시스템 프로그래밍 실습

Assignment2-3

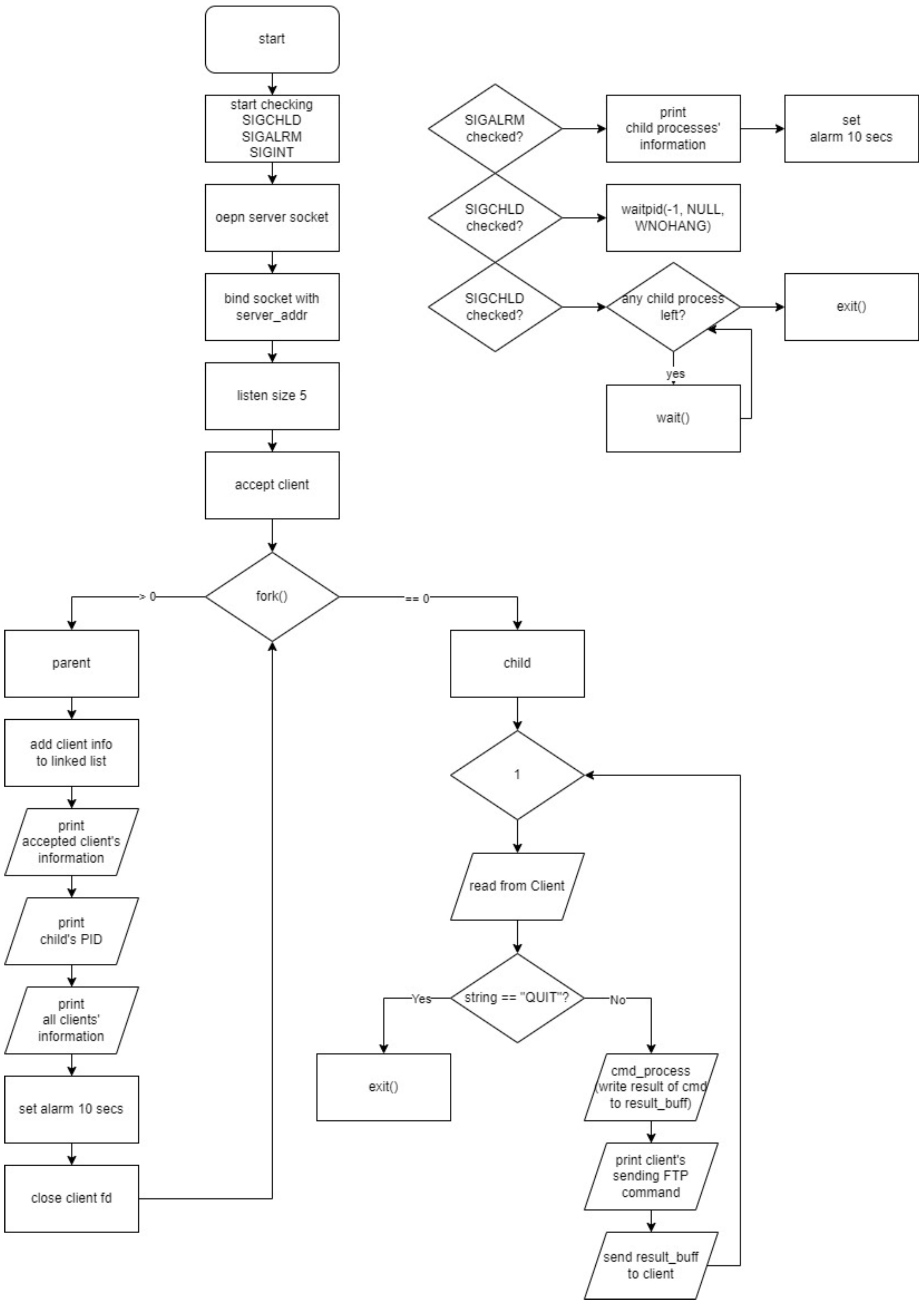
Class	:	금 1, 2 분반
Professor	:	최상호 교수님
Student ID	:	2020202031
Name	:	김재현

Introduction

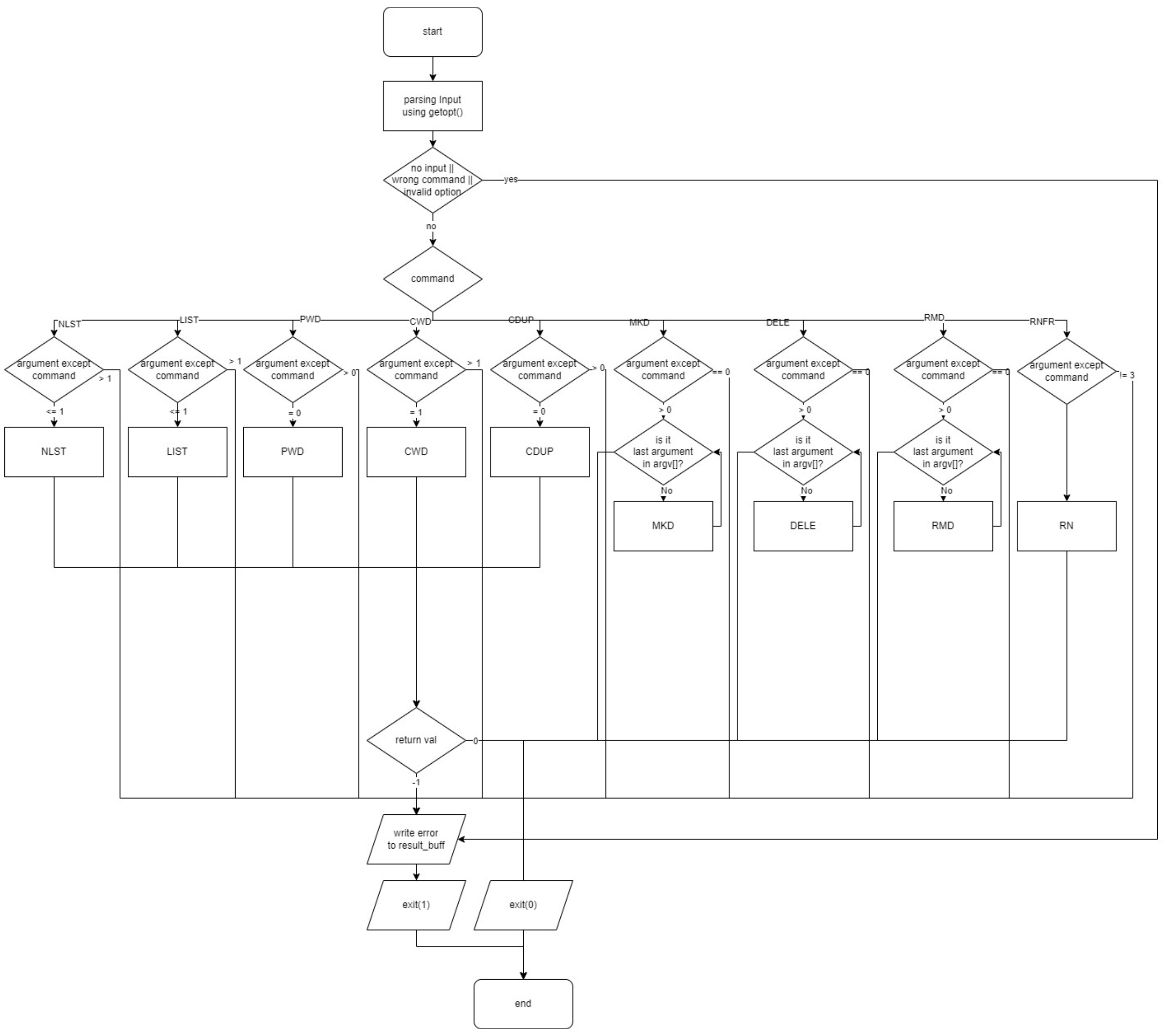
제 2-2 에서는 Socket Programming 과 fork()를 활용하여, 다수의 Client 와 1 대 다수 통신을 구현했습니다. User 로부터 문자열을 입력 받아 Server 로 보내고, Server 에서 Client 로부터 입력 받은 문자열을 그대로 다시 Client 로 보내도록 구현했습니다. 이번 과제에서는 1 대 다수 통신에서 assignment 1 에서는 만든 server client 간의 FTP 명령어 변환 및 동작들을 적용시킵니다. 따라서 다수의 Clients 로부터 받은 명령어들을 server 에서 실행하여 결과를 다시 각 Client 로 보내주는 것을 구현합니다.

Flow chart

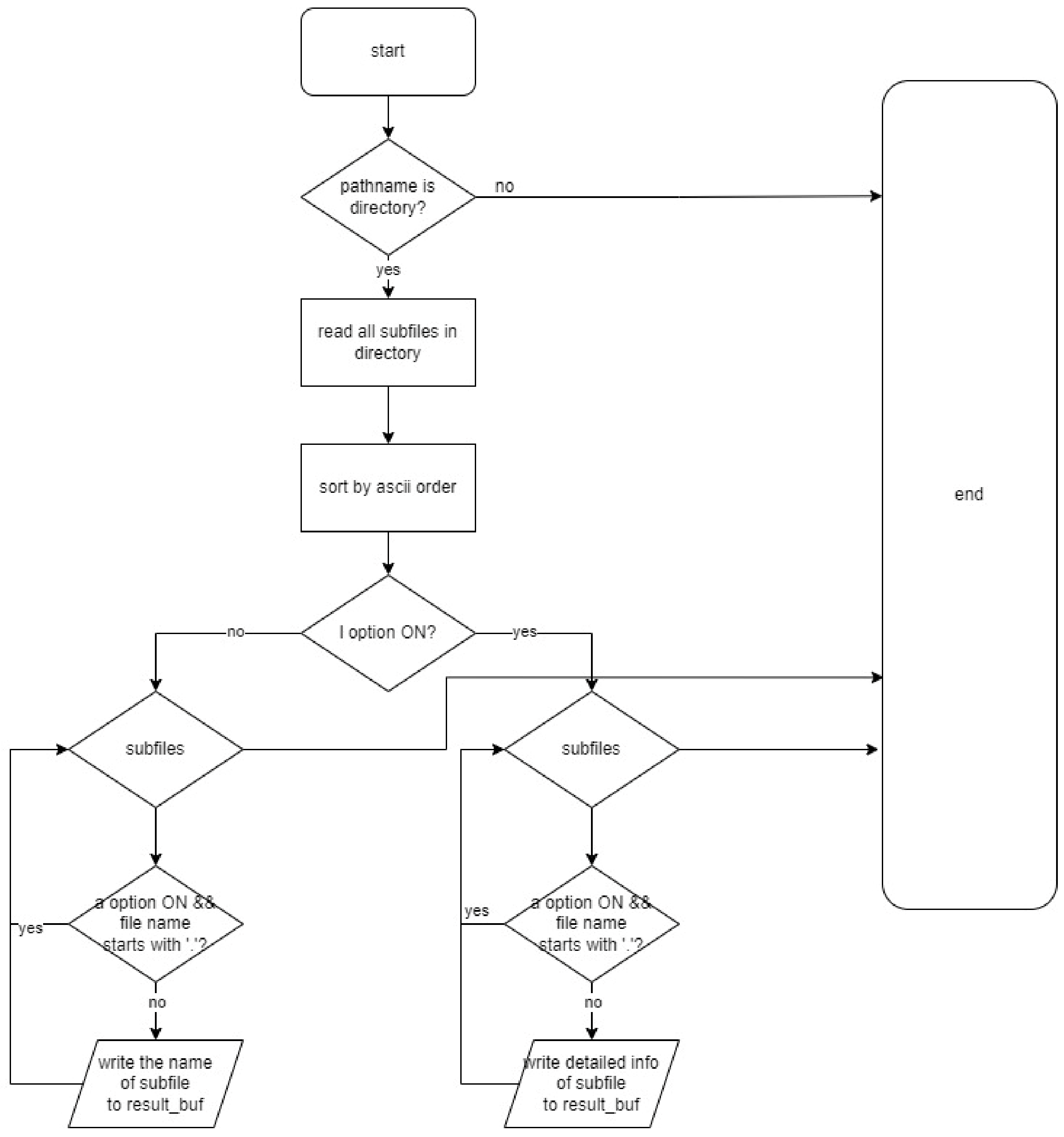
srv.c


start
start checking SIGCHLD SIGALRM SIGINT
oepn server socket
bind socket with server_addr
listen size 5
accept client
fork()
> 0
== 0
parent
add client info to linked list
print accepted client's information
print child's PID
print all clients' information
set alarm 10 secs
close client fd
child
1
read from Client
string == "QUIT"?
Yes
No
exit()
cmd_process (write result of cmd to result_buff)
print client's sending FTP command
send result_buff to client
SIGALRM checked?
print child processes' information
set alarm 10 secs
SIGCHLD checked?
waitpid(-1, NULL, WNOHANG)
SIGCHLD checked?
any child process left?
exit()
yes
wait()

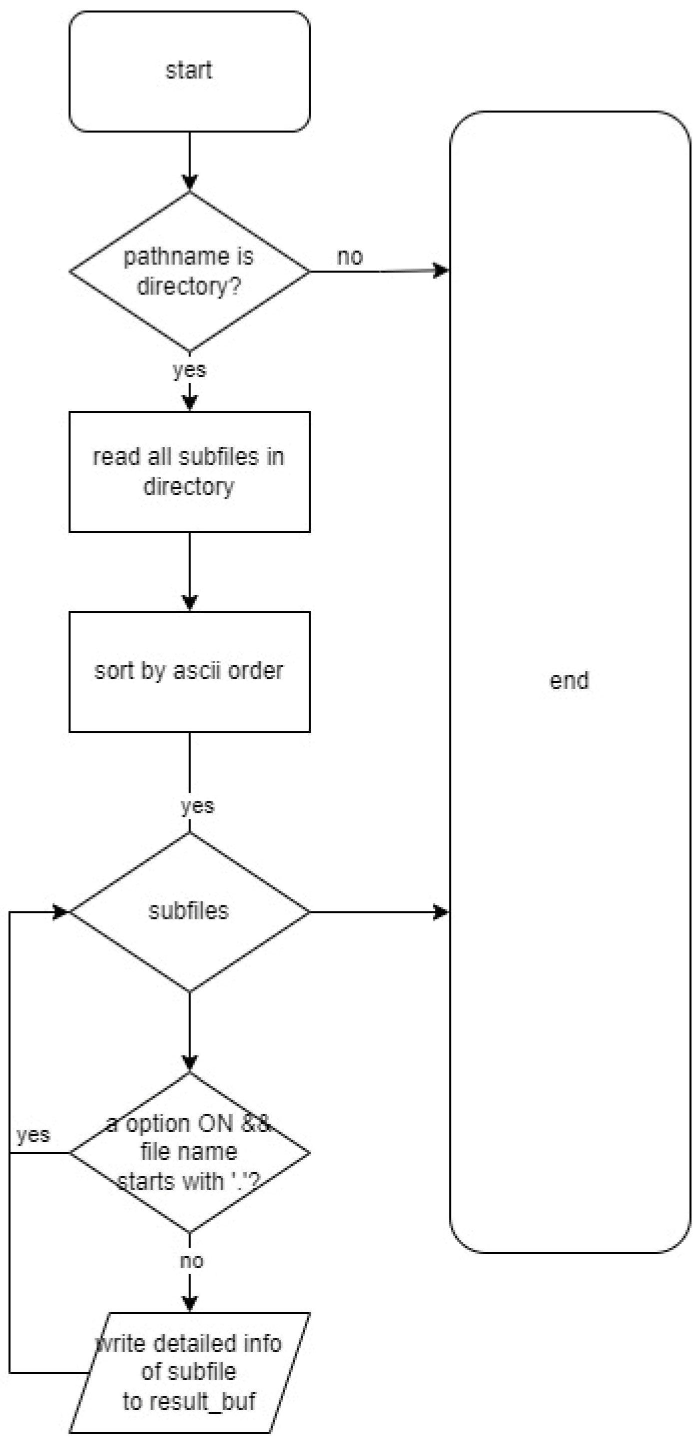
cmd_process


start
parsing Input using getopt()
no input || wrong command || invalid option
yes
no
command
NLST
LIST
PWD
CWD
CDUP
MKD
DELE
RMD
RNFR
argument except command
> 1
<= 1
> 0
= 0
= 1
== 0
!= 3
is it last argument in argv[]?
No
NLST
LIST
PWD
CWD
CDUP
MKD
DELE
RMD
RN
return val
0
-1
write error to result_buff
exit(1)
exit(0)
end

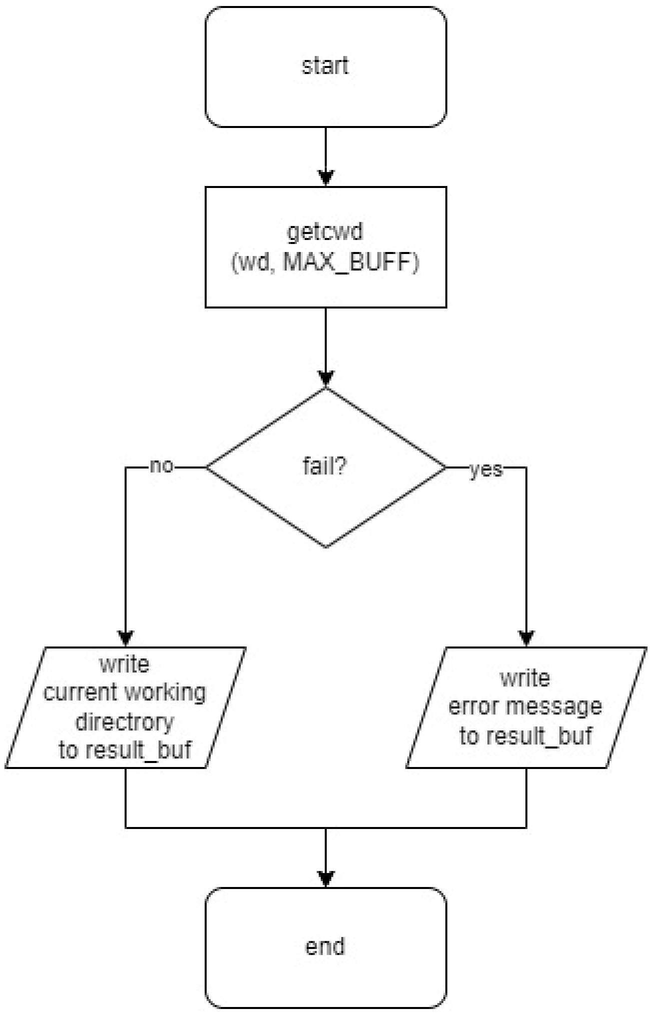
NLST


start
pathname is directory?
no
yes
read all subfiles in directory
sort by ascii order
l option ON?
no
yes
end
subfiles
subfiles
a option ON && file name starts with '.'?
yes
no
a option ON && file name starts with '.'?
yes
no
write the name of subfile to result_buf
write detailed info of subfile to result_buf

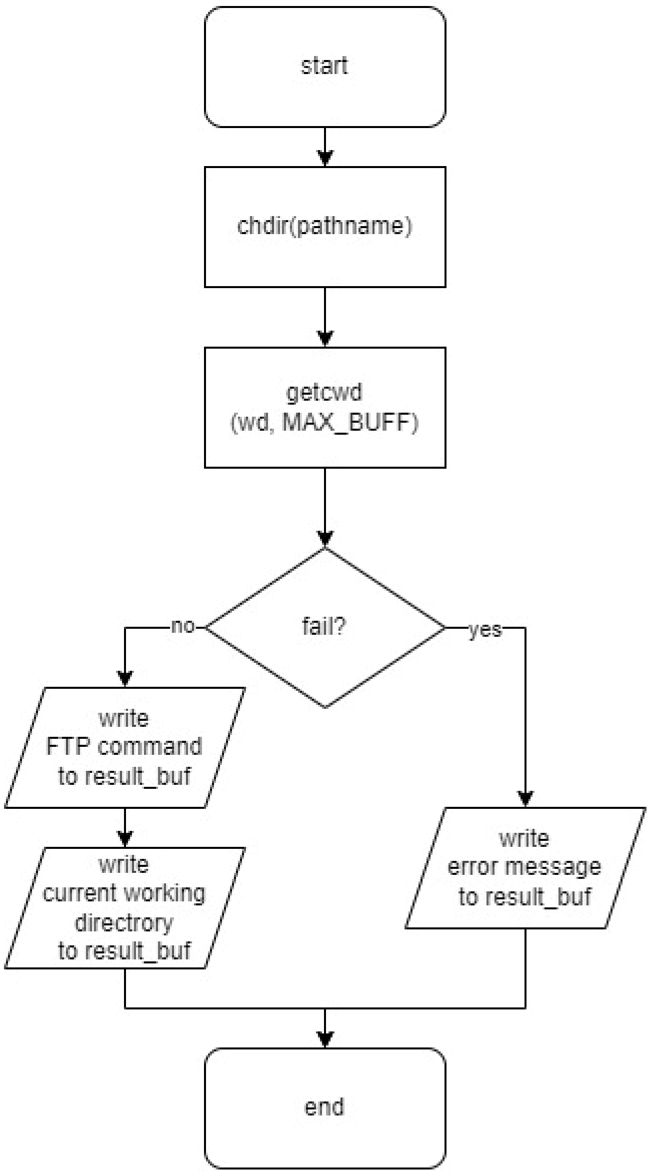
LIST


start
pathname is directory?
no
yes
read all subfiles in directory
sort by ascii order
yes
subfiles
yes
a option ON && file name starts with '.'?
no
write detailed info of subfile to result_buf
end

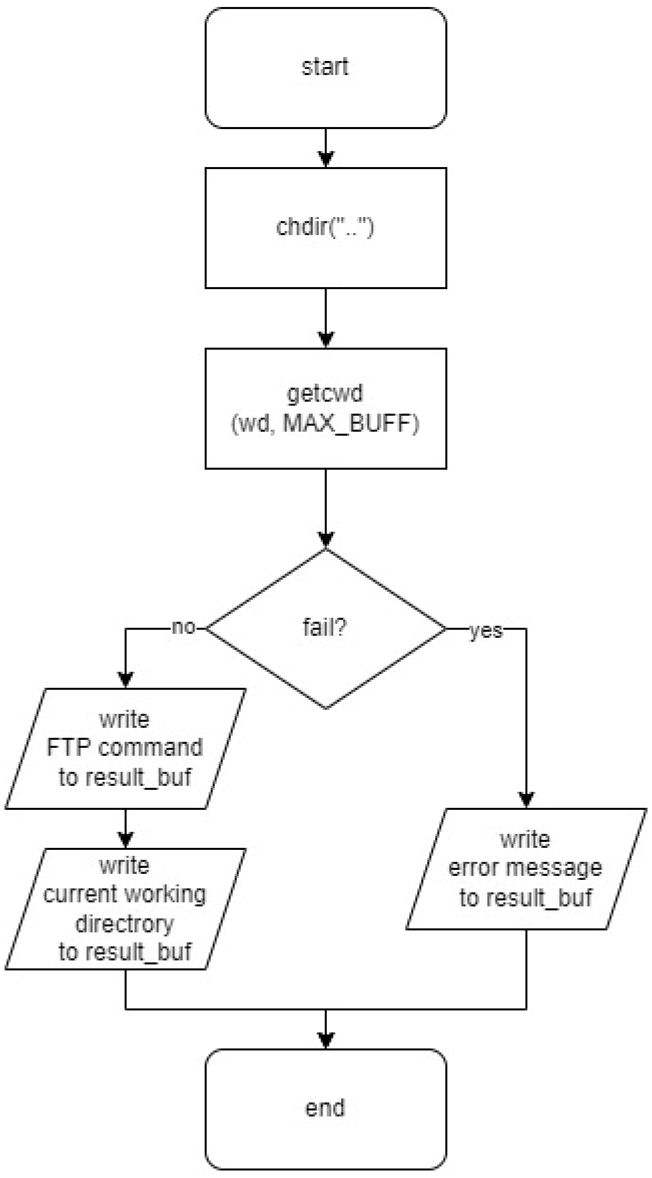
PWD


start
getcwd
(wd, MAX_BUFF)
fail?
no
yes
write
current working
directrory
to result_buf
write
error message
to result_buf
end

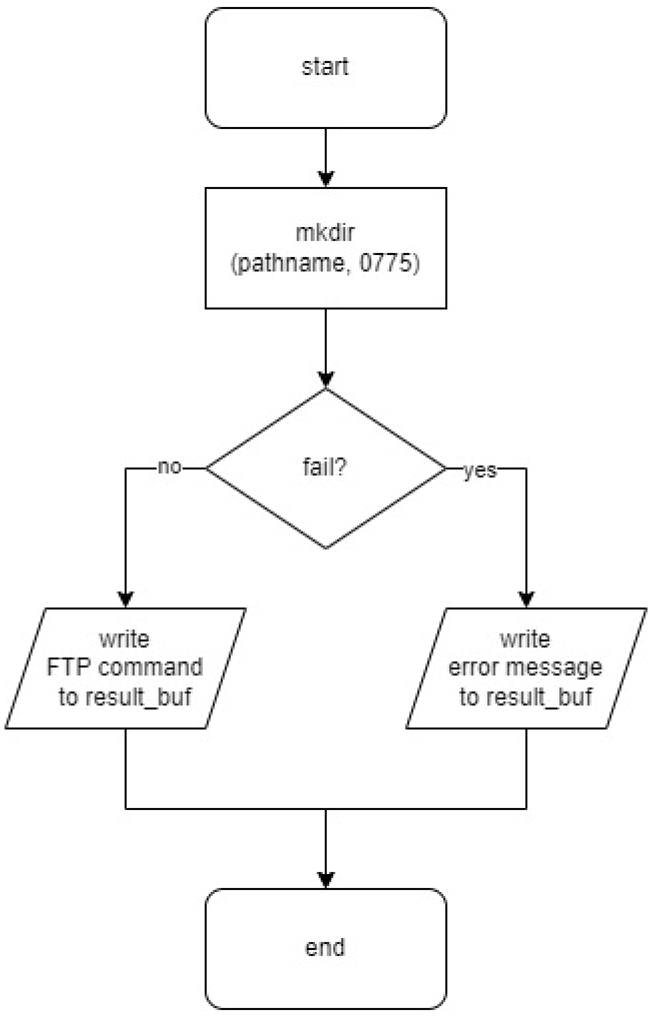
CWD


start
chdir(pathname)
getcwd
(wd, MAX_BUFF)
fail?
no
yes
write
FTP command
to result_buf
write
current working
directrory
to result_buf
write
error message
to result_buf
end

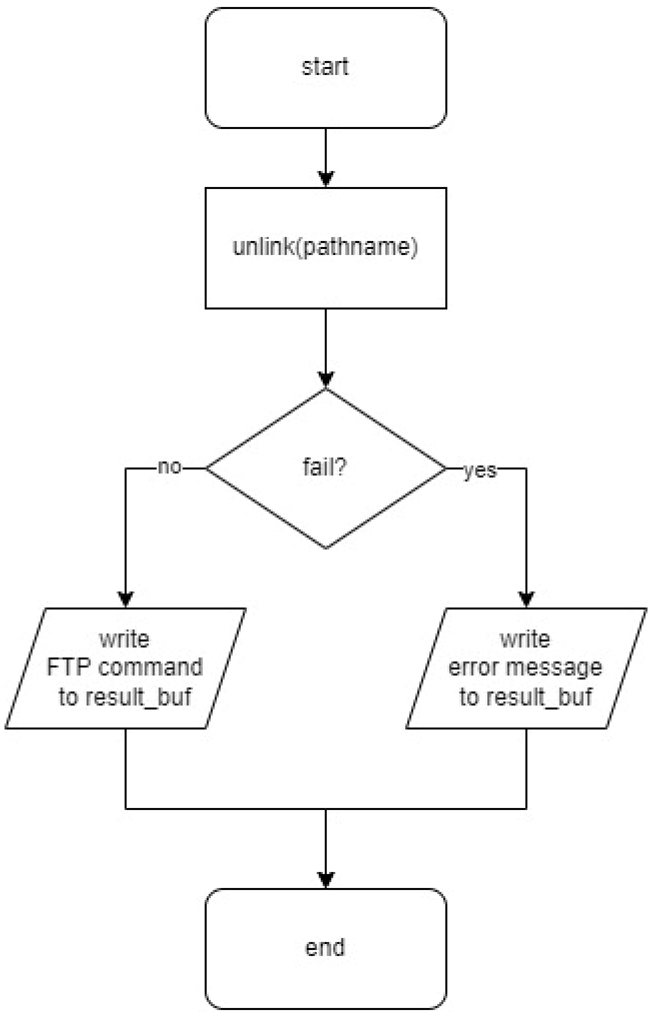
CDUP


start
chdir("..")
getcwd
(wd, MAX_BUFF)
fail?
no
yes
write
FTP command
to result_buf
write
current working
directrory
to result_buf
write
error message
to result_buf
end

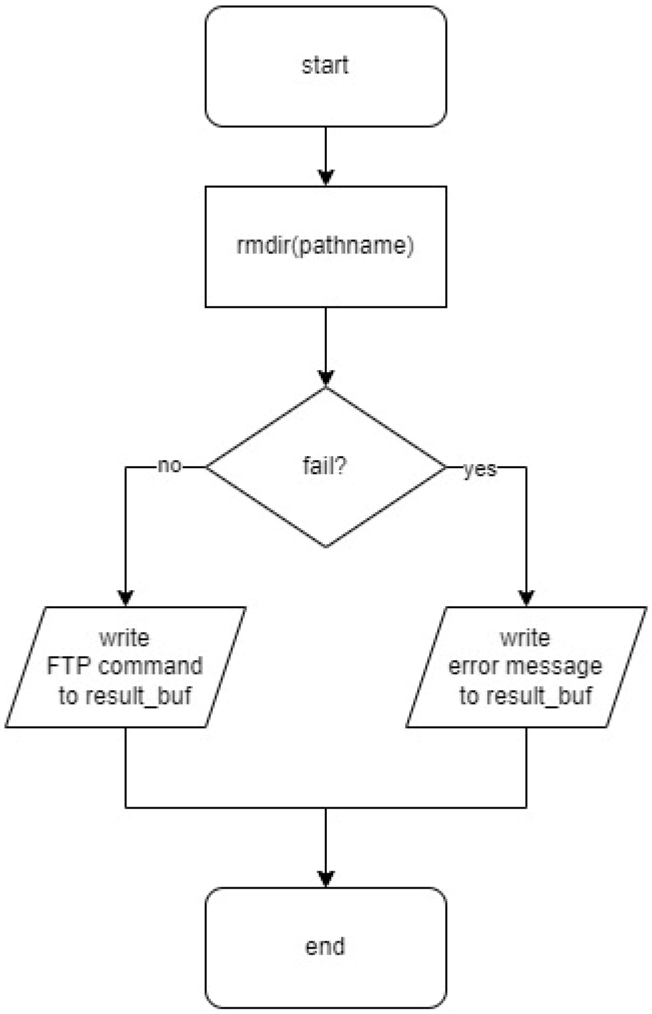
MKD


start
mkdir (pathname, 0775)
fail?
no
yes
write FTP command to result_buf
write error message to result_buf
end

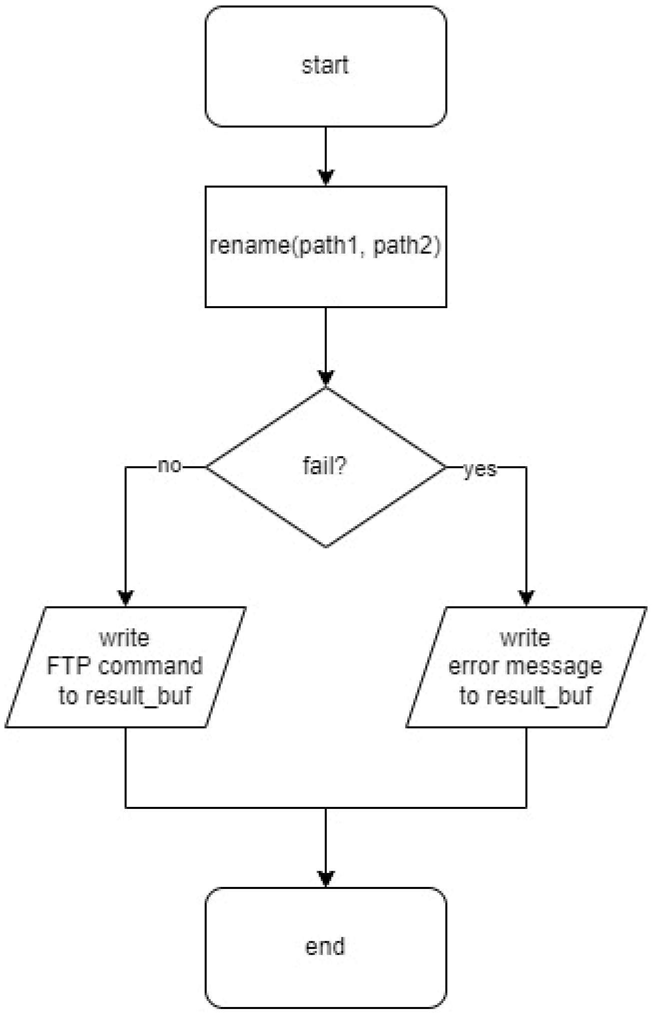
DELE


start
unlink(pathname)
fail?
no
yes
write FTP command to result_buf
write error message to result_buf
end

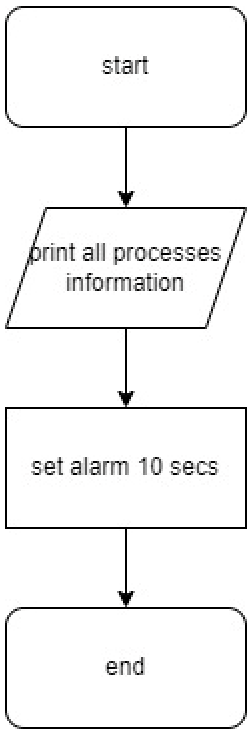
RMD


start
rmdir(pathname)
fail?
no
yes
write FTP command to result_buf
write error message to result_buf
end

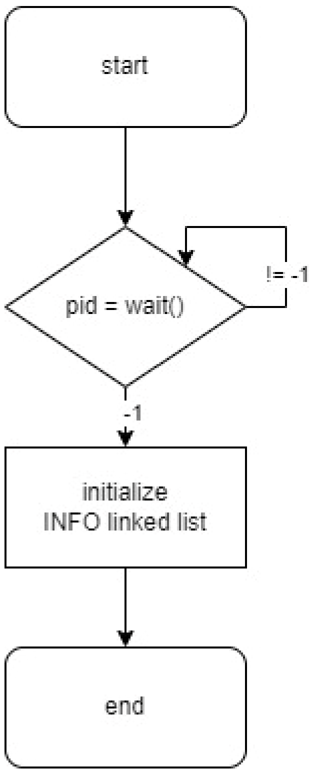
RN


start
rename(path1, path2)
fail?
no
yes
write FTP command to result_buf
write error message to result_buf
end

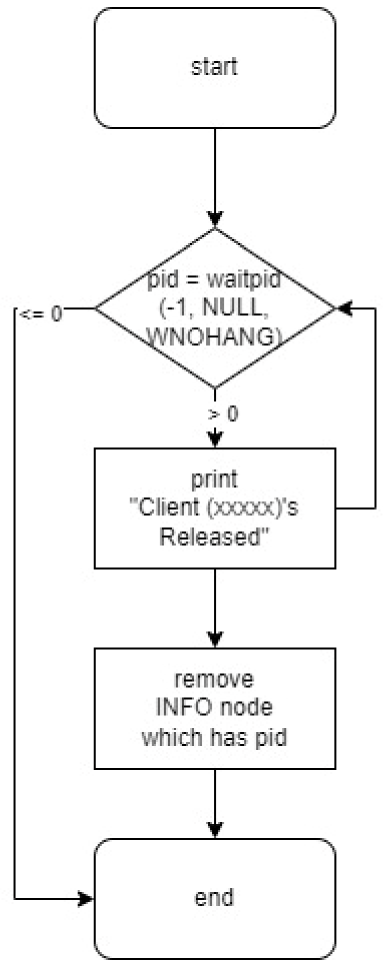
sh_alrm



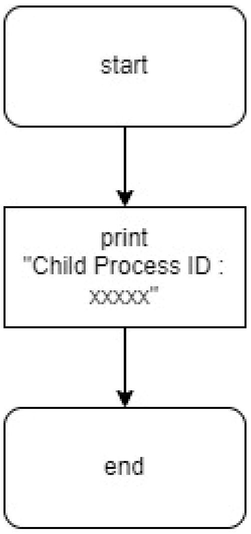
sh_int



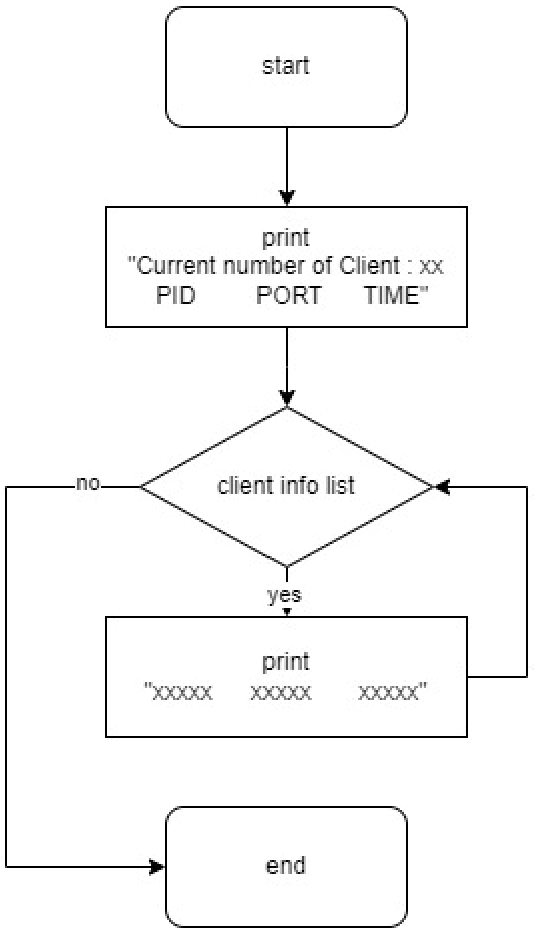
sh_chld


start
pid = waitpid (-1, NULL, WNOHANG)
<= 0
> 0
print "Client (xxxxx)'s Released"
remove INFO node which has pid
end

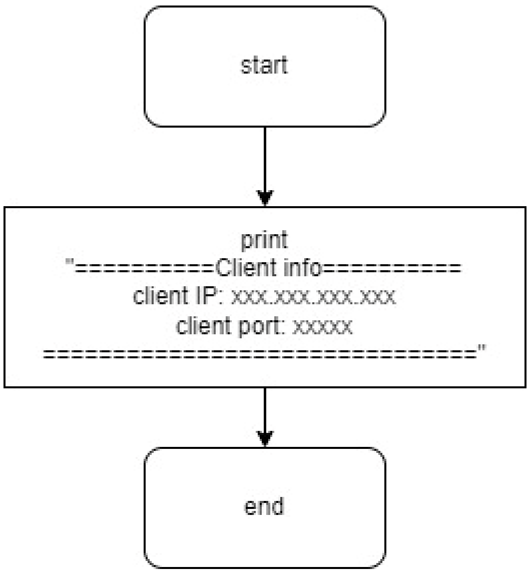
print_chd_pid



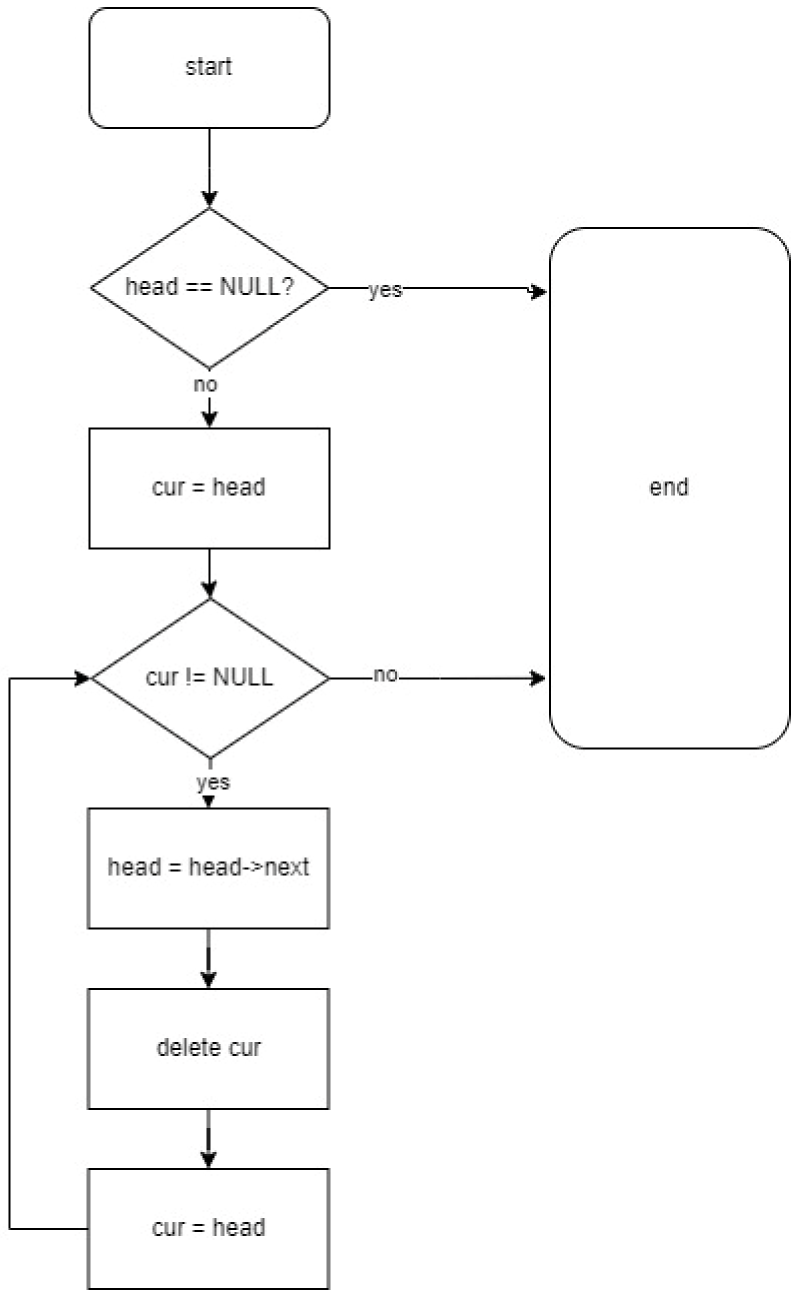
print_processes_info



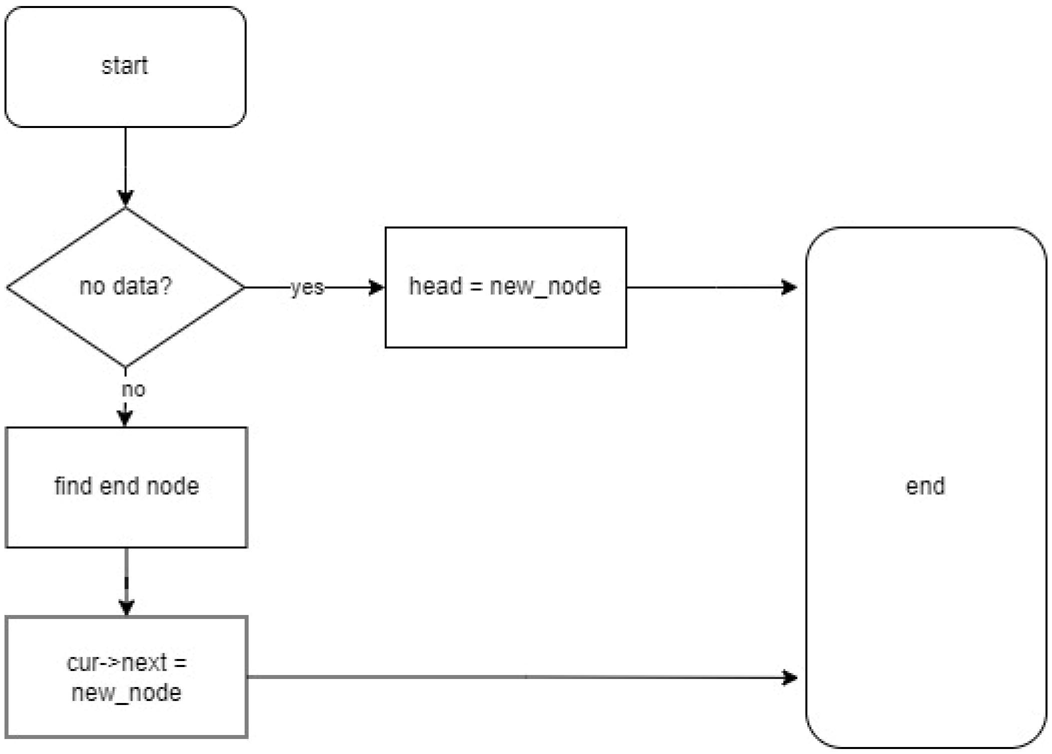
print_client_info



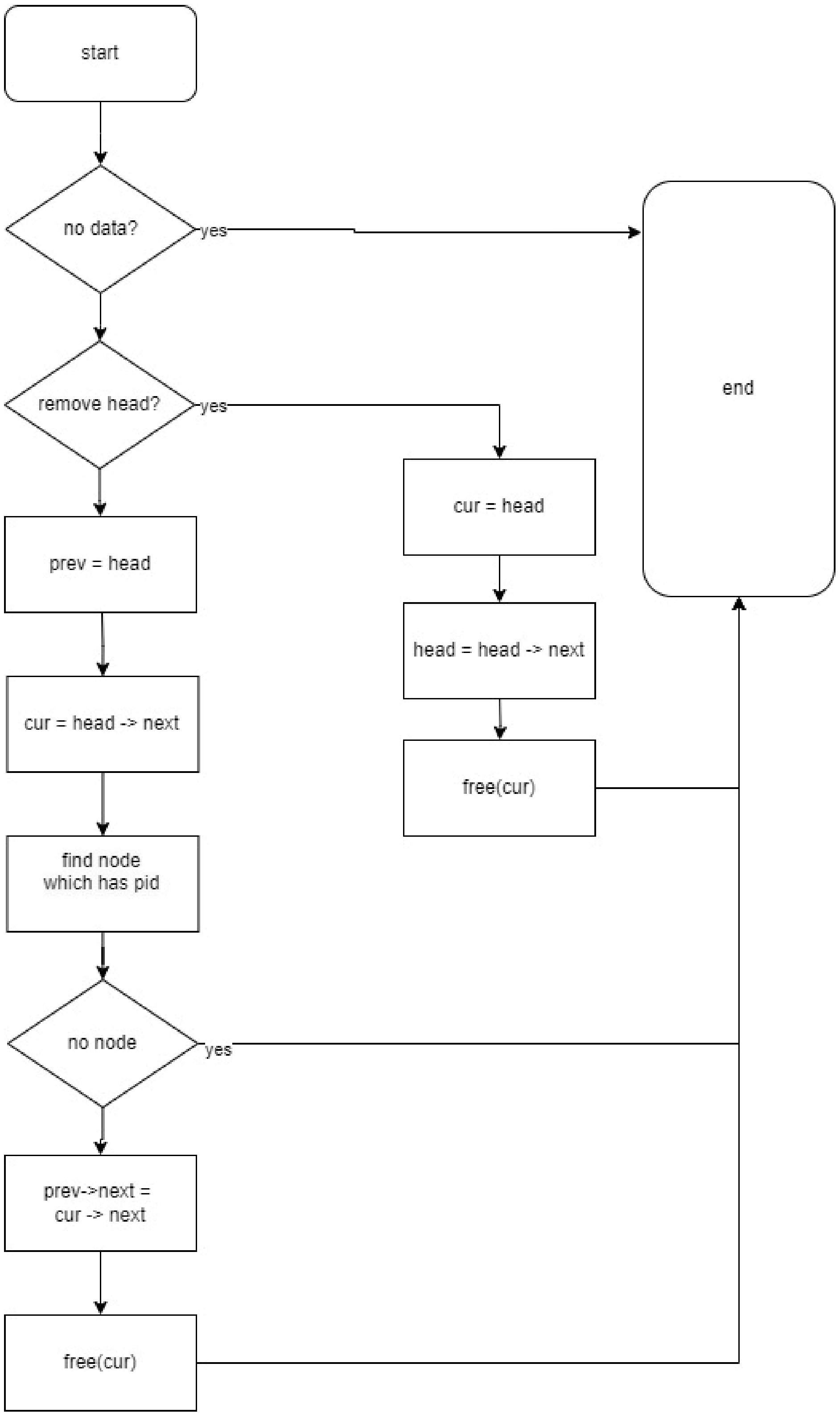
INFO_init


start
head == NULL?
yes
no
cur = head
cur != NULL
no
yes
head = head->next
delete cur
cur = head
end

INFO_add

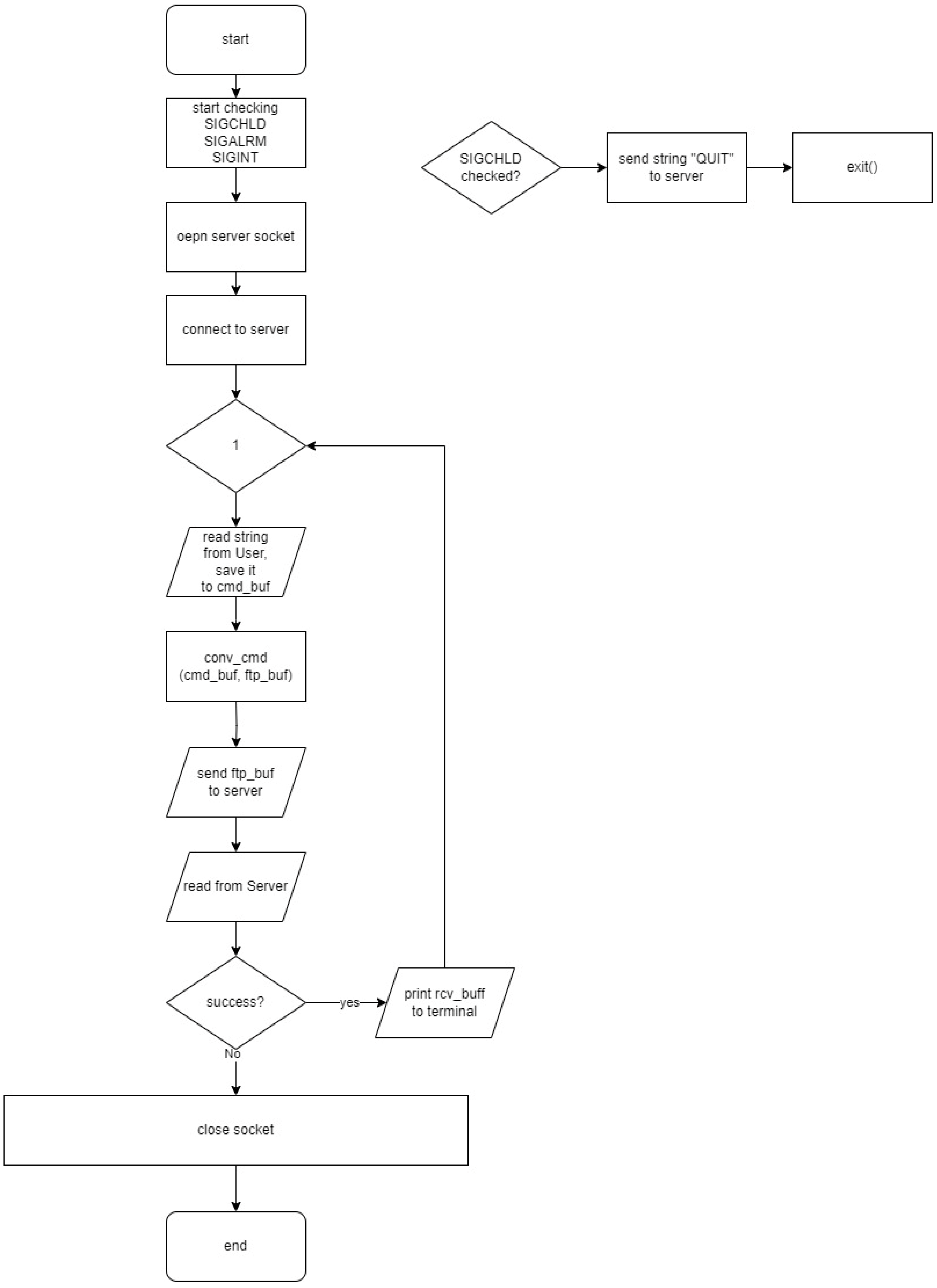

start
no data?
yes
head = new_node
no
find end node
cur->next = new_node
end

INFO_remove

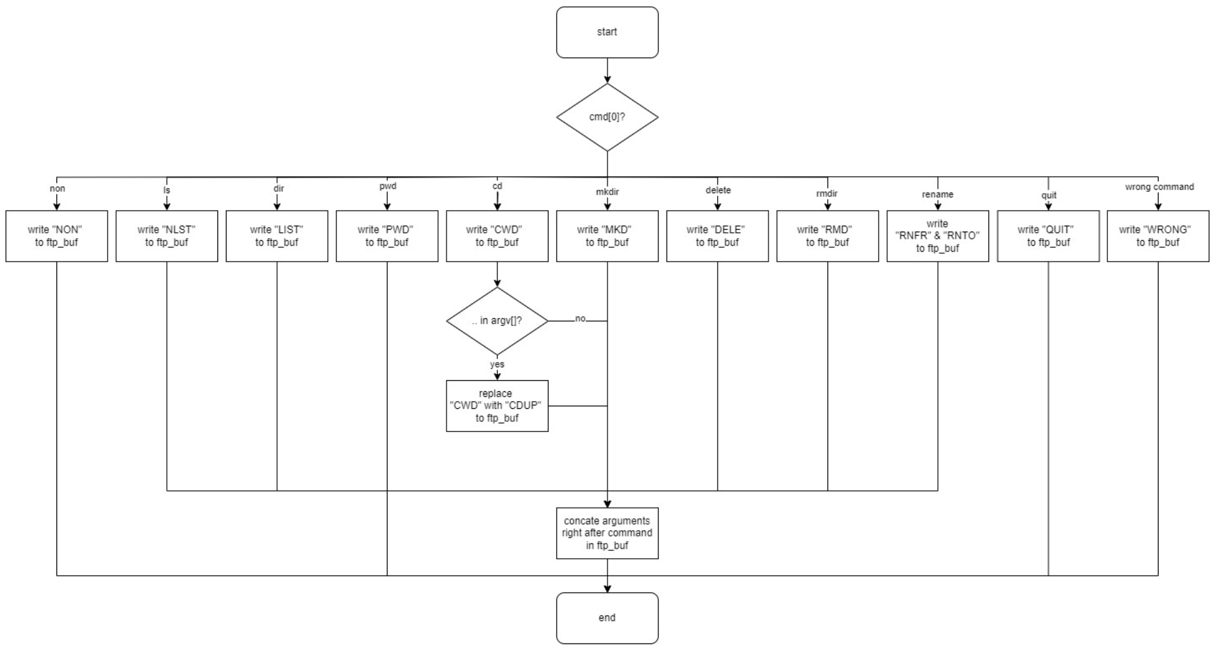

start
no data?
yes
end
remove head?
yes
cur = head
head = head -> next
free(cur)
prev = head
cur = head -> next
find node
which has pid
no node
yes
prev->next =
cur -> next
free(cur)

cli.c

main


start
start checking SIGCHLD SIGALRM SIGINT
oepn server socket
connect to server
1
read string from User, save it to cmd_buf
conv_cmd (cmd_buf, ftp_buf)
send ftp_buf to server
read from Server
success?
yes
print rcv_buff to terminal
No
close socket
end
SIGCHLD checked?
send string "QUIT" to server
exit()

conv_cmd



start
cmd[0]?
non
ls
dir
pwd
cd
mkdir
delete
rmdir
rename
quit
wrong command
write "NON" to ftp_buf
write "NLST" to ftp_buf
write "LIST" to ftp_buf
write "PWD" to ftp_buf
write "CWD" to ftp_buf
write "MKD" to ftp_buf
write "DELE" to ftp_buf
write "RMD" to ftp_buf
write "RNFR" & "RNTO" to ftp_buf
write "QUIT" to ftp_buf
write "WRONG" to ftp_buf
.. in argv[]?
no
yes
replace "CWD" with "CDUP" to ftp_buf
concate arguments right after command in ftp_buf
end

Pseudo code

srv.c

```
main
{
    Declare a buffer buf with size MAX_BUFF;
    Declare a buffer result_buf with size MAX_BUFF;
    Initialize all buf to zero;

    signal(SIGALRM);
    signal(SIGCHLD);
    signal(SIGINT);

    open socket;
    bind socket;
    listen starts;
    while (1)
    {
        accept client connect;
        pid = fork();
        if (pid > 0, that means parent process)
        {
            add client info to linked list;
            print accepted client information;
            print child PID;
            print all clients information;
        }
        else if (pid == 0, that means child process)
        {
            while (1)
            {
                read string from Client;
                if (string is "QUIT")
                    exit();

                cmd_process(buf);
                send result_buf to client;
            }
        }

        close client_fd
    }
    close server_fd return 0
}
```

```
int cmd_process(const char *buff, char *result_buff)
{
    parsing buf using getopt();

    if (input not fit in ftp command form)
        write error message to result_buf;
    else
    {
        if (command is "NLST")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (NLST < 0)
                write an error message to result_buf and return -1;
        }
        else if (command is "LIST")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (LIST < 0)
                print an error message and return -1;
        }
        else if (command is "PWD")
        {
            if (an argument is provided)
                write an error message to result_buf and return 0;
            if (PWD < 0)
                return -1;
        }
        else if (command is "CWD")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (CWD < 0)
                return -1;
        }
        else if (command is "CDUP")
        {
            if (there are too many arguments)
                write an error message to result_buf and return 0;
            if (CDUP < 0)
                return -1;
        }
        else if (command is "MKD")
        {
            if (there is no arguments)
                write an error message to result_buf and return 0;
            for (argv[])
                MKD;
        }
        else if (command is "DELE")
        {
            if (there is no arguments)
                write an error message to result_buf and return 0;
            for (argv[])
                DELE;
        }
        else if (command is RMD)
        {
            if (there is no arguments)
                write an error message to result_buf and return 0;
            for (argv[])
                RMD;
        }
        else if (command is RNFR and RNTO)
        {
            if (the number of arguments != 2)
                write an error message to result_buf and return 0;
            if (filename already exists)
                write an error message to result_buf and return 0;
            RN;
        }
    }

    return 0;
}
```

```
int NLST(char *result_buff, const char *pathname, int opflag)
{
    if (pathname is not directory)
        return -1;

    read all subfiles in directory named pathname;
    sort subfiles by ascii order;

    if (l option ON)
    {
        while (subfiles)
        {
            if (a option off && filename starts with '.')
                continue;
            else
                write detailed information of subfile to result_buf;
        }
    }
    else // l option OFF
    {
        while (subfiles)
        {
            if (a option off && filename starts with '.')
                continue;
            else
                write name of subfile to result_buf;
        }
    }
}
```

```
int LIST(char *result_buff, const char *pathname)
{
    if (pathname is not directory)
        return -1;

    read all subfiles in directory named pathname;
    sort subfiles by ascii order;

    while (subfiles)
    {
        if (a option off && filename starts with '.')
            continue;
        else
            write detailed information of subfile to result_buf;
    }
}
```

```
int PWD(char *result_buff)
{
    char wd[MAX_BUFF];

    if (getcwd(wd, MAX_BUFF) == NULL)
    {
        write error to result_buf;
        return -1;
    }
    else
    {
        write current working directory to result_buf;
        return 0;
    }
}
    }
}
```

```
int CWD(char *result_buff, const char *pathname)
{
    char wd[MAX_BUFF];

    if (chdir(pathname) < 0 || getcwd(wd, MAX_BUFF) == NULL)
    {
        write error to result_buf;
        return -1;
    }
    else
    {
        write FTP command to result_buf;
        write current working directory to result_buf;
        return 0;
    }
}
```

```
int CDUP(char *result_buff)
{
    char wd[MAX_BUFF];

    if (chdir("..") < 0 || getcwd(wd, MAX_BUFF) == NULL)
    {
        write error to result_buf;
        return -1;
    }
    else
    {
        write FTP command to result_buf;
        write current working directory to result_buf;
        return 0;
    }
}
```

```
int MKD(char *result_buff, const char *pathname)
{
    char str[MAX_BUFF];

    if (mkdir(pathname, 0775) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
int DELE(char *result_buff, const char *pathname)
{
    char str[MAX_BUFF];

    if (unlink(pathname) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
int RMD(char *result_buff, const char *pathname)
{
    char str[MAX_BUFF];

    if (rmdir(pathname) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
int RN(char *result_buff, const char *pathname1, const char *pathname2)
{
    if (rename(pathname1, pathname2) == 0)
    {
        write FTP command to result_buf;
        return 0;
    }
    else
    {
        write error to result_buf;
        return -1;
    }
}
```

```
void sh_alrm(int signum)
{
    print all processes information;
    set alarm 10 secs;
}
```

```
void sh_int(int signum)
{
    pid_t pid;

    while ((pid = wait(NULL)) != -1)
    {
    }

    INFO_init();
    exit(1);
}
```

```
void sh_chld(int signum)
{
    pid_t pid;

    while ((pid = waitpid(-1, NULL, WNOHANG)) > 0)
    {
        print string according to the following form;
        // Client( xxxxx)'s Release.
        INFO_remove(pid);
        num_children--;
    }
}
```

```
void print_cli_info(struct sockaddr_in cli_addr)
{
    print string according to the following form;
    // ==========Client info==========
    // client IP: xxx.xxx.xxx.xxx
    // client port: xxxxx
    // ===============================
}
```

```
void print_chd_pid(int pid)
{
    print string according to the following form;
    // Child Process ID : xxxxx
}
```

```
void print_processes_info()
{
    print string according to the following form;
    // Current Number of Client :  xx
    //   PID     PORT    TIME

    while (client info list)
    {
        print PID, PORT, TIME according to the following form;
        // xxxxx    xxxxx   xxxxx
    }
}
```

```
void INFO_init()
{
    INFO *cur = head;

    if (head == NULL)
        return;
    else
    {
        while (cur != NULL)
        {
            head point to the next node;
            delete cur;
            cur = head;
        }
    }
}
```

```
void INFO_add(pid_t pid, int port, time_t start)
{
    INFO *cur = head;
    INFO *new_node = INFO(pid, port, start);

    if (there is no INFO)
        head = new_node;
    else
        insert new_node to the end of linked list;
}
```

```
int INFO_remove(pid_t pid)
{
    if (there is no INFO)
        return 0;

    if (head->pid == pid)
    {
        cur = head;
        head = head->next;
        free(cur);
        return 1;
    }
    else
    {
        prev = head;
        cur = head->next;
        while (cur != NULL && cur->pid != pid)
        {
            prev = cur;
            cur = cur->next;
        }

        if (cur == NULL)
            return 0;

        prev->next = cur->next;
        free(cur);
        return 1;
    }
}
```

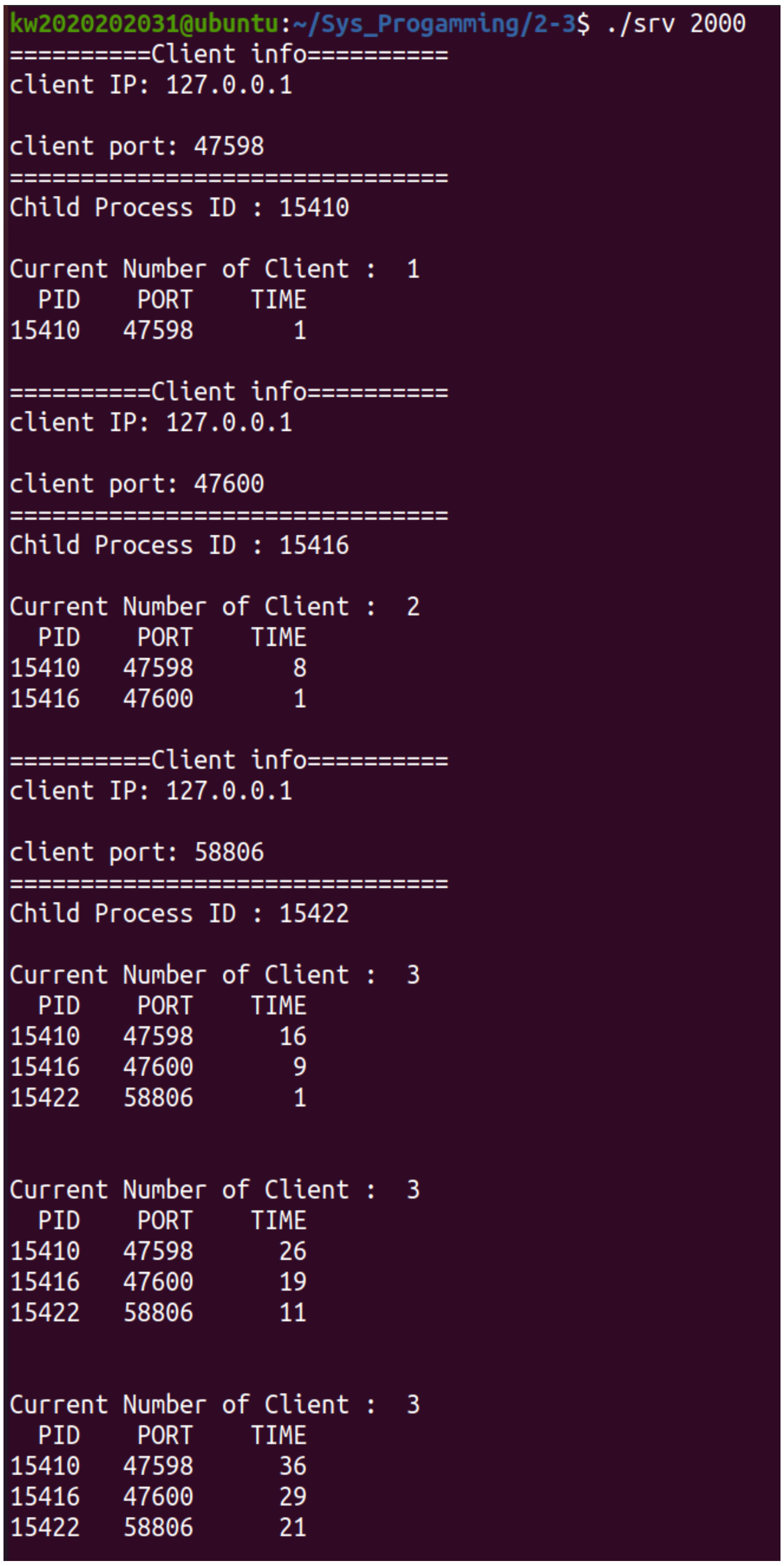
cli.c

```
main
{
    Declare a buffer cmd_buf with size MAX_BUFF.
    Declare a buffer ftp_buf with size MAX_BUFF.
    Declare a buffer rcv_buf with size MAX_BUFF.

    signal(SIGINT)

    Initialize all buf to zero.
    open socket
    connect to server

    while(1)
    {
        read string from USER
        conv_cmd(cmd_buf)
        write ftp_buf to Server
        if(write fails)
            break
        else
        {
            read from Server
            if(read fails)
                break
            else
                print rcv_buf
        }
    }
    close server_fd

    return 0
}
```

```
conv_cmd
{
    getopt(cmd_buf)
    if( the number of input arguments is 0)
        Copy the string "NON" to ftp_buf.
    else if (first input argument is "ls")
        Copy the string "NLST" to ftp_buf.
    else if (first input argument is "dir")
        Copy the string "LIST" to ftp_buf.
    else if (first input argument is "pwd")
        Copy the string "PWD" to ftp_buf.
    else if (first input argument is "cd")
        Copy the string "CWD" to ftp_buf.

    If additional argument is ".."
        Copy the string "CDUP" to ftp_buf.
    else
        append additional argument to ftp_buf.
    else if (first input argument is "mkdir")
        Copy the string "MKD" to ftp_buf.
    else if (first input argument is "delete")
        Copy the string "DELE" to ftp_buf.
    else if (first input argument is "rmdir")
        Copy the string "RMD" to ftp_buf.
    else if (first input argument is "rename")
        Copy the string "RNFR" and the second argument to ftp_buf.
        Copy the string "RNTO" and the third argument to ftp_buf.
    else if (first input argument is "quit")
        Copy the string "QUIT" to ftp_buf.
    else (incorrect command entered)
        Copy the string "WRONG" to ftp_buf.

    If there are additional arguments:
        Append a space to ftp_buf.
        Append the additional argument to ftp_buf.
}
```

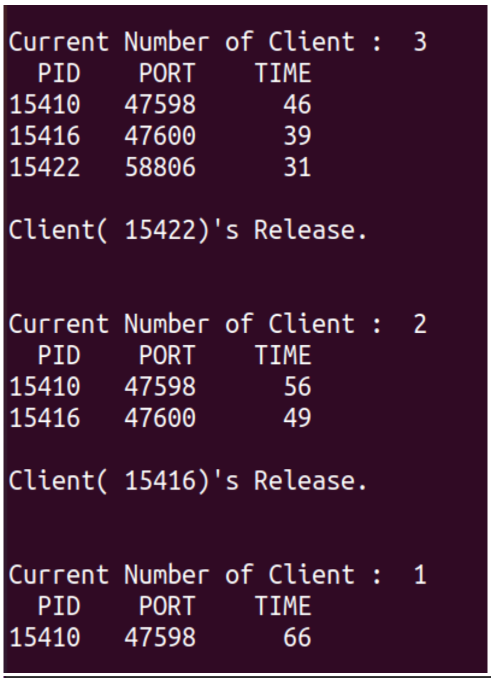
```
sh_int
{
    send sting "QUIT" to server
    exit()
}
```

결과화면

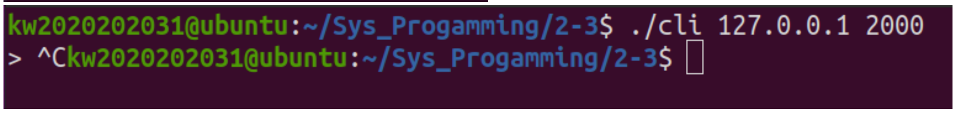


```
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ ./srv 2000
==========Client info==========
client IP: 127.0.0.1

client port: 47598
===============================
Child Process ID : 15410

Current Number of Client :  1
  PID     PORT     TIME
15410   47598       1

==========Client info==========
client IP: 127.0.0.1

client port: 47600
===============================
Child Process ID : 15416

Current Number of Client :  2
  PID     PORT     TIME
15410   47598       8
15416   47600       1

==========Client info==========
client IP: 127.0.0.1

client port: 58806
===============================
Child Process ID : 15422

Current Number of Client :  3
  PID     PORT     TIME
15410   47598      16
15416   47600       9
15422   58806       1


Current Number of Client :  3
  PID     PORT     TIME
15410   47598      26
15416   47600      19
15422   58806      11


Current Number of Client :  3
  PID     PORT     TIME
15410   47598      36
15416   47600      29
15422   58806      21
```

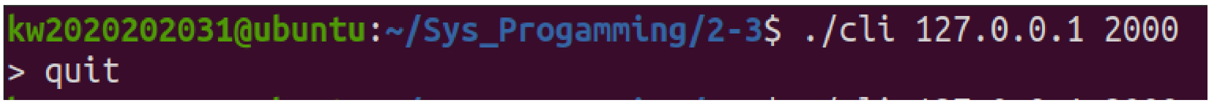
server 에 client 가 연결될 때마다 client info 와 함께 client 가 현재 연결돼 있는 child process 의 정보를 출력합니다.



```
Current Number of Client :  3
  PID     PORT     TIME
15410   47598       46
15416   47600       39
15422   58806       31

Client( 15422)'s Release.


Current Number of Client :  2
  PID     PORT     TIME
15410   47598       56
15416   47600       49

Client( 15416)'s Release.


Current Number of Client :  1
  PID     PORT     TIME
15410   47598       66
```



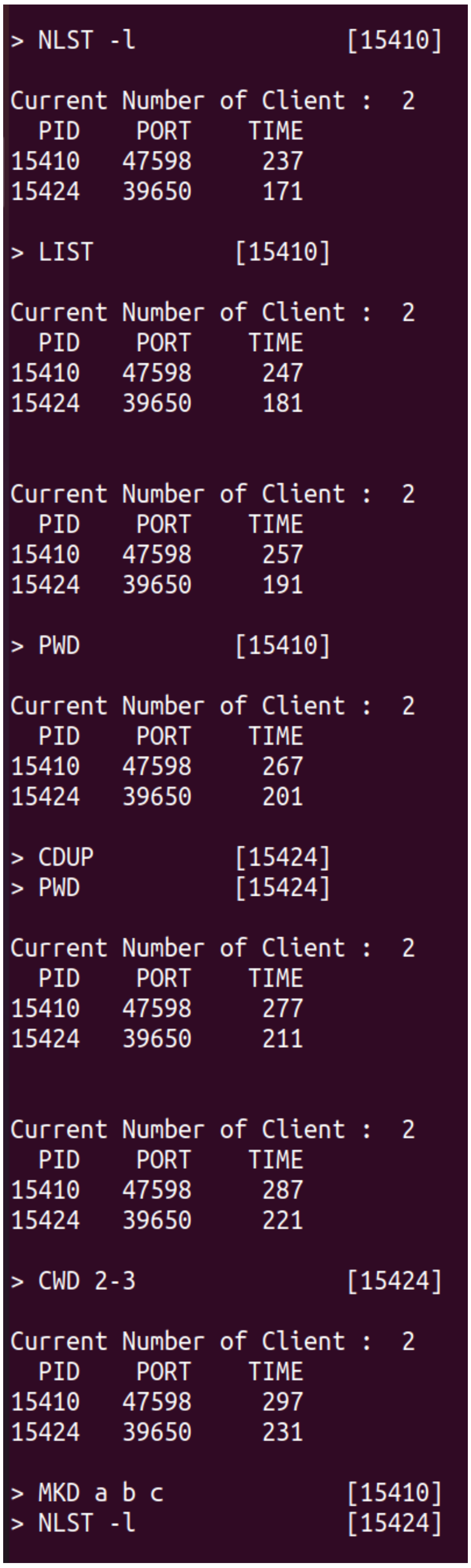
```
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ ./cli 127.0.0.1 2000
> ^Ckw2020202031@ubuntu:~/Sys_Progamming/2-3$ 
```



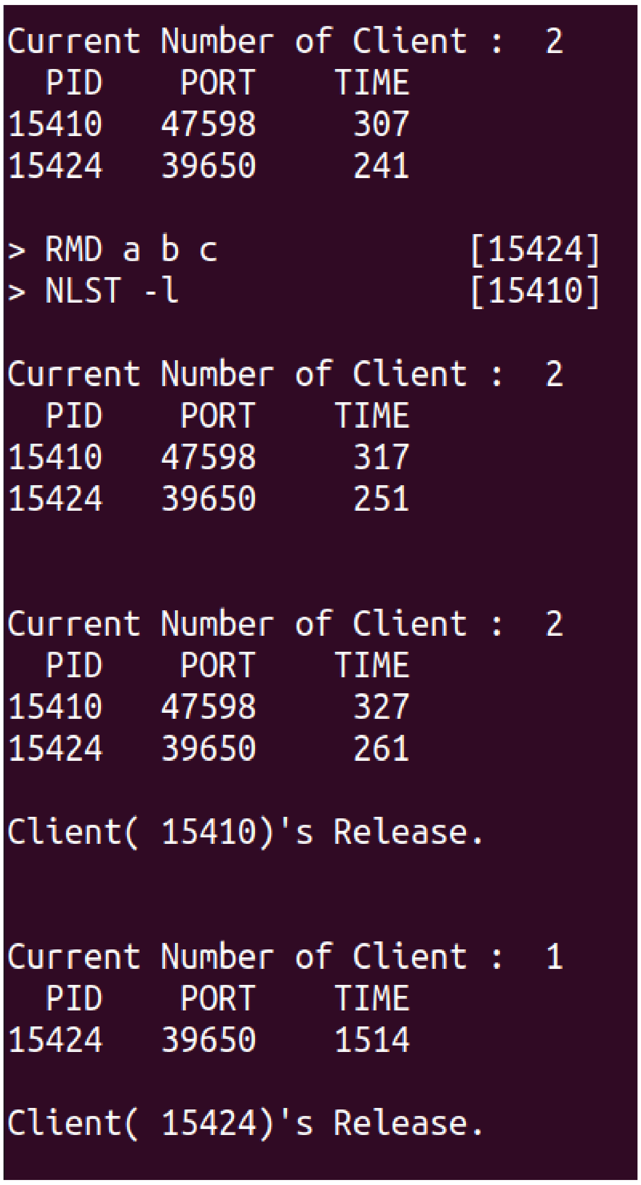
```
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ ./cli 127.0.0.1 2000
> quit
```

client 가 각각 ctrl+C, quit 를 입력하자 server 와의 연결이 끊기고 서버에서도 해당 child process 를 종료시켜서 연결된 process 들이 줄어드는 것을 확인할 수 있습니다.

server

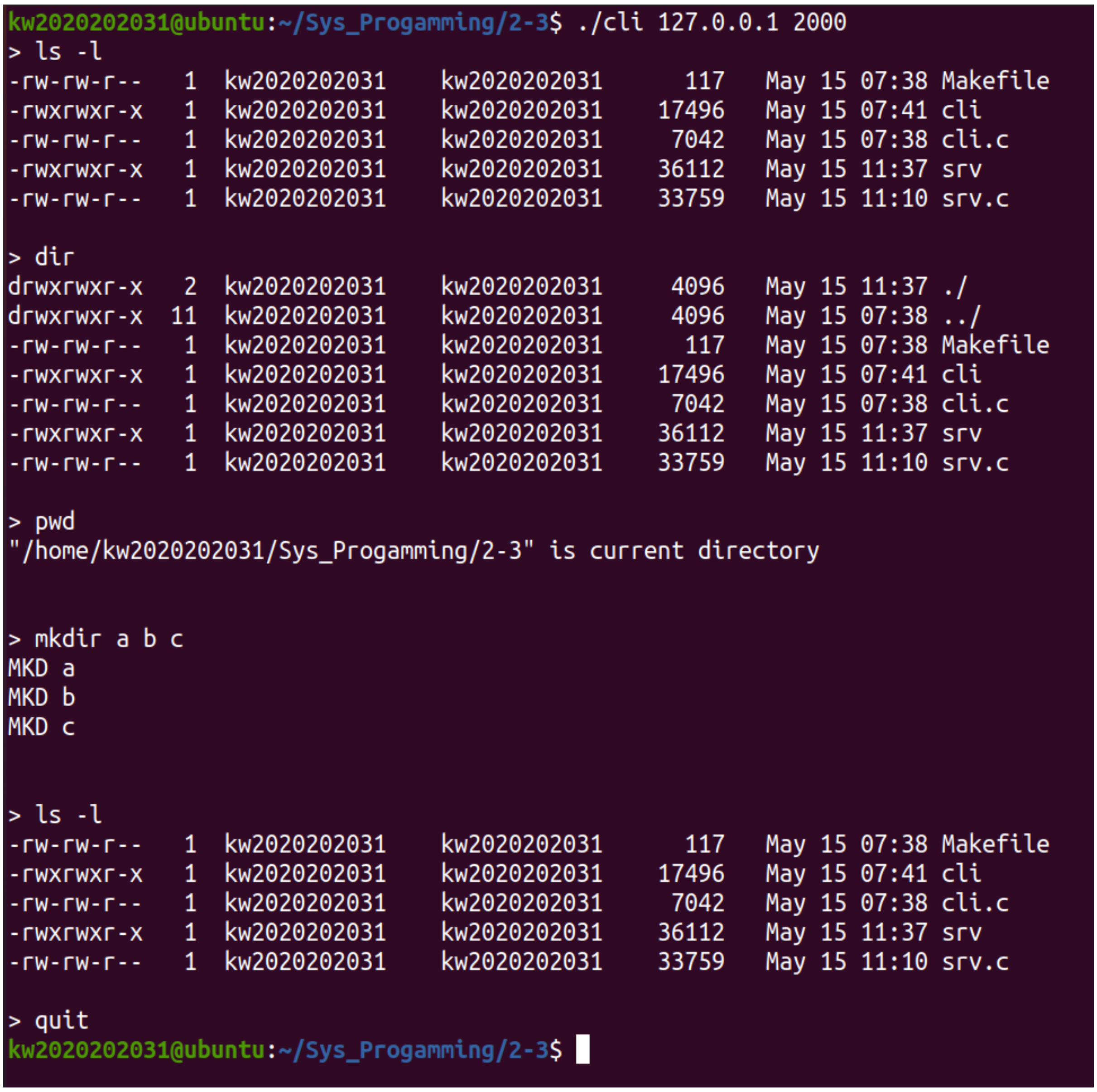


```
> NLST -l                 [15410]

Current Number of Client :  2
  PID     PORT     TIME
15410   47598      237
15424   39650      171

> LIST            [15410]

Current Number of Client :  2
  PID     PORT     TIME
15410   47598      247
15424   39650      181


Current Number of Client :  2
  PID     PORT     TIME
15410   47598      257
15424   39650      191

> PWD             [15410]

Current Number of Client :  2
  PID     PORT     TIME
15410   47598      267
15424   39650      201

> CDUP            [15424]
> PWD             [15424]

Current Number of Client :  2
  PID     PORT     TIME
15410   47598      277
15424   39650      211


Current Number of Client :  2
  PID     PORT     TIME
15410   47598      287
15424   39650      221

> CWD 2-3                 [15424]

Current Number of Client :  2
  PID     PORT     TIME
15410   47598      297
15424   39650      231

> MKD a b c               [15410]
> NLST -l                 [15424]
```



```
Current Number of Client :  2
  PID     PORT     TIME
15410   47598      307
15424   39650      241

> RMD a b c               [15424]
> NLST -l                 [15410]

Current Number of Client :  2
  PID     PORT     TIME
15410   47598      317
15424   39650      251


Current Number of Client :  2
  PID     PORT     TIME
15410   47598      327
15424   39650      261

Client( 15410)'s Release.


Current Number of Client :  1
  PID     PORT     TIME
15424   39650     1514

Client( 15424)'s Release.
```

client [15410]



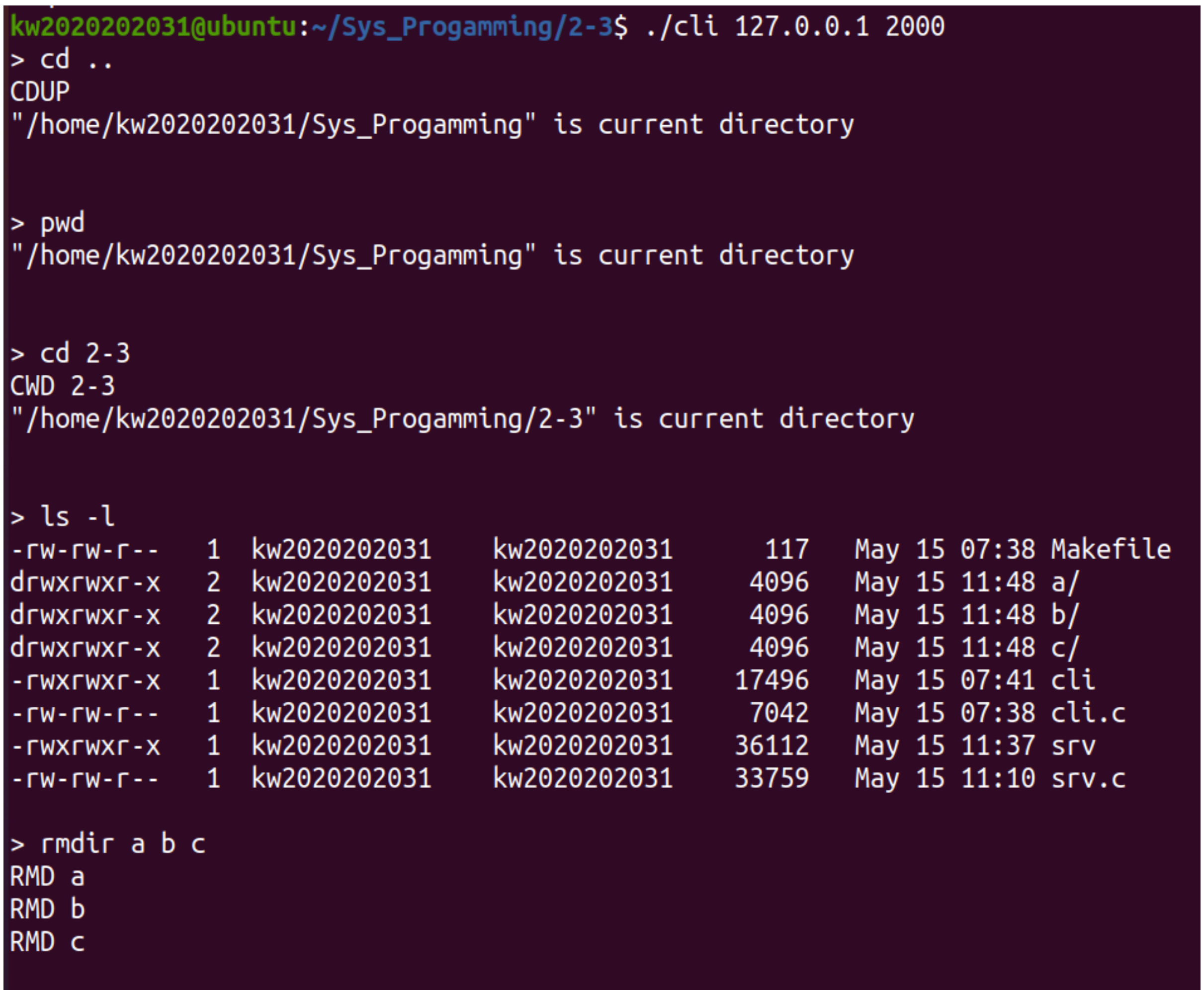
```
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ ./cli 127.0.0.1 2000
> ls -l
-rw-rw-r--    1  kw2020202031    kw2020202031      117   May 15 07:38 Makefile
-rwxrwxr-x    1  kw2020202031    kw2020202031    17496   May 15 07:41 cli
-rw-rw-r--    1  kw2020202031    kw2020202031     7042   May 15 07:38 cli.c
-rwxrwxr-x    1  kw2020202031    kw2020202031    36112   May 15 11:37 srv
-rw-rw-r--    1  kw2020202031    kw2020202031    33759   May 15 11:10 srv.c

> dir
drwxrwxr-x    2  kw2020202031    kw2020202031     4096   May 15 11:37 ./
drwxrwxr-x   11  kw2020202031    kw2020202031     4096   May 15 07:38 ../
-rw-rw-r--    1  kw2020202031    kw2020202031      117   May 15 07:38 Makefile
-rwxrwxr-x    1  kw2020202031    kw2020202031    17496   May 15 07:41 cli
-rw-rw-r--    1  kw2020202031    kw2020202031     7042   May 15 07:38 cli.c
-rwxrwxr-x    1  kw2020202031    kw2020202031    36112   May 15 11:37 srv
-rw-rw-r--    1  kw2020202031    kw2020202031    33759   May 15 11:10 srv.c

> pwd
"/home/kw2020202031/Sys_Progamming/2-3" is current directory


> mkdir a b c
MKD a
MKD b
MKD c


> ls -l
-rw-rw-r--    1  kw2020202031    kw2020202031      117   May 15 07:38 Makefile
-rwxrwxr-x    1  kw2020202031    kw2020202031    17496   May 15 07:41 cli
-rw-rw-r--    1  kw2020202031    kw2020202031     7042   May 15 07:38 cli.c
-rwxrwxr-x    1  kw2020202031    kw2020202031    36112   May 15 11:37 srv
-rw-rw-r--    1  kw2020202031    kw2020202031    33759   May 15 11:10 srv.c

> quit
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ 
```

ls dir, pwd 다 잘 작동하는 것을 확인할 수 있습니다.

mkdir a b c 로 a b c 디렉토리를 생성했지만 ls 에서 출력되지 않은 이유는 client[15424]에서 rmdir a b c 를 통해 a b c 디렉토리를 삭제했기 때문입니다.

quit 명령어를 입력하여 server 와의 연결을 끊은 것을 확인할 수 있습니다.

client [15424]

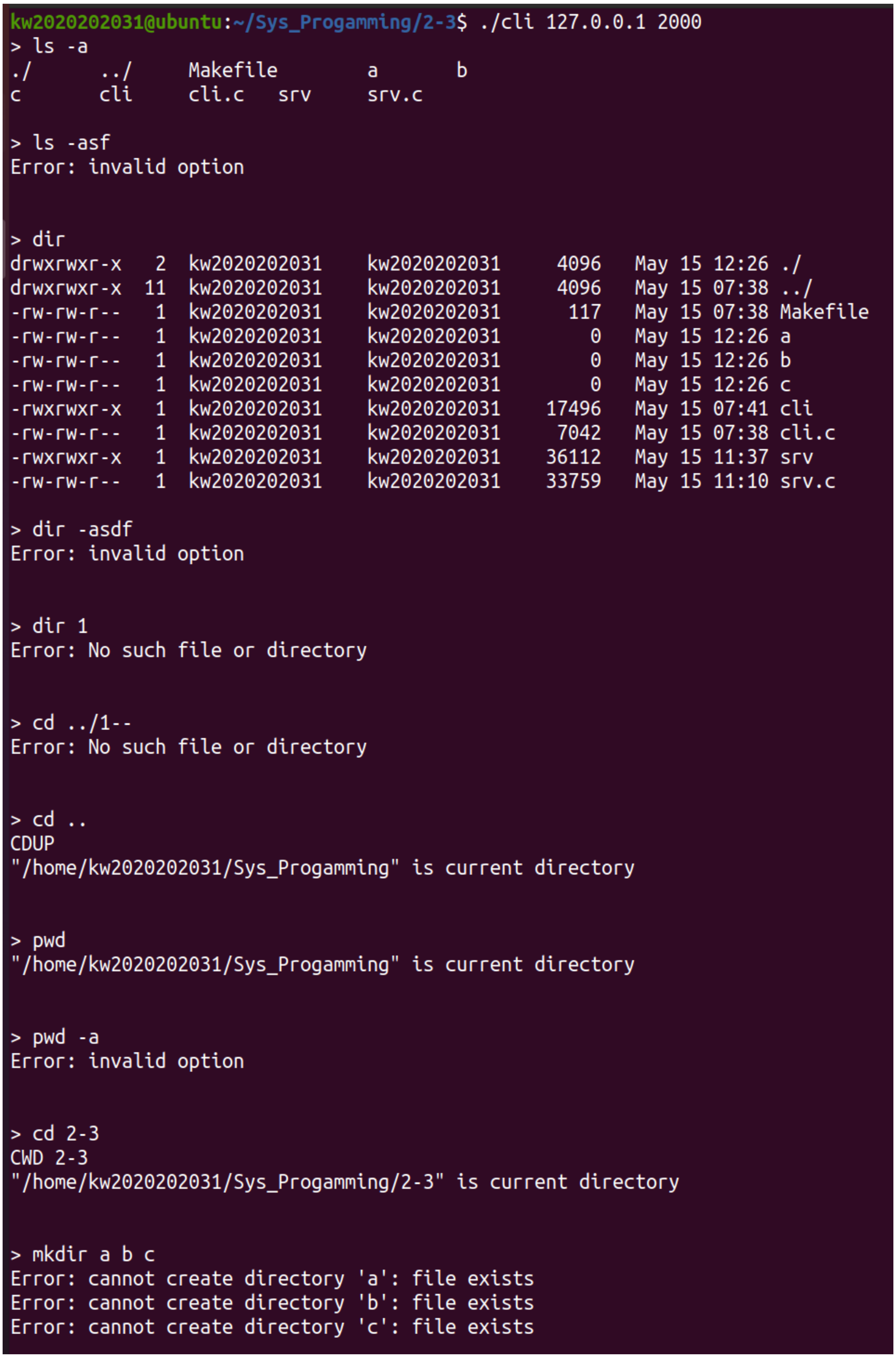


```
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ ./cli 127.0.0.1 2000
> cd ..
CDUP
"/home/kw2020202031/Sys_Progamming" is current directory


> pwd
"/home/kw2020202031/Sys_Progamming" is current directory


> cd 2-3
CWD 2-3
"/home/kw2020202031/Sys_Progamming/2-3" is current directory


> ls -l
-rw-rw-r--   1  kw2020202031    kw2020202031      117   May 15 07:38 Makefile
drwxrwxr-x   2  kw2020202031    kw2020202031     4096   May 15 11:48 a/
drwxrwxr-x   2  kw2020202031    kw2020202031     4096   May 15 11:48 b/
drwxrwxr-x   2  kw2020202031    kw2020202031     4096   May 15 11:48 c/
-rwxrwxr-x   1  kw2020202031    kw2020202031    17496   May 15 07:41 cli
-rw-rw-r--   1  kw2020202031    kw2020202031     7042   May 15 07:38 cli.c
-rwxrwxr-x   1  kw2020202031    kw2020202031    36112   May 15 11:37 srv
-rw-rw-r--   1  kw2020202031    kw2020202031    33759   May 15 11:10 srv.c

> rmdir a b c
RMD a
RMD b
RMD c
```

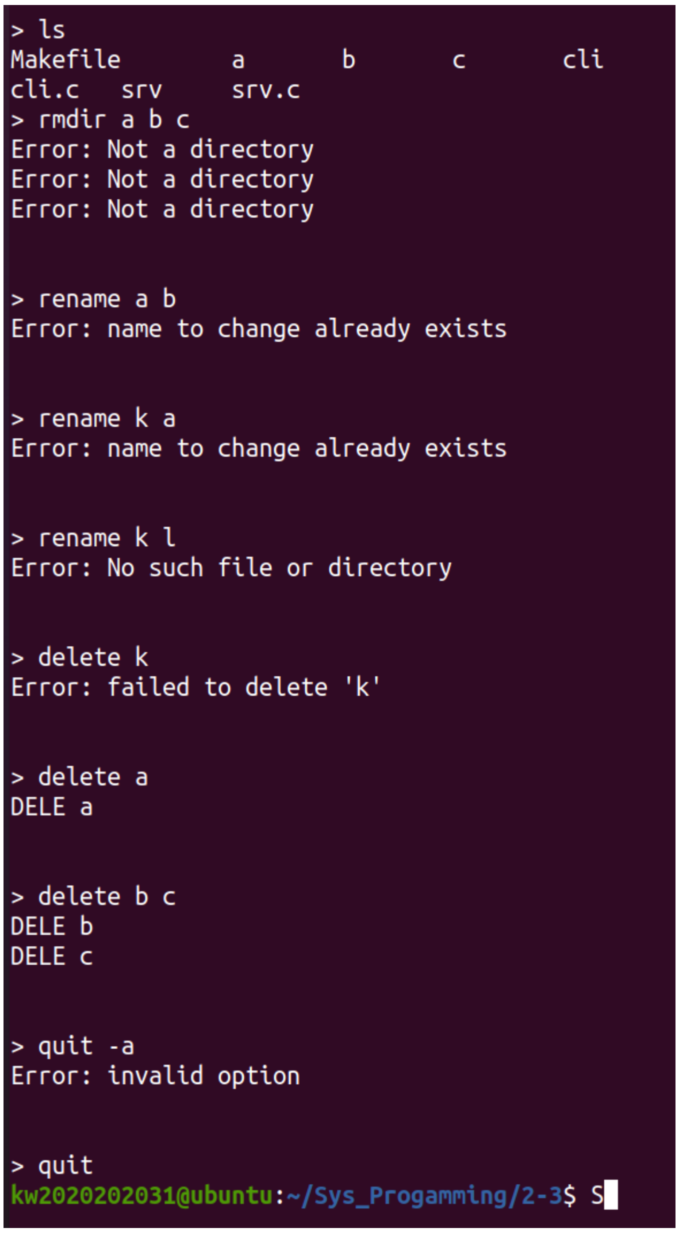
cd, pwd, ls, rmdir 다 잘 작동하는 것을 확인할 수 있습니다.

ls 에서 보이는 디렉토리 a b c 는 client [15410]에서 mkdir a b c 를 통해 생성한 것입니다.

rmdir a b c 로 a b c 디렉토리를 삭제했고, 이는 위 client [15410]에서 입력한 ls 명령어의 결과를 통해 확인할 수 있습니다.



```
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ ./cli 127.0.0.1 2000
> ls -a
./      ../     Makefile        a       b
c       cli     cli.c   srv     srv.c

> ls -asf
Error: invalid option


> dir
drwxrwxr-x   2  kw2020202031    kw2020202031     4096   May 15 12:26 ./
drwxrwxr-x  11  kw2020202031    kw2020202031     4096   May 15 07:38 ../
-rw-rw-r--   1  kw2020202031    kw2020202031      117   May 15 07:38 Makefile
-rw-rw-r--   1  kw2020202031    kw2020202031        0   May 15 12:26 a
-rw-rw-r--   1  kw2020202031    kw2020202031        0   May 15 12:26 b
-rw-rw-r--   1  kw2020202031    kw2020202031        0   May 15 12:26 c
-rwxrwxr-x   1  kw2020202031    kw2020202031    17496   May 15 07:41 cli
-rw-rw-r--   1  kw2020202031    kw2020202031     7042   May 15 07:38 cli.c
-rwxrwxr-x   1  kw2020202031    kw2020202031    36112   May 15 11:37 srv
-rw-rw-r--   1  kw2020202031    kw2020202031    33759   May 15 11:10 srv.c

> dir -asdf
Error: invalid option


> dir 1
Error: No such file or directory


> cd ../1--
Error: No such file or directory


> cd ..
CDUP
"/home/kw2020202031/Sys_Progamming" is current directory


> pwd
"/home/kw2020202031/Sys_Progamming" is current directory


> pwd -a
Error: invalid option


> cd 2-3
CWD 2-3
"/home/kw2020202031/Sys_Progamming/2-3" is current directory


> mkdir a b c
Error: cannot create directory 'a': file exists
Error: cannot create directory 'b': file exists
Error: cannot create directory 'c': file exists
```


```
> ls
Makefile        a       b       c       cli
cli.c   srv     srv.c
> rmdir a b c
Error: Not a directory
Error: Not a directory
Error: Not a directory


> rename a b
Error: name to change already exists


> rename k a
Error: name to change already exists


> rename k l
Error: No such file or directory


> delete k
Error: failed to delete 'k'


> delete a
DELE a


> delete b c
DELE b
DELE c


> quit -a
Error: invalid option


> quit
kw2020202031@ubuntu:~/Sys_Progamming/2-3$ S
```

client 에서 잘못된 명령어를 입력했을 때 결과입니다.

어떠한 명령어든 잘못된 옵션이 들어가있으면 invalid option 에러를 출력합니다.

존재하지 않는 디렉토리로의 cd, dir, ls 명령어는 No such file or directory 에러를 출력합니다.

이미 존재하는 파일 or 디렉토리 이름으로 mkdir 시도는 file exists 에러를 출력합니다.

이미 존재하는 파일명으로 rename 하려고 하면 already exists 에러를 출력하고,

존재하지 않는 파일을 rename 하려고 하면 No such file or directory 에러를 출력합니다.

존재하지 않는 파일을 delete 하려고 하면 failed to delete 에러를 출력합니다.

고찰

이번 2-3 과제는, 과제 1-3 에서 구현한 다양한 명령어에 대한 server 동작을 과제 2-2 에서 구현한 소켓 통신에 합치는 과제였습니다.

자연스럽게 과제 1-3 에서 구현한 cmd_process 를 다시 쓰게 됐습니다. 하지만 1-3 과제에서는 동작 결과를 terminal 에 출력만 하면 됐던 반면, 2-3 과제에서는 동작 결과를 buffer 에 write 하여 client 로 보내줘야 하기에, 약간의 수정이 필요했습니다.

수정하는 과정에서 많은 오류를 경험했고, 오류를 하나하나 수정하는 과정에서 코드의 직관성의 중요함을 깨닫게 됐습니다.

Reference

시스템프로그래밍 / 광운대학교 / 최상호 교수님 / 5._SP_-_Process_control

시스템프로그래밍 / 광운대학교 / 최상호 교수님 / 6._SP_-_Sockets

시스템프로그래밍 / 광운대학교 / 최상호 교수님 / 8._SP_-_Signals

시스템프로그래밍 / 광운대학교 / 최상호 교수님 / 2024-1_SPLab_07_FTP2_3

시스템프로그래밍 / 광운대학교 / 최상호 교수님 / 2024-1_SPLab_FTP_Assginment2_3_v2