;
        }
        else 'non-port'를 서버에 전송

        if(convcomm이 –1을 반환) continue;
        if(변환된 명령어 문자열 CMD 전송에 성공)
        {
            if(port message를 보낸적이 있으면)
                server로부터의 data port connect를 받음
            if(서버로부터 result를 받는데 성공)
            {
```

```
            if(ls 명령어의 결과)
            { data link로 result 받고, 전송 완료 reply도 받음
                얼마나 받았는지를 log에 남김}
            if(put 명령어의 결과)
            {
                보낼 파일 open
                while(1)
                {
                    파일에서 MAX만큼 temp2에 read
                    전송 모드에 따라 remove_rn을 차등 적용해서 temp에 저장
                    전송할 data 크기를 따로 더해서 저장해둠
                    if(더이상 보낼 내용이 없으면) break;
                    temp 내용을 data port를 통해 서버로 전송
                }
                전송에 대한 reply를 받고 출력
                얼마나 보냈는지 출력하고 log에 저장
                data port close
            }
            if(get 명령어의 결과)
            {
                전송 모드에 따라 받은 data를 저장할 파일을 생성 및 open
                while(1)
                {
                    data port로 받은 data를 MAX만큼 temp에 저장
                    if(받은 크기가 0) break;
                    temp 내용을 새로운 파일에 기록
                }
                새 파일 close
                전송에 대한 reply를 받고 출력
                얼마나 보냈는지 출력하고 log에 저장
                data port close
            }
        }
        else break;
    }
else break;
```

```
        }

    }
    서버와 연결된 소켓을 닫음
    return 0
}
```

2. SRV.C

```
void sh_int(int signal)
{
    temp = list의 phead
    for(;;)
    {
        if(temp == ptail)
        {
            temp->pid를 갖는 자식프로세스에 SIGTERM 신호를 보냄
            temp->client_pid를 갖는 프로세스에 SIGTERM 신호를 보냄
            temp노드를 free()
            break;
        }
        temp2 = temp;
        temp->pid를 갖는 자식프로세스에 SIGTERM 신호를 보냄
        temp->client_pid를 갖는 프로세스에 SIGTERM 신호를 보냄
        temp = temp->next
        temp2를 free()
    }
    서버 종료 문구를 출력하고 현재 프로세스에 SIGTERM 신호를 보냄
}

void sh_term(int signal)
{   현재 프로세스 종료}

void sh_chld(int signal)
{
    int 변수 chlid = wait(NULL) // 종료된 프로세스의 pid를 받음
    노드 temp = list의 첫 노드
    for(;;)
    {
        if(현재 노드의 pid 변수가 child와 같으면)
        {
            if(temp가 phead면)
            {
                if(temp가 ptail이면) phead를 NULL로 set
                else          phead를 다음 노드로 이동
```

```
                temp의 next를 NULL로 set하고 temp를 free()
            }
            else if(temp가 ptail이면)
            {
                ptail을 temp2로 이동시키고
                temp를 free()
            }
            else
            {
                temp2의 next를 temp의 next로 설정하고
                temp를 free()
            }
            break;
        }
        노드 temp2 = temp;
        temp = temp->next
    }
    child를 pid로 갖는 process에 SIGTERM을 보냄 // 자원 회수
    list의 child_count변수 --

    현재 자식 프로세스들의 현황을 출력
    timer를 10으로 reset
}

void sh_alrm(int signal)
{
    현재 자식 프로세스들의 현황을 출력
    timer를 10으로 reset
}
```

```
void client_info(struct sockaddr_in* clientaddr)
{
    if(clientaddr의 IP를 string으로 변환할 수 없으면) return –1
    else ip문자열에 clientaddr의 IP를 저장
    if(clientaddr의 port를 int롤 변환할 수 없으면) return –1
    else port에 clientaddr의 port를 저장
    IP와 port를 출력
}

void print_process()
{
    list에 저장된 현재 연결된 client의 수를 출력
    current_time 변수에 현재 시간을 받음.
    노드* temp = list의 phead
    while(1)
    {
        temp의 pid, port와 current_time-temp의 생성시간을 출력
        if(temp가 마지막 노드) break;
        temp = temp - >next
    }
}

void sort_func(char oblist[50][50], int ob_count)
{
        for(a = 0, a <ob_count; a++)
        {
            for(b = 0; b <ob_count-1; b++)
            {
                com1에 oblist[b] 저장
                com2에 oblist[b+1]을 저장
                com1과 com2를 각각 대문자로 치환

                if(com1 > com2)
                    oblist[b]와 oblist[b+1]의 내용을 swap
            }
        }
}
```

```
void get_permi(char oblist[50][50], char* result, int ob_count)
{
    for(a= 0; a< ob_count; a++)
    {
        oblist[a]의 status, uid, gid, localtime을 받음
        if(oblist[a]가 directory) permission 문자열에 d를 붙임
        else          permission 문자열에 –를 붙임
        if(oblist[a]가 user read권한을 소유) permission 문자열에 r를 붙임
        else          permission 문자열에 –를 붙임
        if(oblist[a]가 user write권한을 소유) permission 문자열에 w를 붙임
        else          permission 문자열에 –를 붙임
        if(oblist[a]가 user exec권한을 소유) permission 문자열에 x를 붙임
        else          permission 문자열에 –를 붙임

        if(oblist[a]가 group read권한을 소유) permission 문자열에 r를 붙임
        else          permission 문자열에 –를 붙임
        if(oblist[a]가 group write권한을 소유) permission 문자열에 w를 붙임
        else          permission 문자열에 –를 붙임
        if(oblist[a]가 group exec권한을 소유) permission 문자열에 x를 붙임
        else          permission 문자열에 –를 붙임

        if(oblist[a]가 other read권한을 소유) permission 문자열에 r를 붙임
        else          permission 문자열에 –를 붙임
        if(oblist[a]가 other write권한을 소유) permission 문자열에 w를 붙임
        else          permission 문자열에 –를 붙임
        if(oblist[a]가 other exec권한을 소유) permission 문자열에 x를 붙임
        else          permission 문자열에 –를 붙임

        result에 permission, link수, user name, group name, 최근 접근일,
        이름 순서로 저장
        if(oblist[a]가 directory면) result의 끝에 '/'를 붙임
        result에 개행 문자 입력
    }
}
```

```
void welcome(int sock_fd)
{
    motd 파일을 열어서
    문장을 읽어오고 %를 기준으로 token을 생성
    토큰을 문자열에 저장
    문자열에 what_time()의 내용을 저장
    client에 문자열의 내용을 전송
}
int ip_check(struct sockaddr_in* cliaddr)
{
    입력받은 소켓 구조체의 IPaddress를 문자열에 저장
    그 IP를 .을 기준으로 4분할해서 token으로 만듦
    access.txt를 open
    while(!eof)
    {
        한줄 읽어서 4분할
        if(token1 != *)
        {
            if(token1 != 1번 ip octat)
                continue;
        }
        if(token2 != *)
        {
            if(token1 != 2번 ip octat)
                continue;
        }
        if(token1 != *)
        {
            if(token3 != 3번 ip octat)
                continue;
        }
        if(token1 != *)
        {
            if(token4 != 4번 ip octat)
                continue;
        }
        flag = 1;
```

```
            break;
        }
        액세스 파일 close
        if(flag == 1) return 1;
        else          return 0;

}
int user_match(char *user)
{

        passwd파일 open
        while(!eof)
        {
            한줄 fgetpwent로 받아서 pw로 pointing
            if(입력받은 이름과 matching 되는 pw_name 찾으면)
            {
                flag = 1;
                break;
            }
        }
        passwd 파일 close
        if(flag == 1) return 1;
        else          return 0;
}
int pass_match(char *user, char *password)
{

        passwd파일 open
        while(!eof)
        {
            한줄 fgetpwent로 받아서 pw로 pointing
            if(입력받은 이름과 matching 되는 pw_name 찾으면)
            {
                if(입력받은 password와 matching 되는 pw_passwd 찾으면)
                {
                    flag = 1;
                    break;
```

```
            }
        }
    }
    passwd 파일 close
    if(flag == 1) return 1;
    else          return 0;
}
int log_auth(int connfd, struct sockaddr_in* clientaddr)
{
while(1)
    {

        client로부터 ID 포함된 message를 각각 token으로 분류

        if(!strcmp(token1, "USER"))
        {
            token2 = ID
            if(user_match에 ID 넣은 결과가 1)
            {
                331 reply를 클라이언트로 전송
                client로부터 passwd가 포함된 message 받음
                passwd를 token1으로 분류

                if(!strcmp(token1,"PASS"))
                {
                    n에 pass_match(token1)한 결과를 반환
                    if(n == 1)
                    {
                        230 reply를 client로 전송
                        그 내용을 log에 남김
                        break;
                    }
                }

            }
            else // Illegal user case
            {
```

```
                미승인 유저의 아이디를 log에 남김
            }

            if(3회 이상 틀리면)
            {
                530 reply를 client에게 전송
                return 0;
            }

            430 reply를 client에게 전송
            count++;
        }
    }
    return 1;

}
char* convert_str_to_addr(char *str, unsigned int *port)
{
    char* addr에 동정 할당해서 저장 공간 마련
    str을 ,을 기준으로 token 4개에 분류
    addr에 분할된 token들을 저장
    남은 str을 ,을 기준으로 token 2개에 분류
    token1 에 256을 곱해서 head에 저장
    token2를 tail에 저장
    head에 tail을 더하고 port가 head를 가리키게 함
    return addr
}
char* what_time()
{
    char* time에 동적할당
    tm t에 localtime을 받음
    t의 localtime 속성들을 이용해 ctime에 현재 시각 저장
    return ctime
}
```

char* remove_rn(char *)
- client의 것과 동일

void main(int argc, char argv)**
{
    linked list인 list 동적 할당
    logfile open

    if(server socket 생성에 실패)
    { 에러문 출력하고 return –1}

    server socket 구조체의 info입력
    if(server socket을 address와 bind하는데 실패)
    { 에러문 출력하고 return –1}

    if(listen에 실패)
    { 에러문 출력하고 return –1}

    서버 시작을 log에 기록

    while(1)
    {
        client를 accept
        client로부터 client_pid를 받음
        client에게 client의 IP address를 전송
        client가 승인된 IP인지확인
        if(승인된 IP)
        {
            IP, PORT#를 출력하고 client에게 ₩n을 전송
        }
        if(비승인 IP)
        {
            431 code를 클라이언트에게 전송 후 connect를 close함
            continue;
        }

```
if(log_auth()가 0을 반환)
{
    클라이언트 연결 close하고 continue;
}
welcome(클라이언트)

pid = fork()
if(pid < 0 )
{ 에러문 출력하고 continue}
else if(pid == 0) // child process
{
    클라이언트로부터 require message를 받음
    if(message가 PORT메세지면)
    {
        data_trans_flag를 1로 셋
        data port용 소켓을 만들어서 client와 connect

        if(connect실패) 550reply를 클라이언트 전송 후 break;
        200 reply를 클라이언트 전송
    }
    else ₩n을 클라이언트에 전송(의미 없는 내용)

    if(MKD, RMD, DELE command 일 때)
    {
        명령어 뒤의 argument들을 oblist에 저장
        그 개수를 ob_count에 저장
    }
    else
    {
        strtok를 통해 token2, token3를 얻음.
    }

        if(명령어가 NLST)
        {
            150 reply를 클라이언트에 전송
            if(token2 == NULL)
            {
```

```
    현재 디렉토리를 open
    while(현재 디렉토리의 모든 파일에 대하여)
    {
        첫글자가 '.'이 아닌 이름만 oblist에 저장
        ob_count ++
    }
    sort_func(oblist, ob_count) // 솔팅
    oblist의 내용 출력
    if(5개 출력했으면) 개행
}


if(token2 != NULL  && token3 == NULL)
{

if(token2 == '-a')
{
    현재 디렉토리를 open
    while(현재 디렉토리의 모든 파일에 대하여)
    {
        대상의 이름을 oblist에 저장
        ob_count ++
    }
    sort_func(oblist, ob_count) // 솔팅
    oblist의 내용 출력
    if(5개 출력했으면) 개행
}

else if(token2 == '-l')
{
    현재 디렉토리를 open
    while(현재 디렉토리의 모든 파일에 대하여)
    {   첫글자가 '.'이 아닌 이름만 oblist에 저장
        ob_count ++
    }
    sort_func(oblist, ob_count) // 솔팅
    get_permi(oblist, result, ob_count)
```

```
}
else if(token2 == '-al')
{
    현재 디렉토리를 open
    while(현재 디렉토리의 모든 파일에 대하여)
    {   모든 대상의 이름을 oblist에 저장
        ob_count ++
    }
    sort_func(oblist, ob_count) // 솔팅
    get_permi(oblist, result, ob_count)
}
else
{
    if(열 수 없는 경로일 경우)
    {에러문을 result에 저장}
    else
    {
        token2의 경로 opendir
        while(현재 디렉토리의 모든 파일에 대하여)
        {
            첫글자가 '.'이 아닌 이름만 oblist에 저장
            ob_count ++
        }
        sort_func(oblist, ob_count) // 솔팅
        oblist의 내용 출력
        if(5개 출력했으면) 개행
    }
}


else if(token2 && token3 != NULL)
{
if(token3가 열 수 없는 경로)
{에러문을 result에 저장}
else // 위의 조건에서 이미 폴더가 열림
{
if(token2 == '-a')
```

```
    {
    while(현재 디렉토리의 모든 파일에 대하여)
    {
        대상의 이름을 oblist에 저장
        ob_count ++
    }
    sort_func(oblist, ob_count) // 솔팅
    oblist의 내용 출력
    if(5개 출력했으면) 개행
    }

    else if(token2 == '-l')
    {
    while(현재 디렉토리의 모든 파일에 대하여)
    {   첫글자가 '.'이 아닌 이름만 oblist에 저장
        ob_count ++
    }
    sort_func(oblist, ob_count) // 솔팅
    get_permi(oblist, result, ob_count)
    }

    else if(token2 == '-al')
    {
        while(현재 디렉토리의 모든 파일에 대하여)
        {   모든 대상의 이름을 oblist에 저장
            ob_count ++
        }
        sort_func(oblist, ob_count) // 솔팅
        get_permi(oblist, result, ob_count)
    }
    else;

    }
result크기 측정해서 data_size에 저장
data port로 result 전송
일반 port로 226 reply 전송 및 log에 기록
data port close
```

```
}

else if(명령어가 LIST)
{
    if(token2 == NULL)
    {
    현재 디렉토리를 opendir
    while(현재 폴더내의 모든 대상에 대해)
    {oblist에 그 이름을 저장, ob_count++}

    sort_func(oblist, ob_count)
    get_permi(oblist, result, ob_count)
    }
    else
    {
    if(token2가 열수 없는 경로)
    {에러문을 result에 저장}
    else
    {
    while(현재 폴더내의 모든 대상에 대해)
    {oblist에 그 이름을 저장, ob_count++}

    sort_func(oblist, ob_count)
    get_permi(oblist, result, ob_count)

    }

    }
}
else if(명령어가 CWD)
{
if(token2 == NULL)
{
    uid를 얻어서 temp에 '/home/'
        + user name 형식으로 저장
    temp로 chdir()
```

```
    result에 250 reply 저장
}
else if(token2가 열수 없는 경로)
{result에 550 reply 저장}
else // 위의 else if 조건문에서 chdir을 사용하면서
    // 경로가 바뀌었음.
{
    result에 250 reply 저장
}
}
else if(명령어가 CDUP)
{
    chidr("..")
    if(경로이동 실패) result에 550 reply 저장
    else              result에 250 reply 저장
}
else if(명령어가 PWD)
{
    result에 250 reply + 현재 위치 저장
}
else if(명령어가 MKD)
{
    umask(0)
    for(oblist의 모든 이름들에 대해)
    {
        if(mkdir(oblist[a], 0744) <0)
        { 실패한 대상의 이름과 550 reply를 result에 저장
          flag = 1;
        }
    }
    if(flag == 0)
    {250 reply를 result에 저장}
}
else if(명령어가 RMD)
{
    for(oblist의 모든 이름들에 대해)
    {
```

```
            if(rmdir(oblist[a]) <0)
            { 실패한 대상의 이름과 550 reply를 result에 저장
              flag = 1;
            }
        }
        if(flag == 0)
        {250 reply를 result에 저장}
}
else if(명령어가 DELE)
{
        for(oblist의 모든 이름들에 대해)
        {
            if(remove(oblist[a]) <0)
            { 실패한 대상의 이름과 550 reply를 result에 저장
              flag = 1;
            }
        }
        if(flag == 0)
        {250 reply를 result에 저장}
}
else if(명령어가 rename)
{ 현재 디렉토리 opendir
 for(현재 디렉토리 내 모든 대상에 대하여)
 {
    if(동일한 이름을 찾으면) flag = 1;
 }
 if(flag == 0) result에 550 reply 저장
 else
 {
    client에게 350 reply 전송
    client로부터 바꿀 이름 받음
    token3 = strtok(NULL, " ") // 바꿀 이름
    if(rename(token2, token3) < 0)
        result에 550 reply 저장
    else
        result에 250reply 저장
 }
```

```
}
else if(명령어가 quit)
{
    client에게 221 reply 전송
    exit(0);
}
else if(명령어가 TYPE)
{
    if(token2가 A) asc_bin = 1; 201 reply를 client에게 전송
    if(token2가 I) asc_bin = 0; 201 reply를 client에게 전송
    else              502 reply를 client에게 전송
}
else if(명령어가 STOR)
{
    mode에 따라 150 reply를 client에게 전송
    전송한 내용 log에 저장
    mode에 따라서 새로운 파일 생성
    while(1)
    {
        data port를 통해 data를 받아옴
        if(data 크기가 0) break;
        새로 생성한 파일에 data 기록
    }
    파일 close
    226reply 전송 및 로그에 기록
    data port close
}
else if(명령어가 RETR)
{
    mode에 따라 150 reply를 client에게 전송
    전송한 내용 log에 저장
    token2의 이름을 갖는 파일 open
    while(1)
    {
        data를 MAX만큼 temp2에 읽어옴
        mode에 따라 remove_rn을 차등 적용해 temp에 저장
```

```
                    if(temp 크기가 0) data port에 temp 전송 /  break;
                    보낼 데이터 크기 측정해서 따로 저장 및 add
                    data port를 통해 temp 전송
                }
                파일 close
                226reply 전송 및 로그에 기록
                data port close
            }
            else;
            if(data port를 이용하는 command 아닌 경우)
            {
                기존의 링크를 통해서 result를 전송 및 로그에 기록
            }

        }
        else
        {
            client_info(&client_addr) // 자식 프로세스의 info를 출력
            자식 프로세스의 pid를 출력

            node를 하나 동적 할당하고 자식 프로세스의 pid, port,
            노드 생성 시간(현재 시간), 연결된 client의 pid를 저장

             list에 node 추가
            list의 child_count++
            print_process()// 자식 프로세스 현황 출력
            timer를 10초로 리셋
        }
        client와의 연결 종료
        return 0
    }

}
```

■ Result

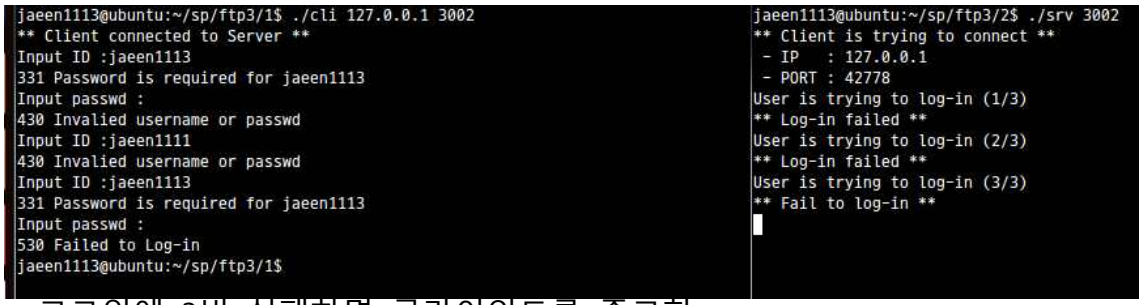
- 사전 setting

1) client

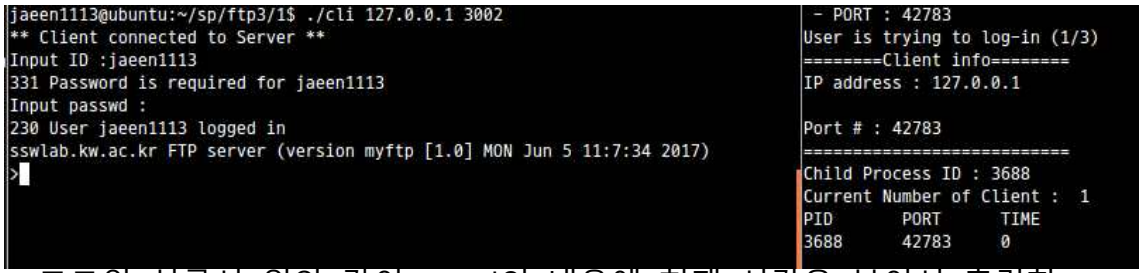
2) server

- 위와 같이 서버와 클라이언트를 다른 위치에 두고 시작

-



- 로그인에 3번 실패하면 클라이언트를 종료함



- 로그인 성공시 위와 같이 motd의 내용에 현재 시각을 붙여서 출력함

```
>pwd
257 "/home/jaeen1113/sp/ftp3/2" is current directory.
>cd ..
250 CWD command succeeds.
>pwd
257 "/home/jaeen1113/sp/ftp3" is current directory.
>mkdir 123
250 MKD command performed successfully.
>rmdir 123
250 RMD command performed successfully.
>
```

```
Current Number of Client :  1
PID       PORT      TIME
3688      42783     80
Current Number of Client :  1
PID       PORT      TIME
3688      42783     90
Current Number of Client :  1
PID       PORT      TIME
3688      42783     100
RMD 123  [3688]
```

-일반 명령어 사용 시 위와 같은 reply를 서버로부터 받는다.

```
jaeen1113@ubuntu:~/sp/ftp3/1$ ./cli 127.0.0.1 4000
** Client connected to Server **
Input ID :jaeen1113
331 Password is required for jaeen1113
Input passwd :
230 User jaeen1113 logged in
sswlab.kw.ac.kr FTP server (version myftp [1.0] MON Jun 5 11:15:46 2017)
>ls
200 Port command is Successful.
150 Opening data connection for directory list.
access.txt       logfile          motd             passwd           srv

226 Complete transmission
86 bytes is received
>
```

```
jaeen1113@ubuntu:~/sp/ftp3/2$ ./srv 4000
** Client is trying to connect **
 - IP   : 127.0.0.1
 - PORT : 60773
User is trying to log-in (1/3)
========Client info========
IP address : 127.0.0.1

Port # : 60773
===========================
Child Process ID : 3802
Current Number of Client :  1
PID       PORT      TIME
3802      60773     0
PORT 127,0,0,1,78,51
NLST  [3802]
```

- PORT 명령을 받는 ls의 PORT message와 변환된 명령어를 오른쪽 서버에서 볼 수 있으며, 새로 만들어진 data port를 통해 data를 받은 것을 확인할 수 있음.

```
226 Complete transmission
86 bytes is received
>bin
201 Type set to I
>get access.txt
200 Port command is Successful.
150 Opening binary mode connection for access.txt.
226 Complete transmission
OK. 7 bytes is received.
>
```

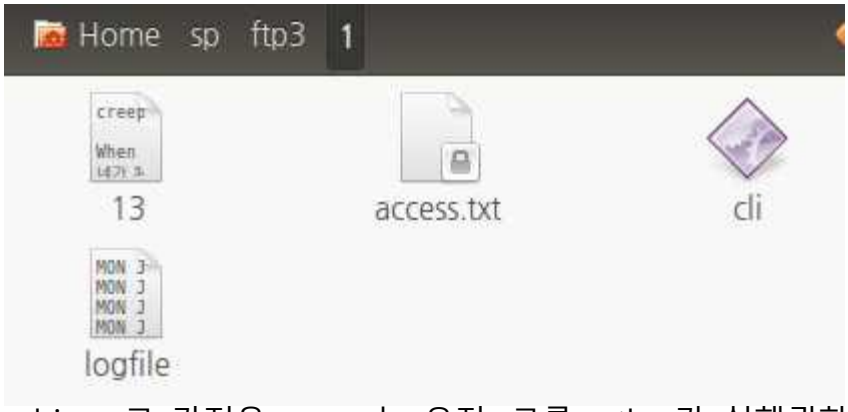
```
TYPE I [3802]
Current Number of Client :  1
PID       PORT      TIME
3802      60773     80
Current Number of Client :  1
PID       PORT      TIME
3802      60773     90
Current Number of Client :  1
PID       PORT      TIME
3802      60773     100
PORT 127,0,0,1,74,61
RETR access.txt [3802]
```

- 서버에 있는 access.txt 파일을 가져오는 get 명령어. binary mode

```
>type ascii
201 Type set to A
>get logfile
200 Port command is Successful.
150 Opening ascii mode connection for logfile.
226 Complete transmission
OK. 4538 bytes is received.
```

```
non-port
TYPE A [3802]
Current Number of Client :  1
PID       PORT      TIME
3802      60773     180
PORT 127,0,0,1,39,200
RETR logfile [3802]
```

- 서버에 있는 logfile을 가져오는 get. ascii mode



- binary로 가져온 access는 유저, 그룹, other가 실행권한만 있음
- ascii로 가져온 logfile은 노멀함

- binary mode로 13 파일을 put함

- 전송 결과 . binary mode

- ascii mode로 13file put

- ascii mode로 전송한 결과

- bye bye....

■ 결론 및 고찰

이번 과제가 기존에 진행해왔던 Assignment 들에 여러 가지 기능을 추가하는 과제였었던 만큼 알고리즘 상으로는 큰 문제가 없었다고 생각한다. 다만 write/read의 짝을 안맞춰 주게 되는 경우가 발생하면 끊임없이 그저 기다리고만 있는 상태가 지속되는 것이 굉장히 답답하였고 찾아내기 힘들었던 기억이 난다. 지금까지 찍어보았던 printf를 이 보고서에 쓴다면 현재 50쪽을 보고 있는 것이 70쪽으로 늘어나지 않을까 싶다. 그리고 data포트를 뚫어 줘야 하는 부분에서 코드가 상당히 더럽게 짜여 졌던 기억이 난다. 뭔가 read/write 짝을 맞춰주려고 의미 없는 문장을 전송하기도 했었고, 이번 과제에서 한가지 아쉬웠던 점은 get, put의 transfer mode에 대해서 뭔가 설명이 조금 부족한 느낌이었다. 그래도 못 알아먹었으면 구현을 못했겠죠? 사실 잘 구현했습니다. 친구랑 충분히 토의하고 나서요 아 그리고 조교님께서 pdf대로만 하라고 하셨기 때문에, ls와 비슷한 동작을 수행하는 dir 명령어도 data port가 필요하다고 생각은 했으나 구현은 안했습니다. 나중에 알아챘거든요. ㅎㅎ..... 아무튼 채점하실 때 참고해주시구요.
한 학기동안 고생 많으셨습니다. 감사합니다♡