시스템 프로그래밍 실습

9차 과제

실습 일시 : 화 1,2
담당 교수님 : 김태석 교수님
학번 : 2013722095
이름 : 최재은
실습 번호 : Assignment #2 FTP2

FTP1

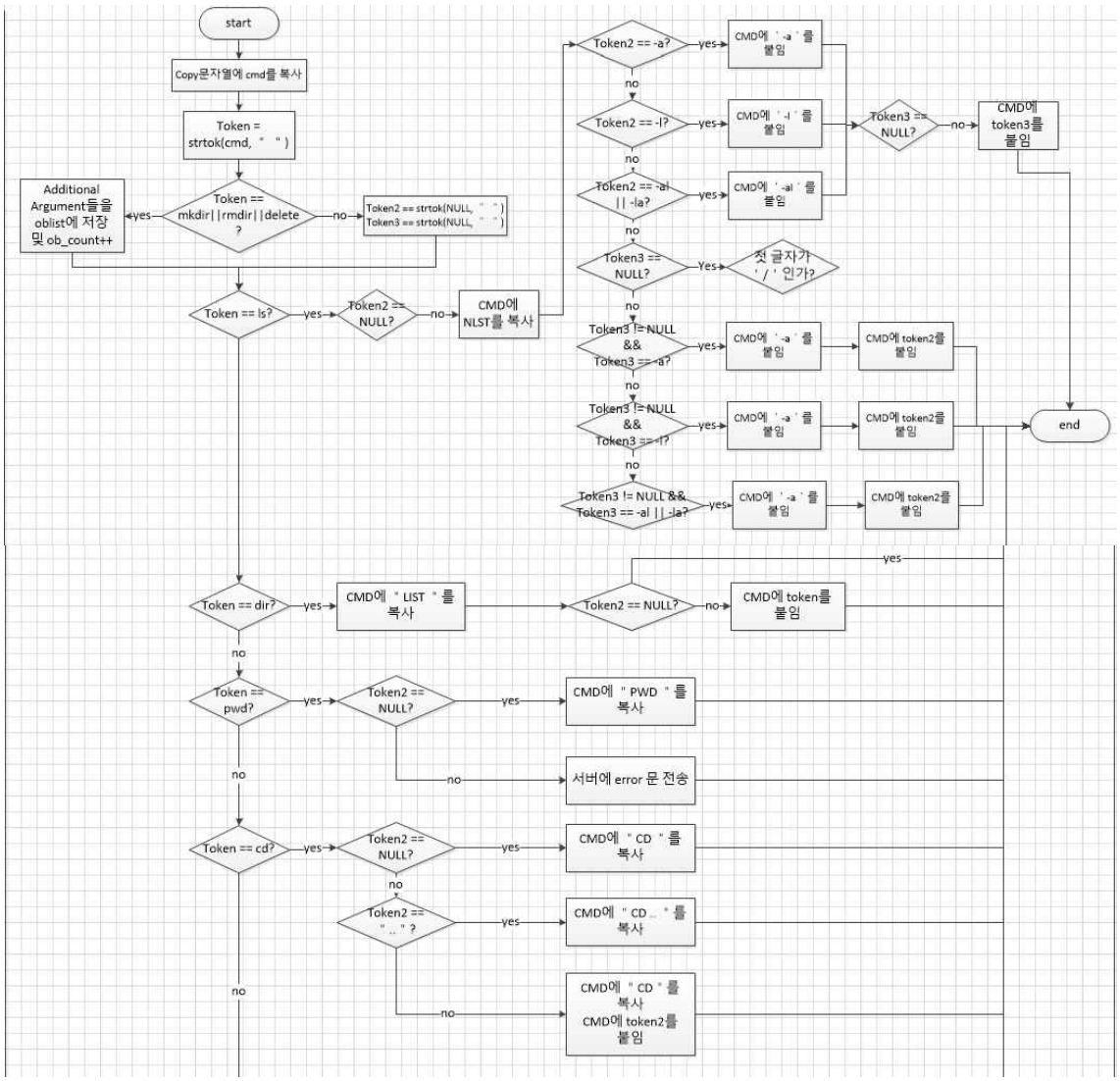
■ Introduction

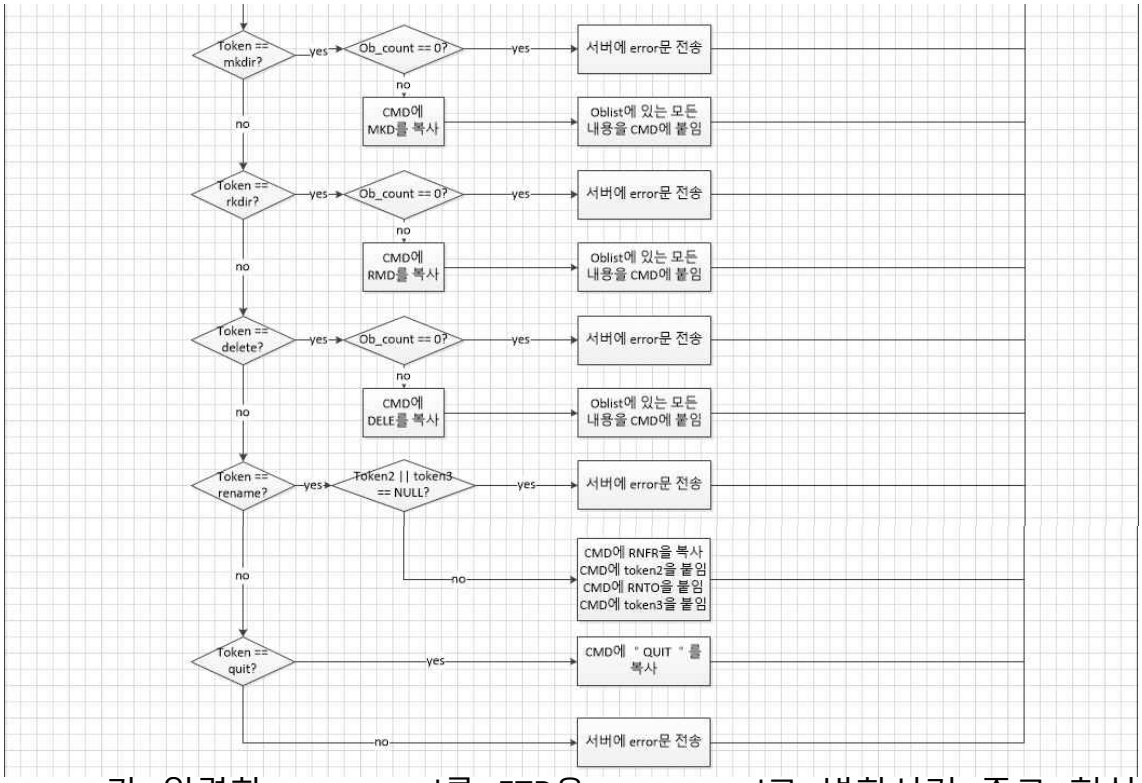
Assignment #1에서 Tee 통신을 통해서 구현했던 command ftp client/server를 응용하여 소켓을 생성하고 각자의 소켓을 통해 데이터를 주고받는 형태의 client/server network를 구축해 본다. 또한 각종 signal을 활용해보고 그 signal의 handler를 만들어 주어 여러 가지 동작을 수행해 본다.

■ Flow Chart

1. cli.c

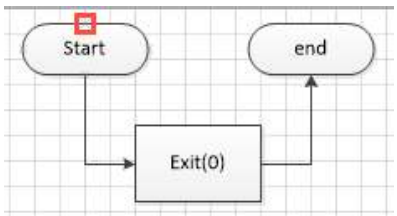
1) convcomm





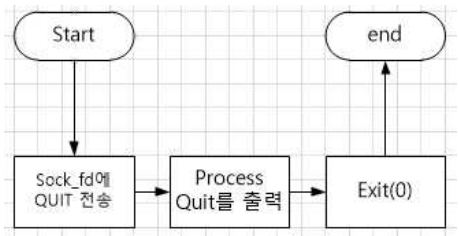
- user가 입력한 command를 FTP용 command로 변환시켜 주고 형식에 맞게 재정렬 해주는 함수

2) sh_term



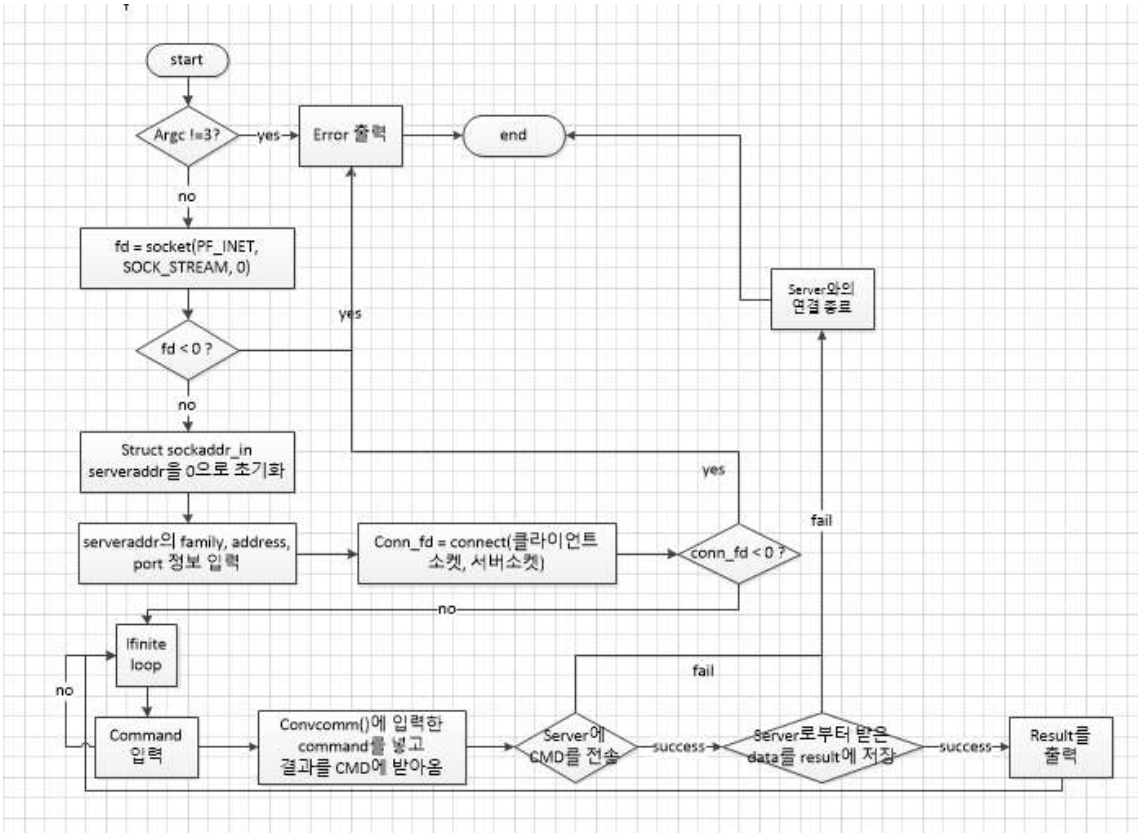
- SIGTERM signal handler

3) sh_int



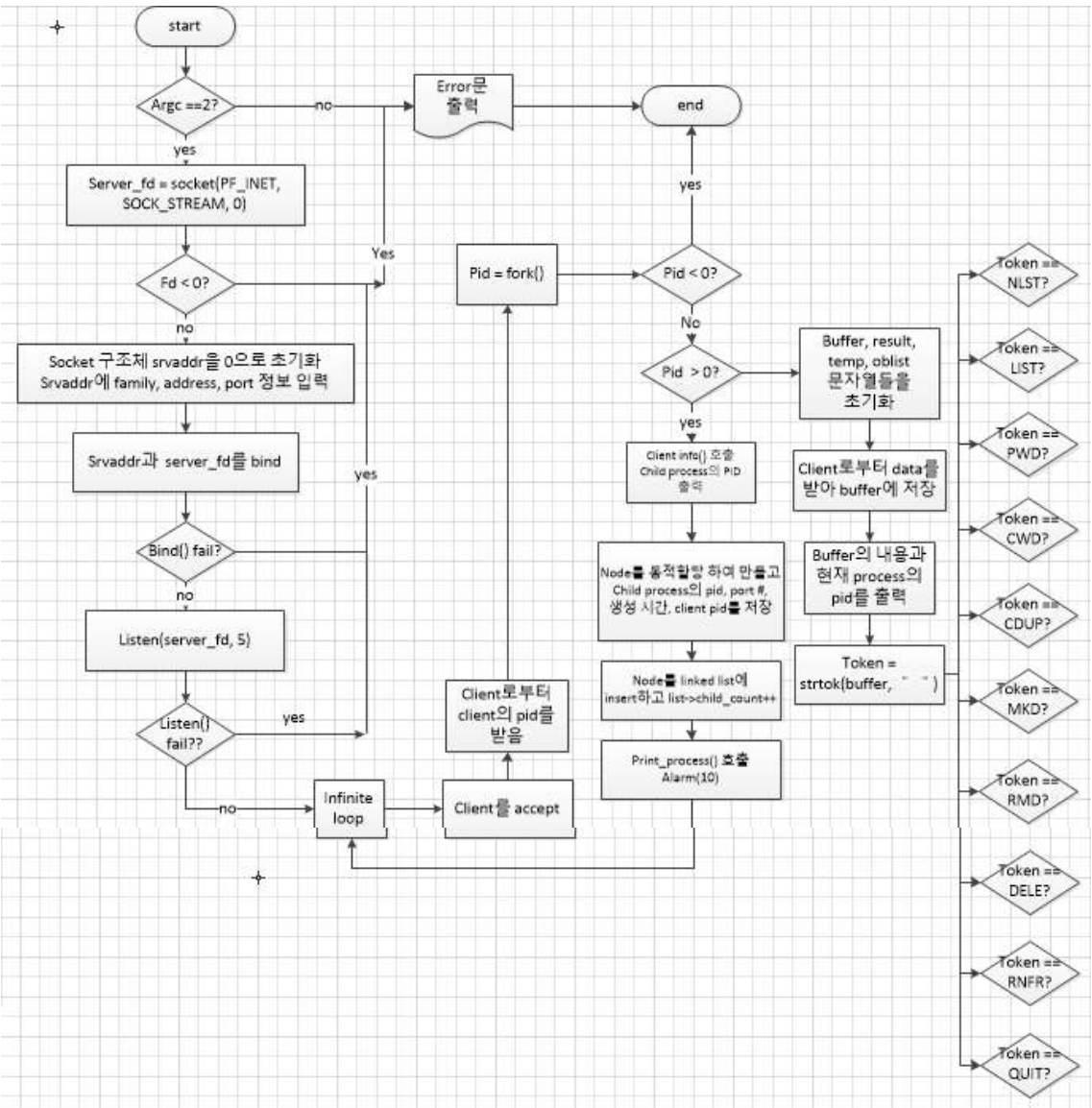
- SIGINT signal handler

4) client 전체 동작

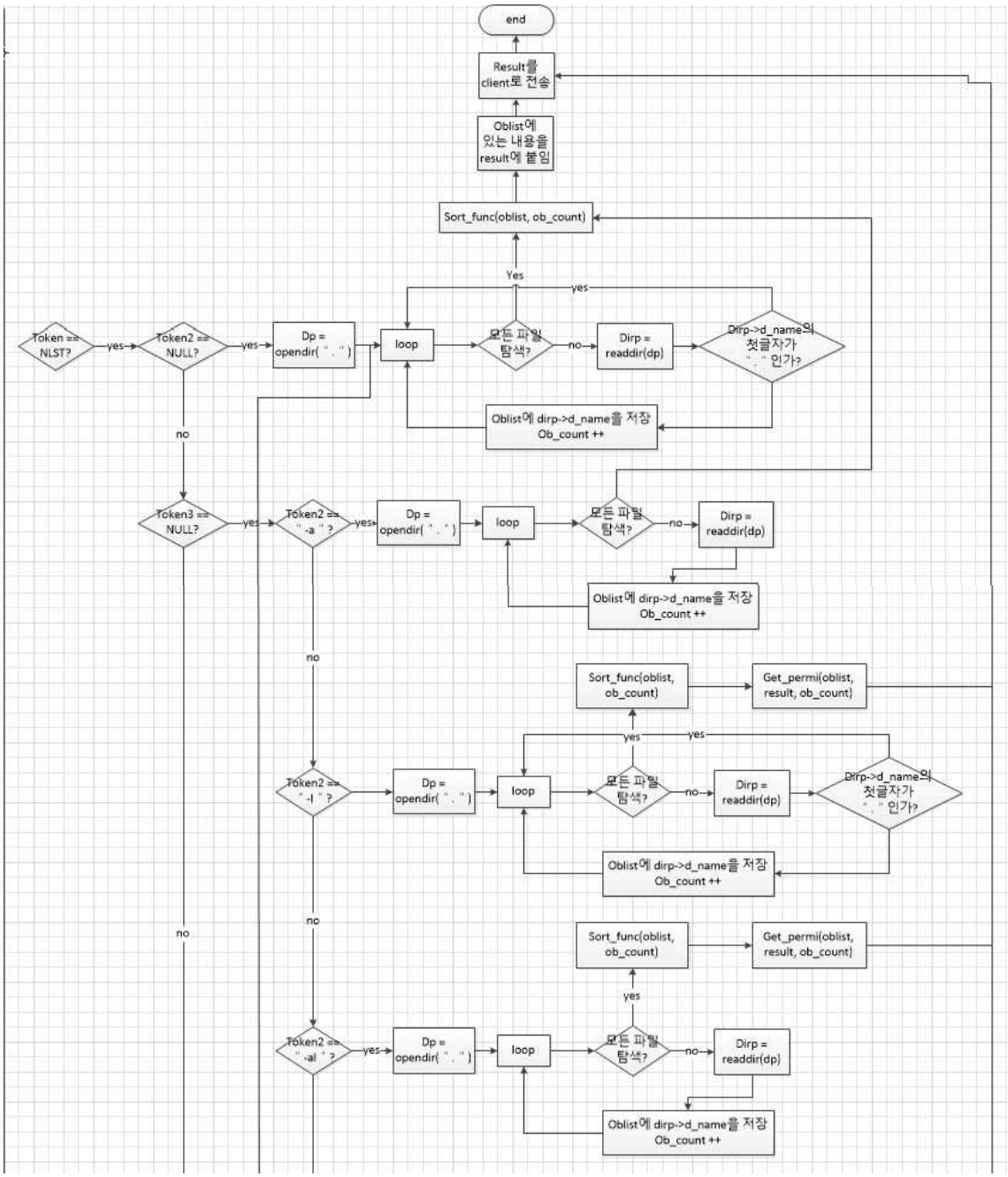


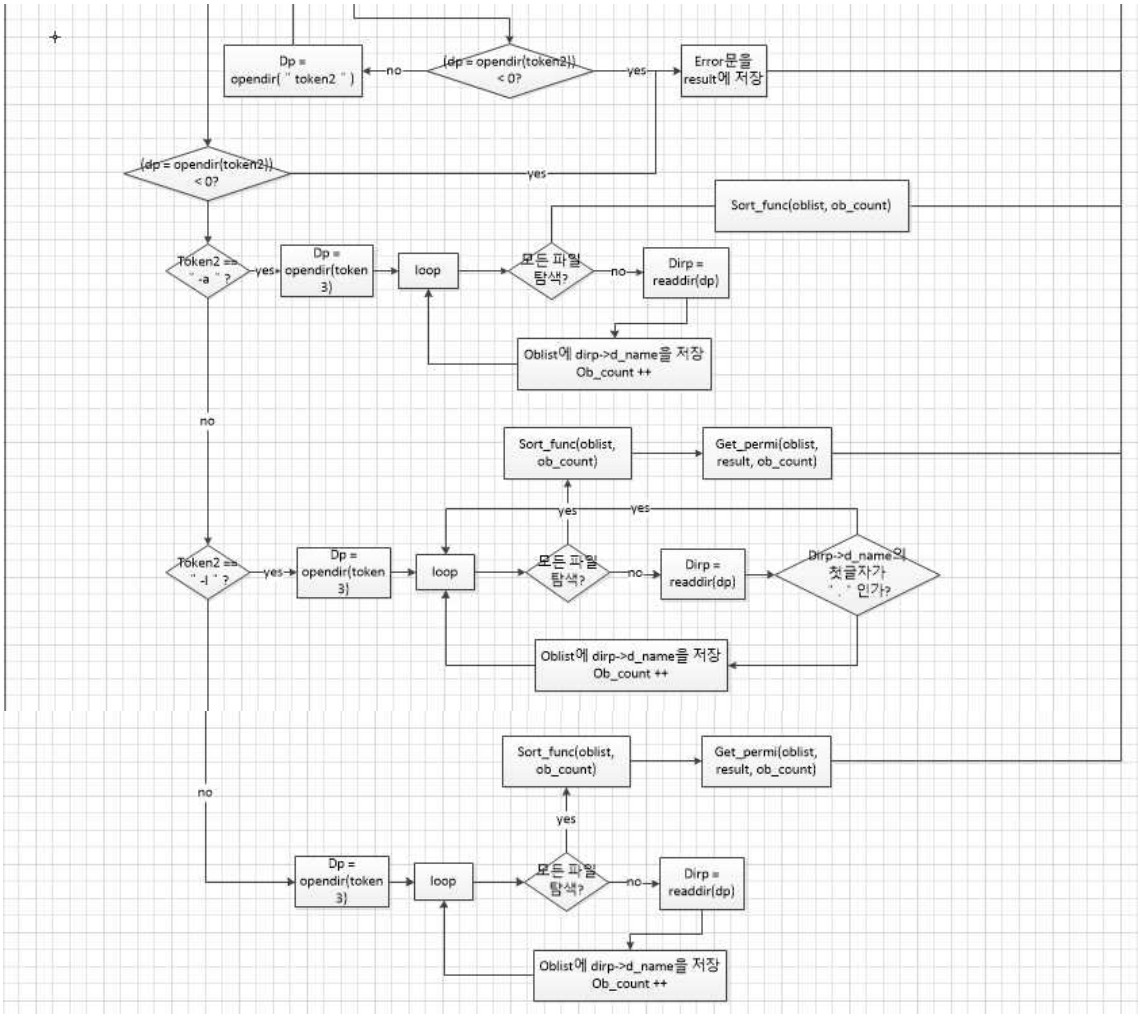
2. srv.c

1) 각 command진입 전까지

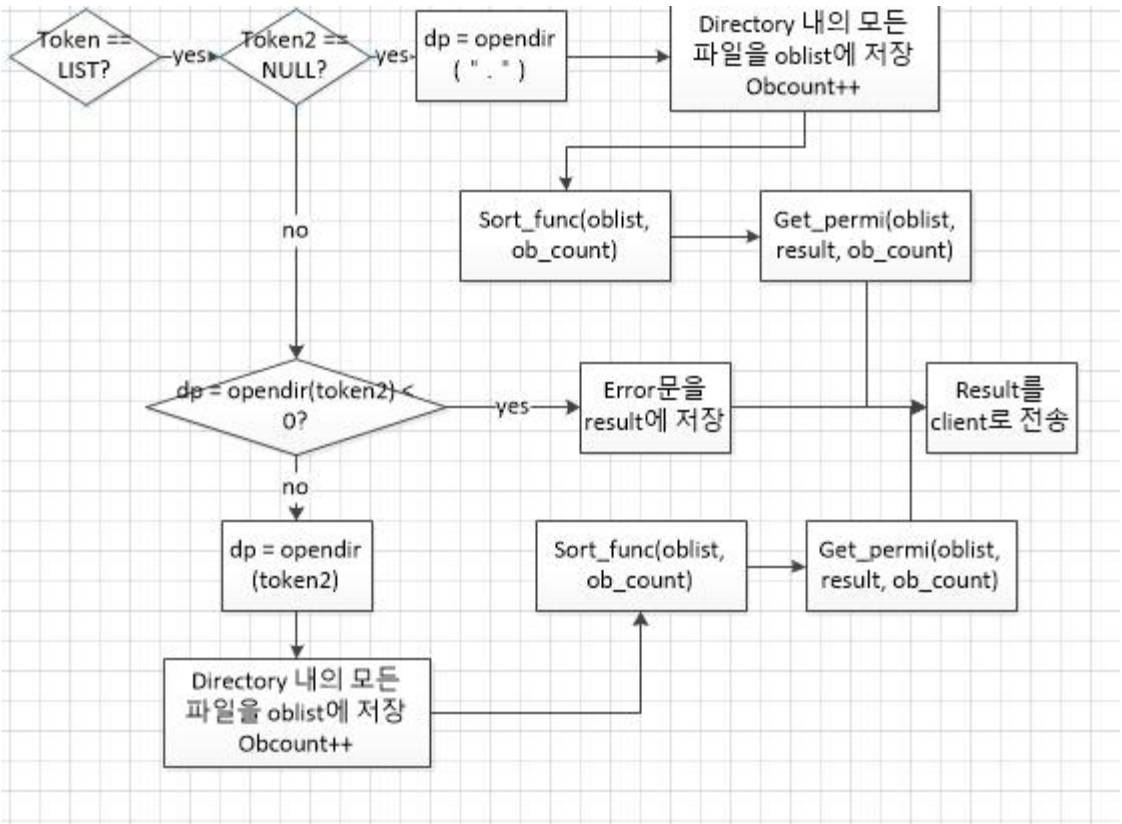


*NLST

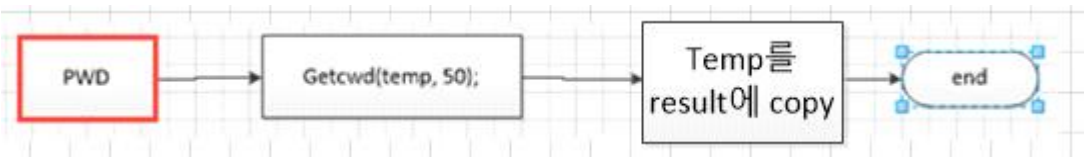

end
Result를 client로 전송
Oblist에 있는 내용을 result에 붙임
Sort_func(oblist, ob_count)
Yes
yes
Token == NLST?
yes
Token2 == NULL?
yes
Dp = opendir(" . ")
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Dirp->d_name의 첫글자가 " . " 인가?
Oblist에 dirp->d_name을 저장 Ob_count ++
no
Token3 == NULL?
yes
Token2 == " -a " ?
yes
Dp = opendir(" . ")
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 ++
no
Sort_func(oblist, ob_count)
Get_permi(oblist, result, ob_count)
yes
Dp = opendir(token 3)
loop
모든 파일 탐색?
no
Dirp = readdir(dp)
Oblist에 dirp->d_name을 저장
Ob_count ++

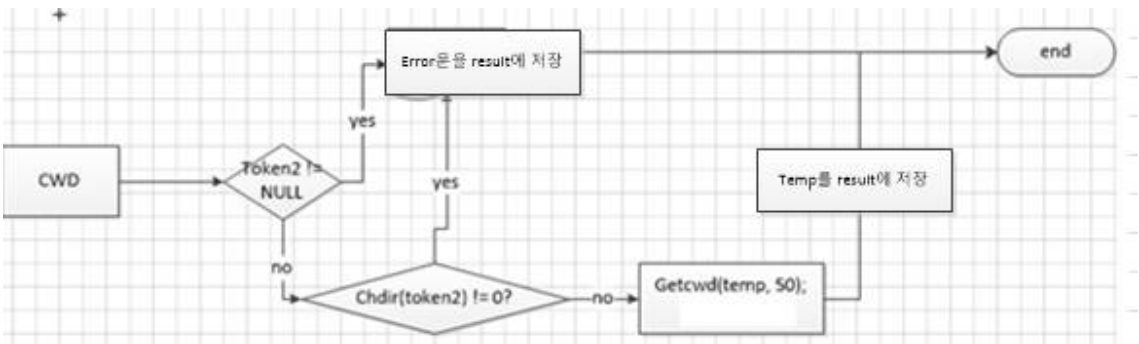
*LIST


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++

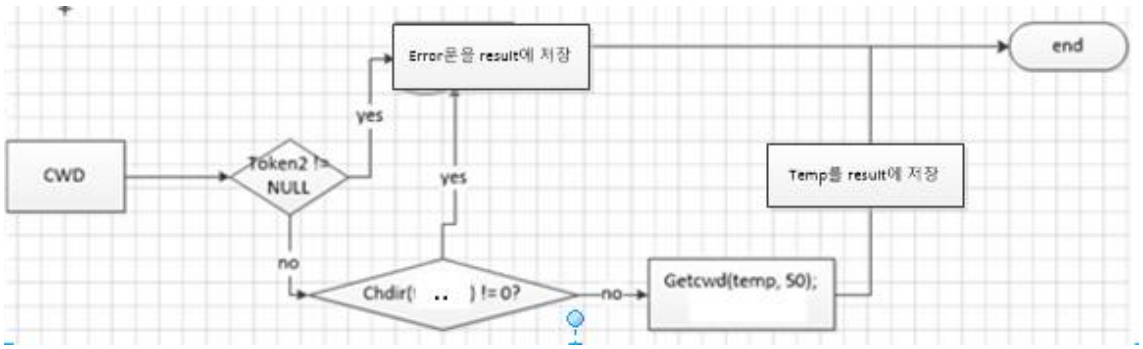
*PWD


PWD
Getcwd(temp, 50);
Temp를 result에 copy
end

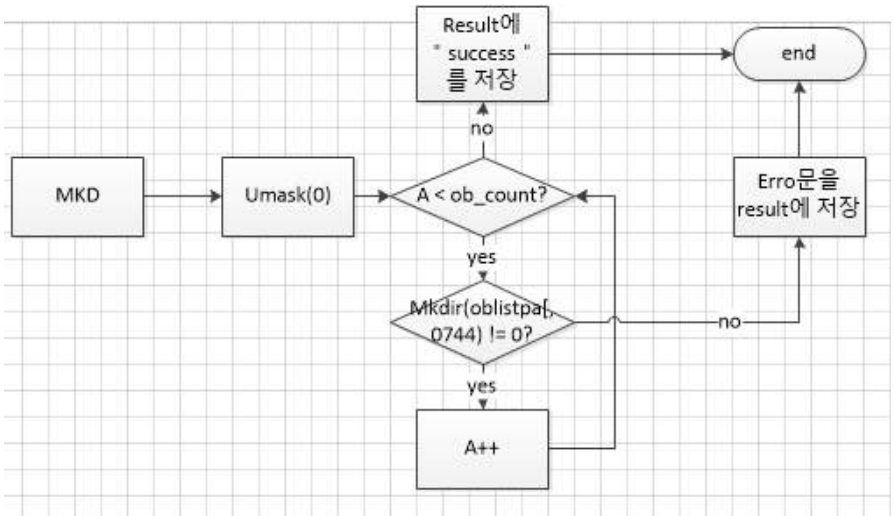
*CWD


CWD
Token2 != NULL
yes
Error문을 result에 저장
end
yes
Temp를 result에 저장
no
Chdir(token2) != 0?
no
Getcwd(temp, 50);

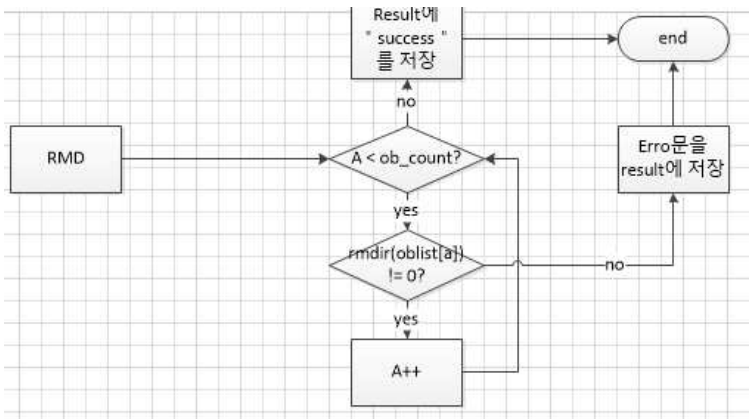
*CDUP


Error문을 result에 저장
end
yes
CWD
Token2 != NULL
yes
Temp를 result에 저장
no
Chdir(..) != 0?
no
Getcwd(temp, 50);

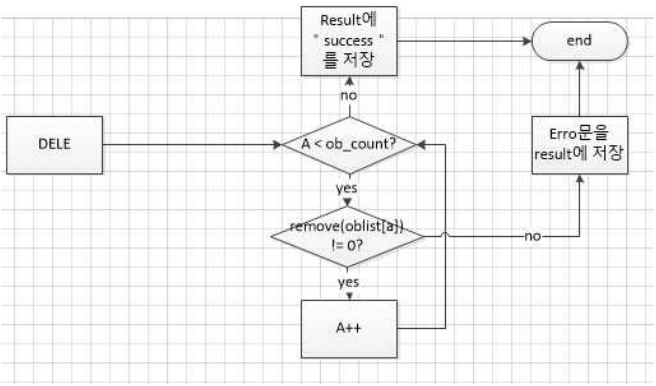
*MKD


Result에 " success " 를 저장
end
no
MKD
Umask(0)
A < ob_count?
Erro문을 result에 저장
yes
Mkdir(oblistpa[, 0744) != 0?
no
yes
A++

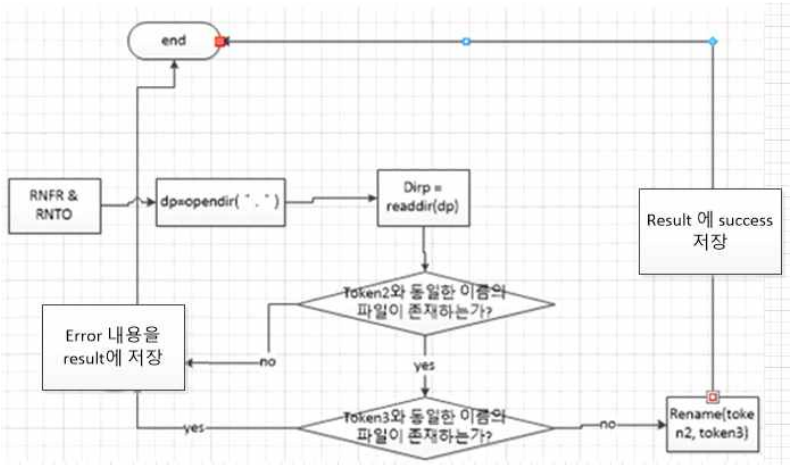
*RMD


Result에 " success " 를 저장
end
no
RMD
A < ob_count?
Erro문을 result에 저장
yes
rmdir(oblist[a]) != 0?
no
yes
A++

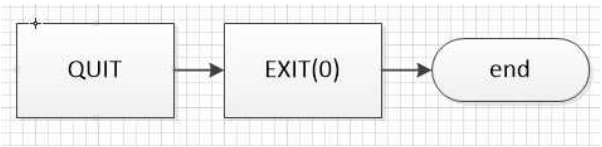
*DELE



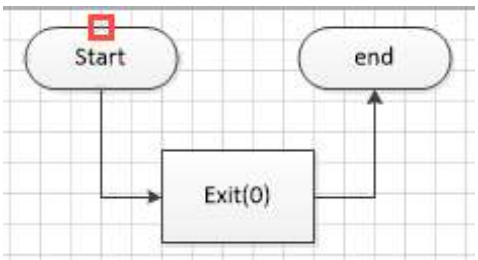
*RNFR & RNTO



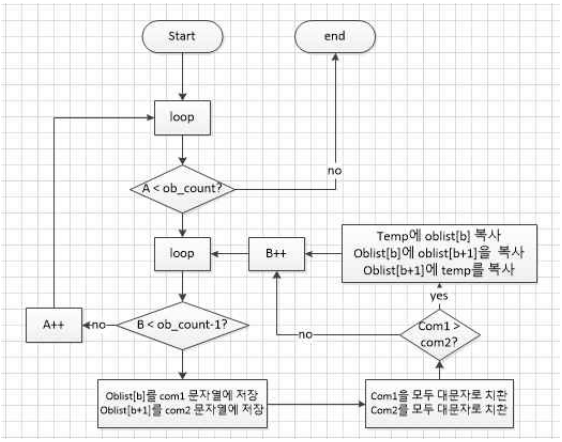
*QUIT



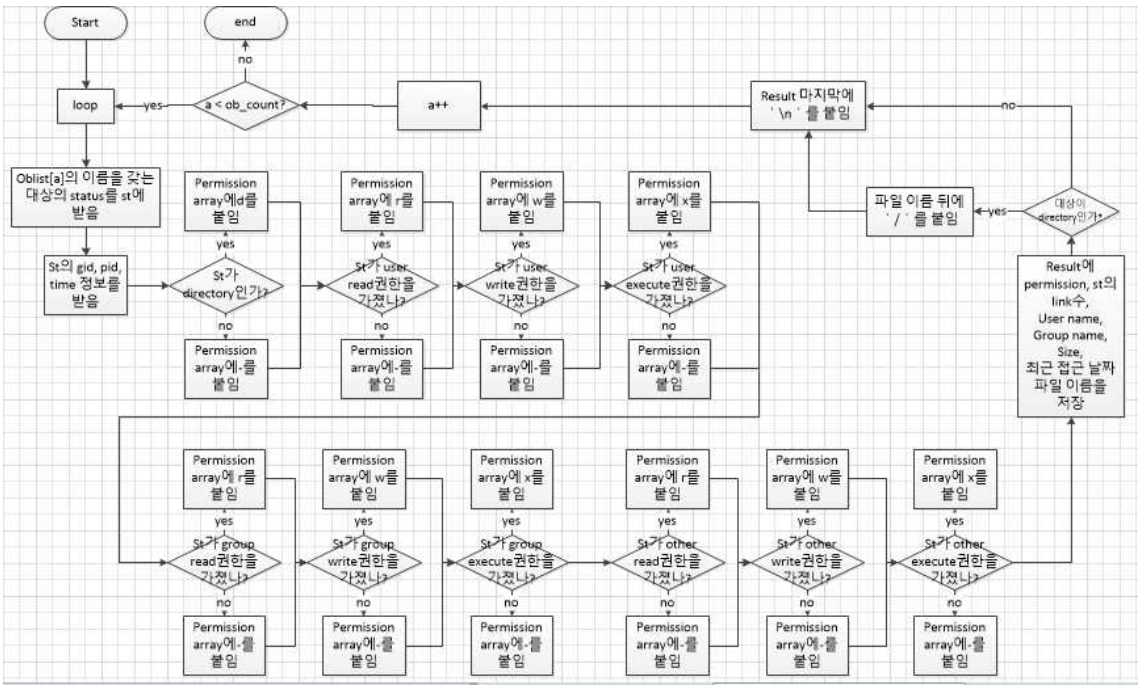
2) sh_term



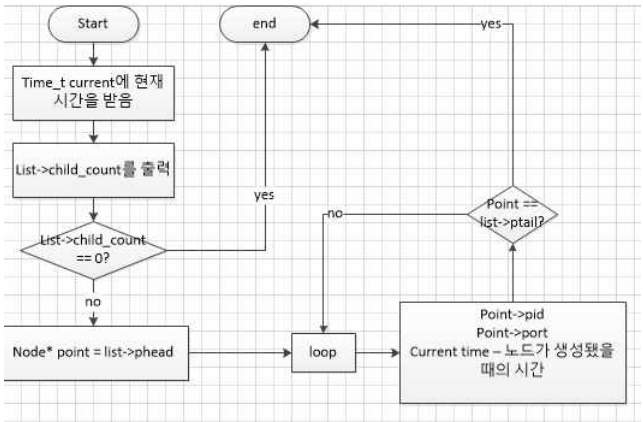
3) sort_func



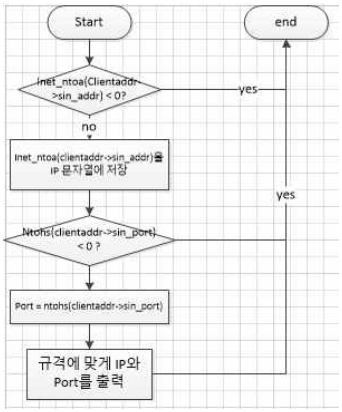
4) get_permi



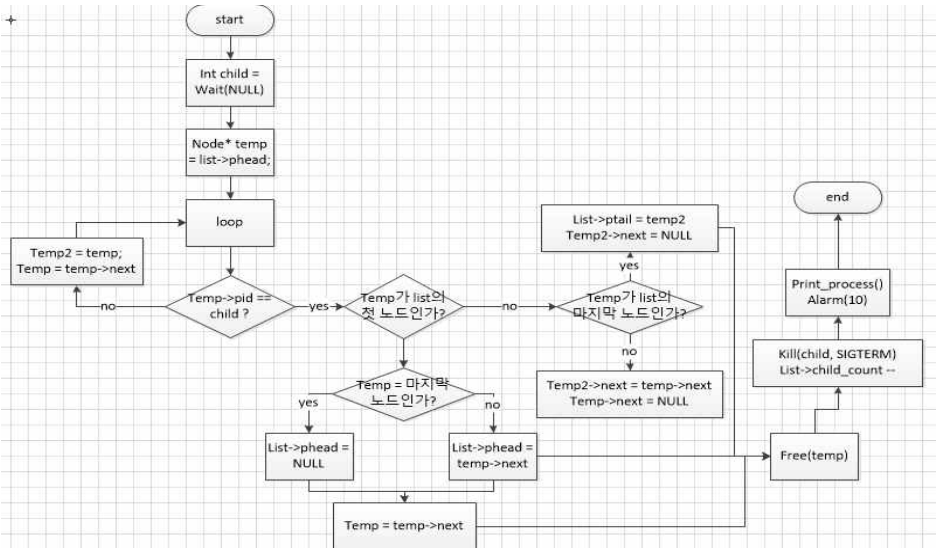
5) printf_process



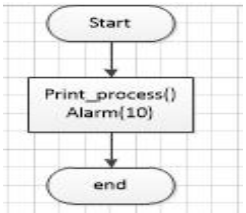
6) client_info



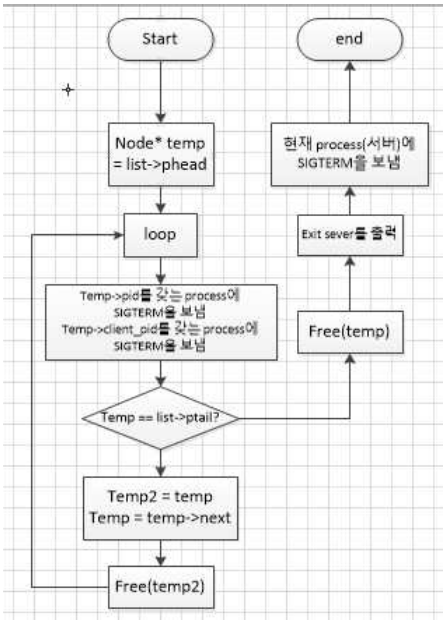
7)sh_chld



8) sh_alrm



9)sh_int


Start
end
Node* temp
= list->phead
현재 process(서버)에
SIGTERM을 보냄
loop
Exit sever를 출력
Temp->pid를 갖는 process에
SIGTERM을 보냄
Temp->client_pid를 갖는 process에
SIGTERM을 보냄
Free(temp)
Temp == list->ptail?
Temp2 = temp
Temp = temp->next
Free(temp2)

■ 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
```

```
        {
        현재 cmd창에 에러문 출력
        서버로 에러문 전송 return -1
        }

        return 0
}

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

        while(1)
        {
                사용자로부터 command를 입력받음
                if(그냥 엔터만 쳤을 때)
                        continue;
                else
                {


                                        if(convcomm(cmd, CMD)가 –1을 반환하면)
                                                continue;
                                        if(server로 메시지 전송에 성공)
                                        {
                                                if(server로부터 message 받기 성공)
                                                        받은 내용 출력
```

```
                                        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 main(int argc, char** argv)
{
        linked list인 list 동적 할당
        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}

        while(1)
        {
                client를 accept
                client로부터 client_pid를 받음
                pid = fork()
                if(pid < 0 )
                { 에러문 출력하고 continue}
                else if(pid == 0) // child process
                {
                        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)
                                {
                                        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;
}


}
}
```

```
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()
        현재 경로 출력
}
else if(token2가 열수 없는 경로)
```

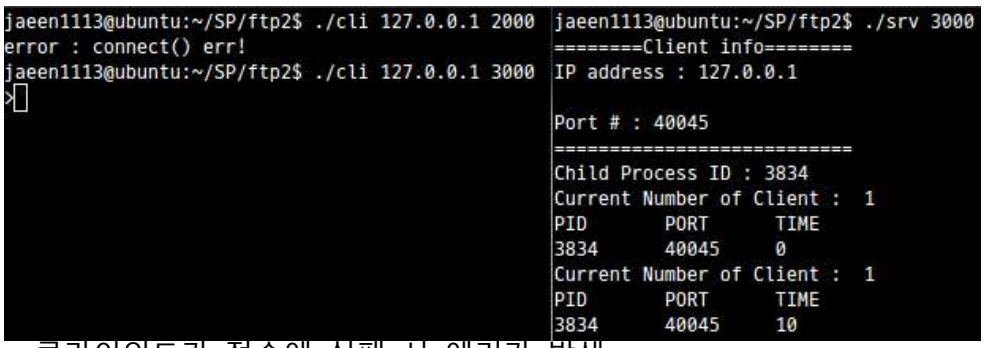
```
{에러문을 result에 저장}
else // 위의 else if 조건문에서 chdir을 사용하면서
        // 경로가 바뀌었음.
{
        현재 경로를 result에 저장
}
}
else if(명령어가 CDUP)
{
        chidr("..")
        현재 경로를 result에 저장
}
else if(명령어가 PWD)
{
        현재 경로를 result에 저장
}
else if(명령어가 MKD)
{
        umask(0)
        for(oblist의 모든 이름들에 대해)
        {
                if(mkdir(oblist[a], 0744) <0)
                { 실패한 대상의 이름을 result에 저장
                  flag = 1;
                }
        }
        if(flag == 0)
        {result에 success 저장}
}
else if(명령어가 RMD)
{
        for(oblist의 모든 이름들에 대해)
        {
                if(rmdir(oblist[a]) <0)
                { 실패한 대상의 이름을 result에 저장
                  flag = 1;
                }
```

```
        }
        if(flag == 0)
        {result에 success 저장}
}
else if(명령어가 DELE)
{
        for(oblist의 모든 이름들에 대해)
        {
                if(remove(oblist[a]) <0)
                { 실패한 대상의 이름을 result에 저장
                  flag = 1;
                }
        }
        if(flag == 0)
        {result에 success 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 에러문 저장
 else
 {
        token3 = strtok(NULL, " ") // 바꿀 이름
        if(rename(token2, token3) < 0)
                result에 에러문 저장
        else
                result에 Success 저장
 }

}
else if(명령어가 quit)
{
        exit(0);
else;
```

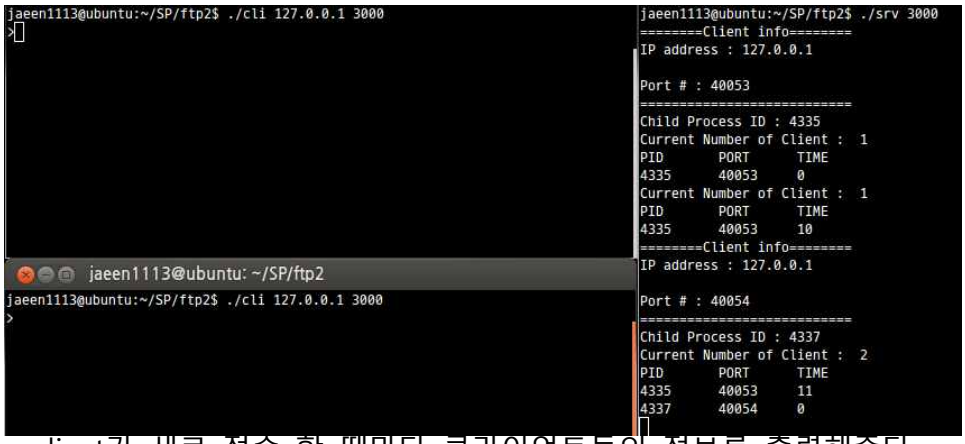
클라이언트로 result를 전송

```
                }
                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
        }

}
```

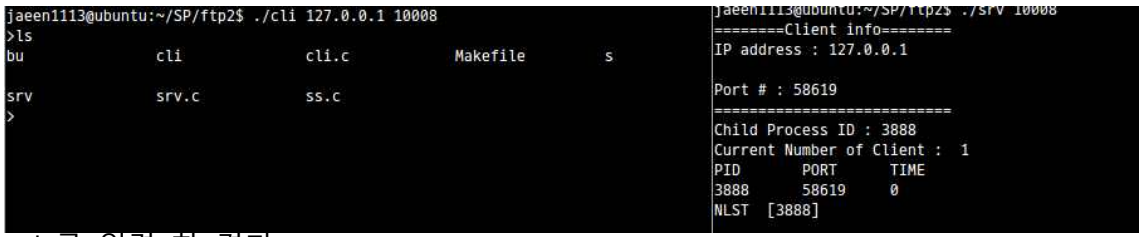
■ Result



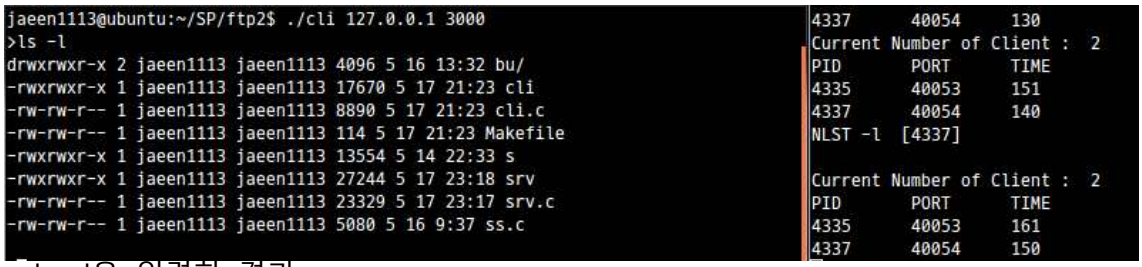
- 클라이언트가 접속에 실패 시 에러가 발생
- 최초 접속 시 현재 클라이언트 수를 서버에서 보여주고 10초마다 클라이언트 내용들을 보여준다.



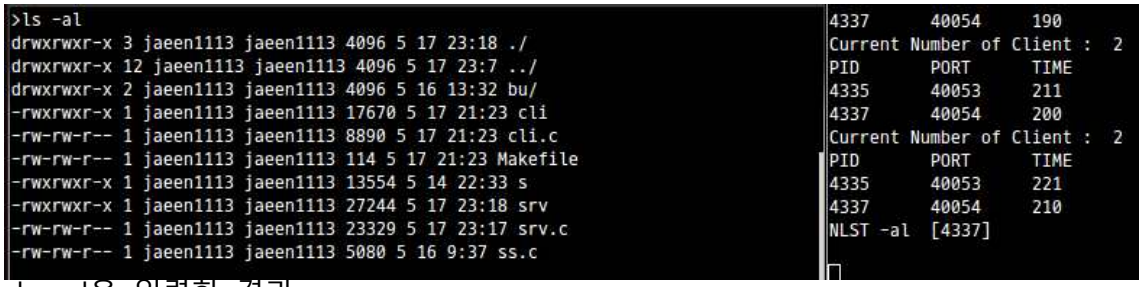
- client가 새로 접속 할 때마다 클라이언트들의 정보를 출력해준다.



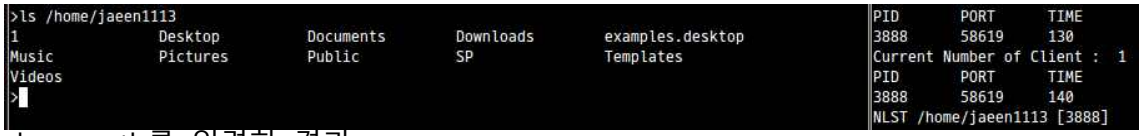
- ls를 입력 한 결과.
- server에서는 변환된 command와 생성된 자식 process의 pid가 출력된다.



- ls –l을 입력한 결과



-ls –al을 입력한 결과



-ls + path를 입력한 결과

-ls + option + path를 입력한 결과

-없는 경로를 입력하면 위와 같은 에러를 클라이언트로 보낸다.

-모르는 옵션에 대하여 에러문을 출력

- pwd를 입력

-cd .. 를 입력

-cd를 입력

```
>cd /home/jaeen1113/Downloads
/home/jaeen1113/Downloads<- current working directory
```

- cd + path를 입력

```
>dir
drwxr-xr-x 3 jaeen1113 jaeen1113 4096 5 17 23:31 ./
drwxr-xr-x 30 jaeen1113 jaeen1113 4096 5 17 23:7 ../
drwxrwxrwx 4 jaeen1113 jaeen1113 4096 5 2 11:16 1/
-rwxrw-rw- 1 jaeen1113 jaeen1113 8093299 3 14 10:36 D2Coding-1.2.zip
-rwxr-xr-x 1 jaeen1113 jaeen1113 7887448 3 16 14:31 D2Coding.ttc
-rw------- 1 jaeen1113 jaeen1113 7746994 5 2 9:47 sublime-text_build-3126_amd64.deb
```

- dir를 입력

```
>dir /home
drwxr-xr-x 3 jaeen1113 jaeen1113 4096 5 17 23:31 ./
drwxr-xr-x 30 jaeen1113 jaeen1113 4096 5 17 23:7 ../
drwxr-xr-x 30 jaeen1113 jaeen1113 4096 5 17 23:7 jaeen1113/
>
```

-dir + path를 입력

```
jaeen1113@ubuntu:~/1$ ls
1  11.c  a  b  cli.c
>rename 11.c 22.c
Success!
jaeen1113@ubuntu:~/1$ ls
1  22.c  a  b  cli.c
```

- rename command

```
>mkdir apple banana orange
Success!
jaeen1113@ubuntu:~/1$ ls
1  22.c  a  apple  b  banana  cli.c  orange
```

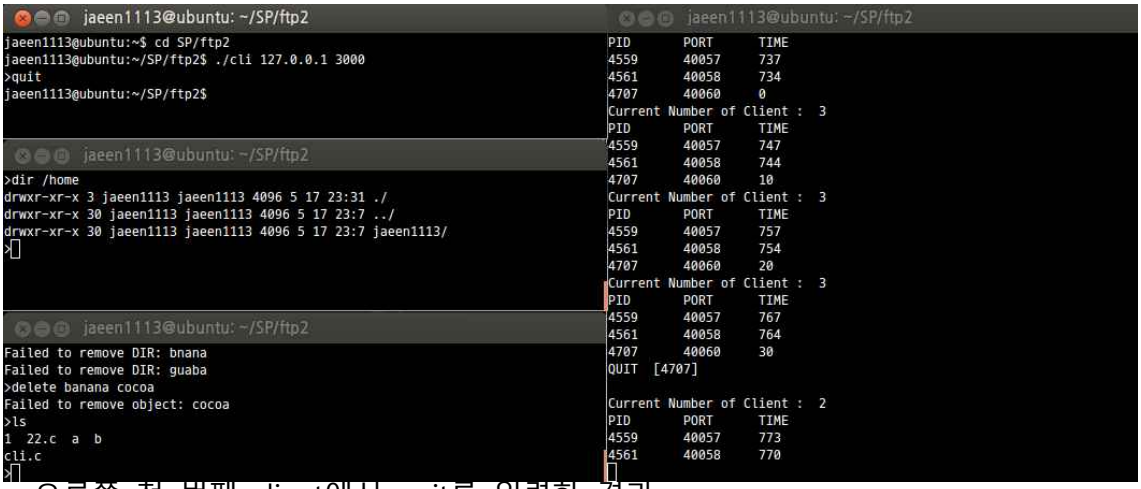
- mkdir command

```
>rmdir apple bnana orange guaba
Failed to remove DIR: bnana
Failed to remove DIR: guaba
jaeen1113@ubuntu:~/1$ ls
1  22.c  a  b  banana  cli.c
```

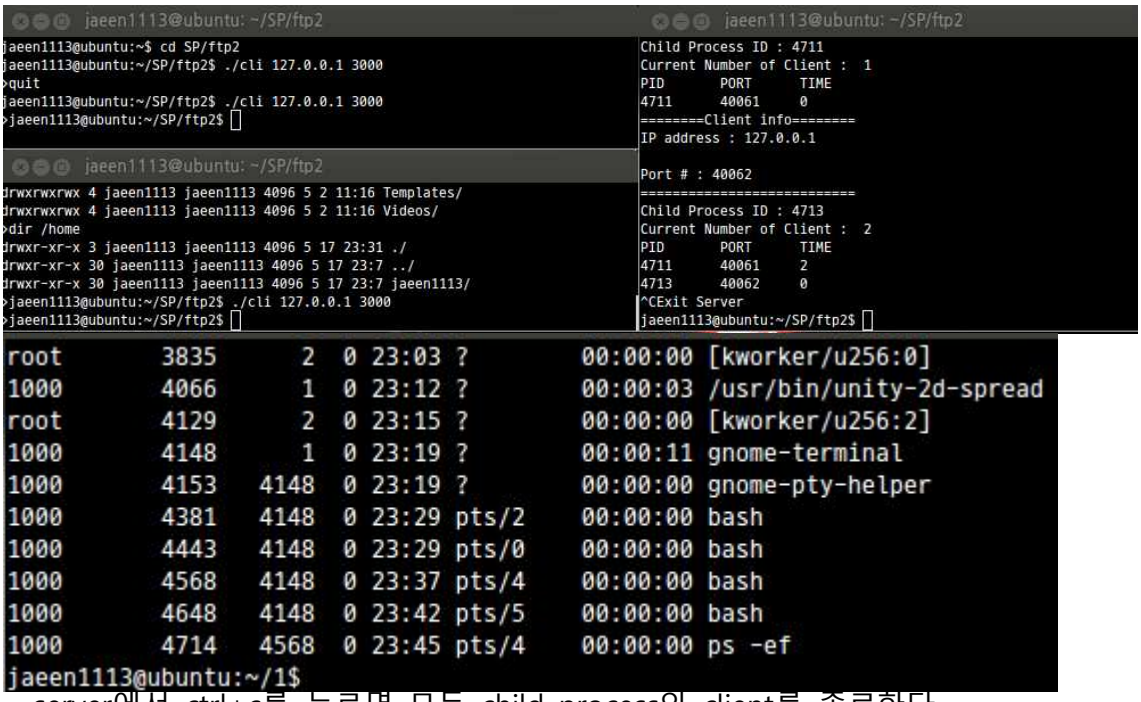
- rmdir command

```
>delete banana cocoa
Failed to remove object: cocoa
jaeen1113@ubuntu:~/1$ ls
1  22.c  a  b  cli.c
```

- delete command



- 오른쪽 첫 번째 client에서 quit를 입력한 결과



- server에서 ctrl+c를 누르면 모든 child process와 client를 종료한다.
- 두 번째 결과를 보면 자원 회수를 잘 하였음을 확인할 수 있다.

■ 결론 및 고찰

- 위의 과제를 진행하면서 컴퓨터 네트워크 시간에 배웠던 end system의 통신을 직접 구현해 봄으로써 좀더 구체적인 이해가 되었다. 다만 이번 과제 질문 중에 서버에서 ctrl+c를 눌러 서버를 종료하면 클라이언트들의 커맨드 창이 기본 shell로 돌아가야 한다고 하셨는데, 실제 서버와 클라이언트의 시스템을 생각해보면 그 둘은 서로 다른 pc인데 서버가 터졌다고 클라이언트가 자동으로 종료되는 것은 이상하다고 생각한다.