System Programming Report

Assignment 4-2 – process pool management

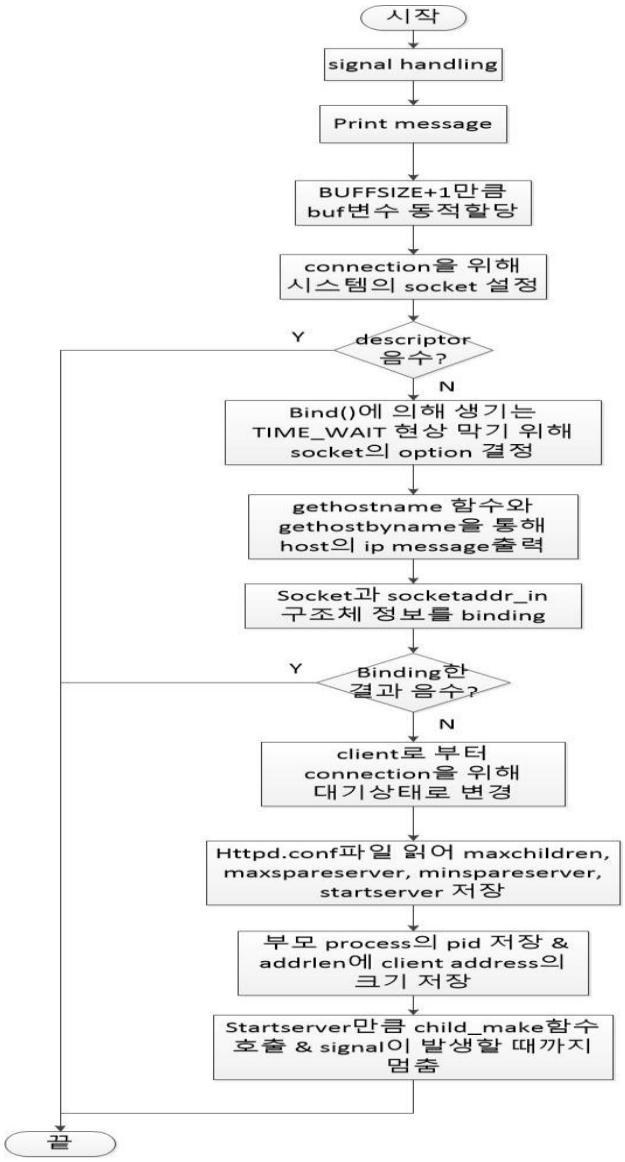
Professor	황호영 교수님
Department	Computer engineering
Student ID	2014722057
Name	김 진아
Class	설계 (화6 목4) / 실습 (금 56)
Date	2016. 6. 3

◆ **Introduction**

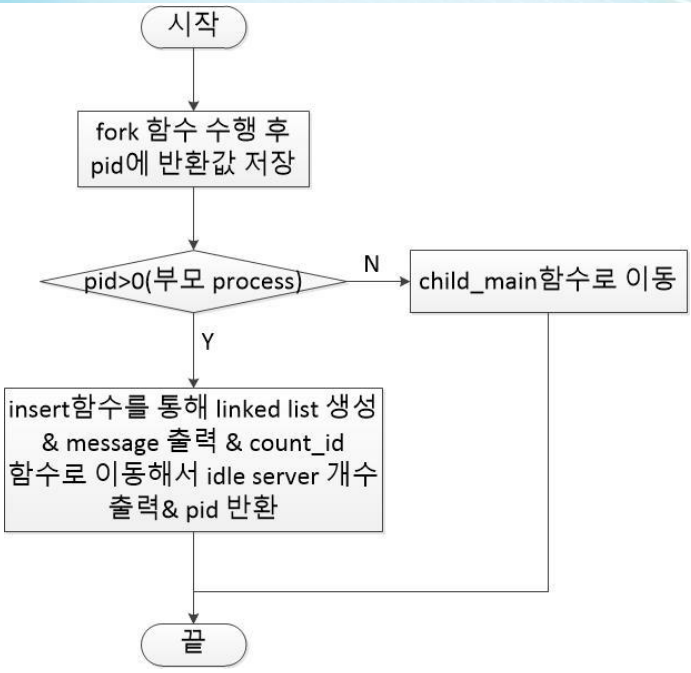
이번 과제에서는 4-1에서 구현한 code에 shared memory을 사용해 server와 client가 정보를 공유하도록 한다. shared memory에 접근하기 위해 thread를 생성하며 shared memory 동기화를 위해 pthread_mutex_lock, pthread_mutex unlock함수를 사용한다. shared memory를 이용해 child의 상태에 따른 idle server process의 수를 변경한다. idle process의 수가 4미만이거나 6초과하면 process를 생성하거나 종료하며 5개의 process가 되도록 한다.

◆ **Flowchart**

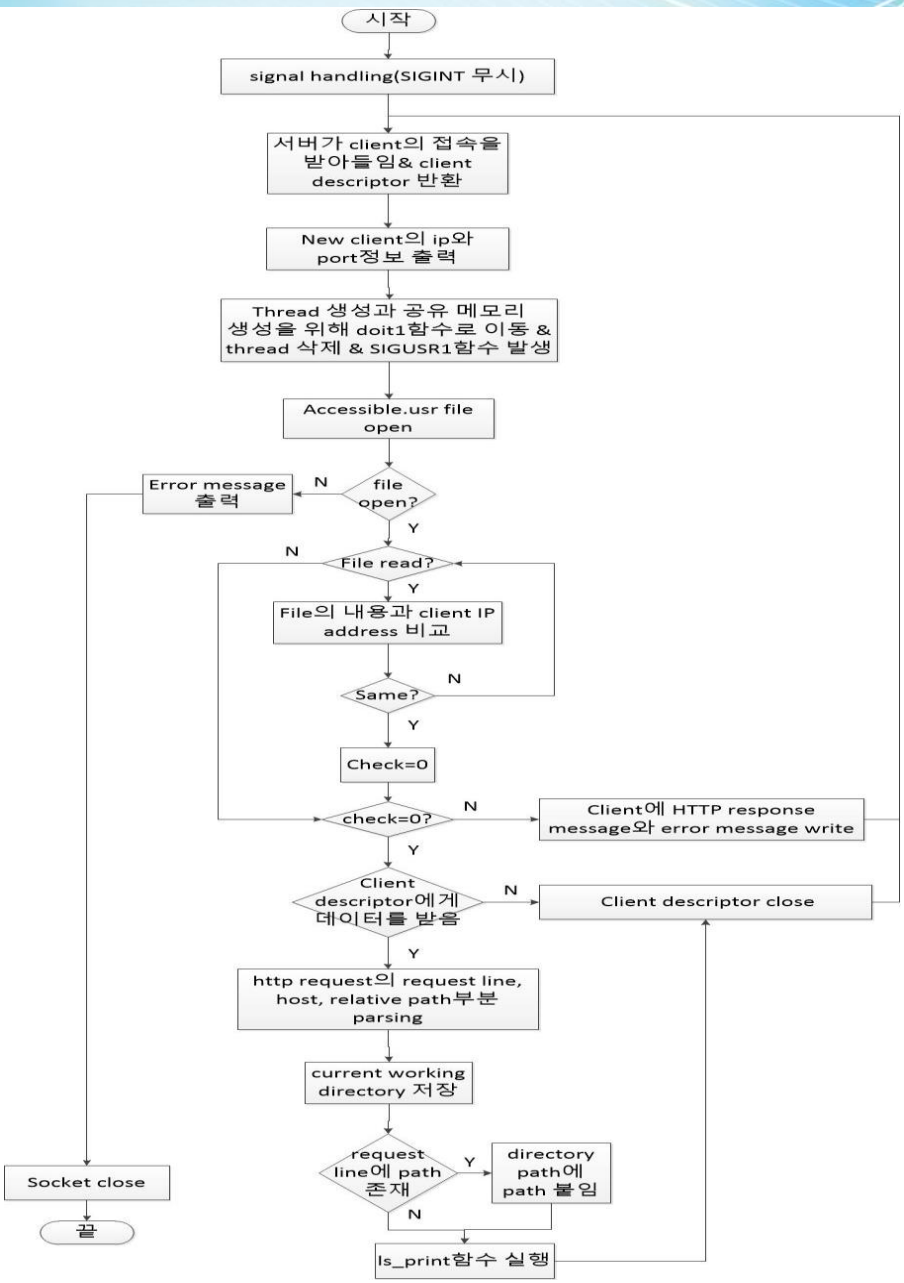
- main 함수



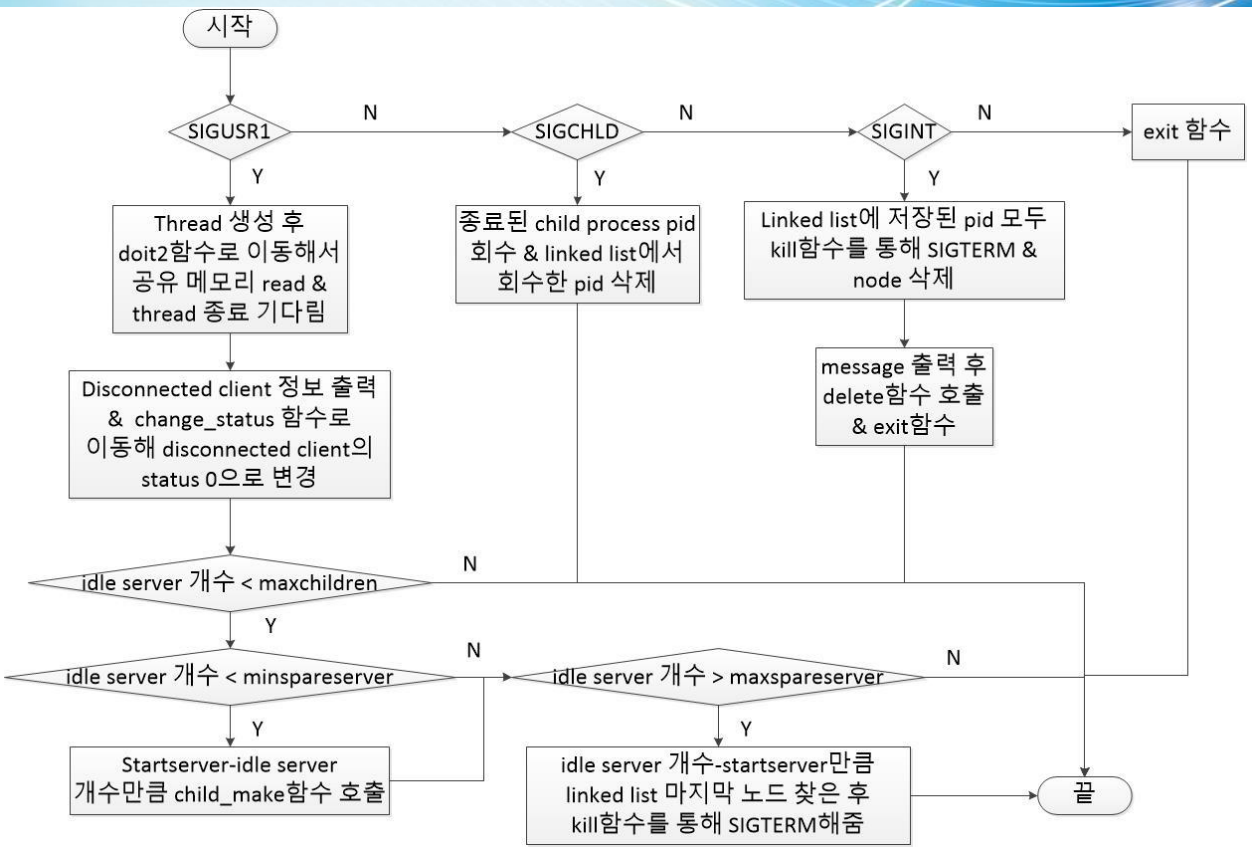
- child_make 함수



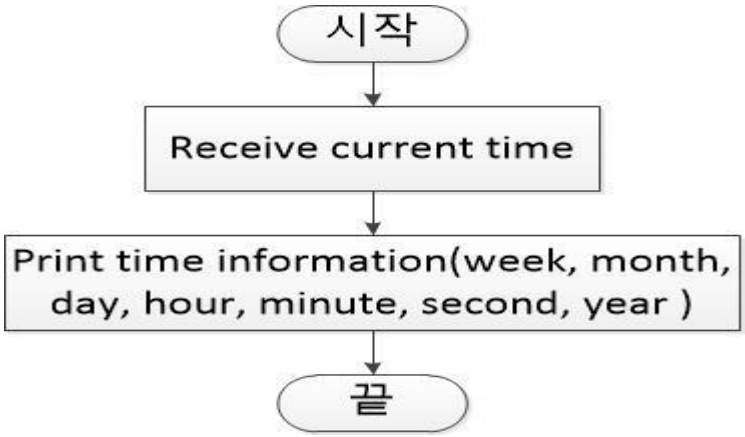
- child_main 함수



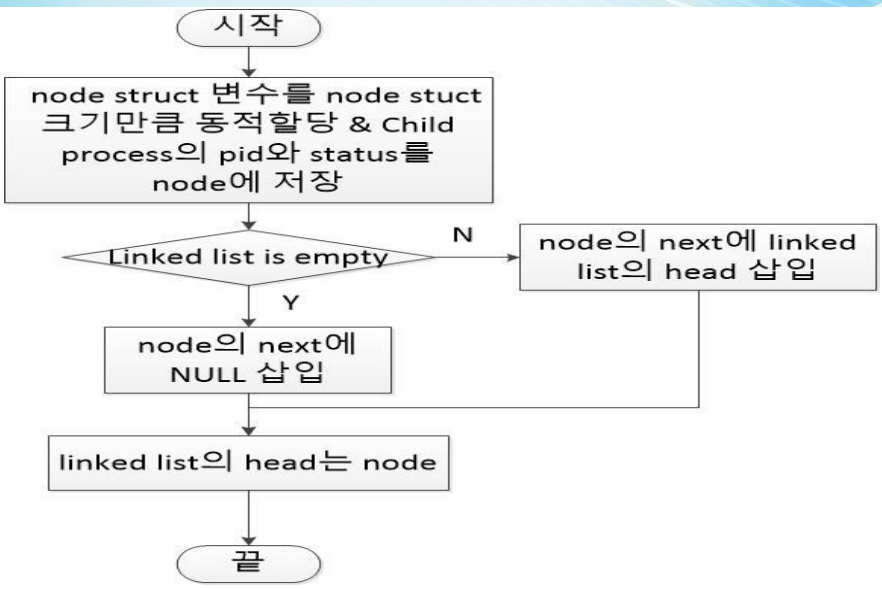
- signal handling 함수



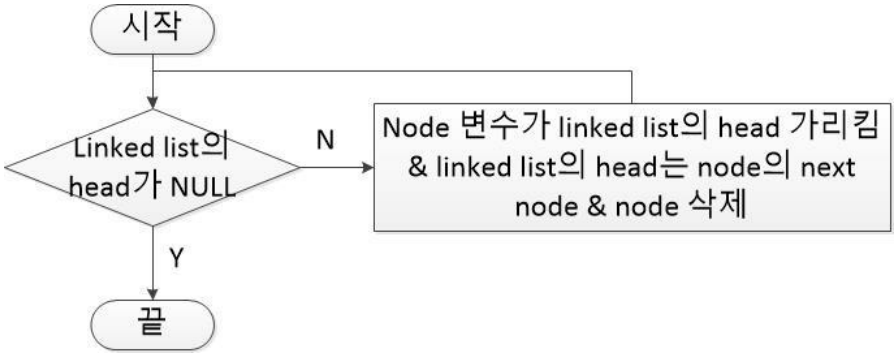
- print_t 함수



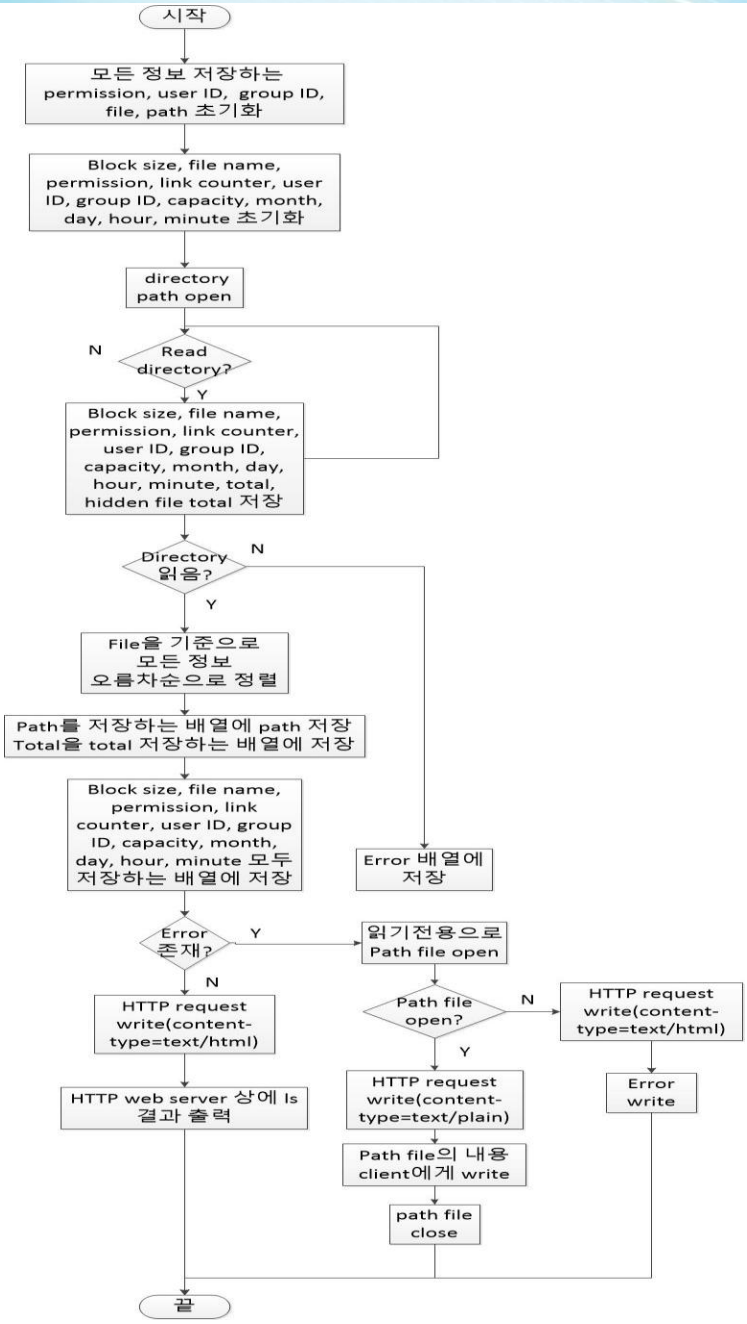
- insert 함수



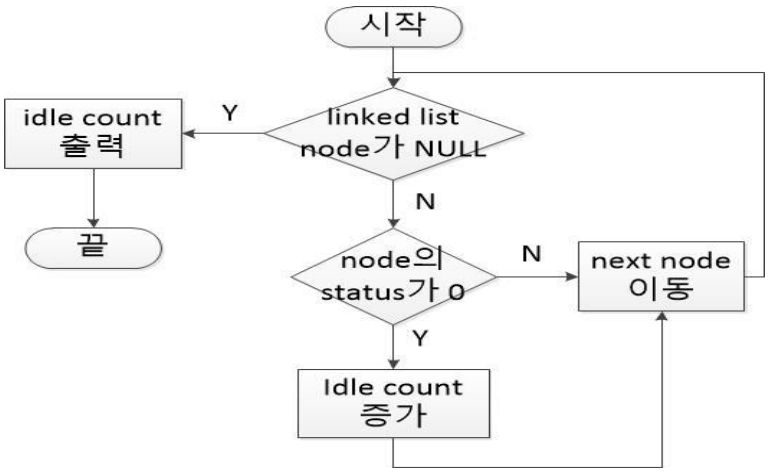
- delete 함수



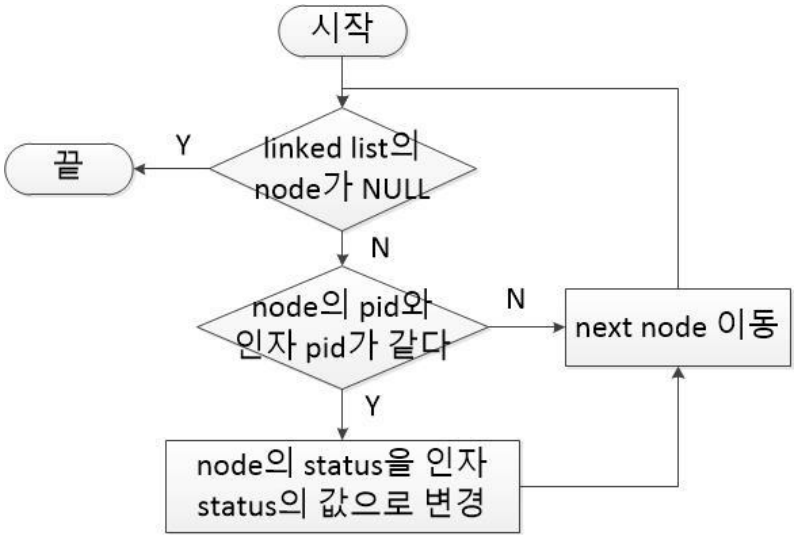
- ls_print 함수


시작
모든 정보 저장하는 permission, user ID, group ID, file, path 초기화
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute 초기화
directory path open
Read directory?
N
Y
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute, total, hidden file total 저장
Directory 읽음?
N
Y
File을 기준으로 모든 정보 오름차순으로 정렬
Path를 저장하는 배열에 path 저장 Total을 total 저장하는 배열에 저장
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute 모두 저장하는 배열에 저장
Error 배열에 저장
Error 존재?
Y
N
읽기전용으로 Path file open
HTTP request write(content-type=text/html)
Path file open?
N
Y
HTTP request write(content-type=text/html)
HTTP request write(content-type=text/plain)
Error write
HTTP web server 상에 ls 결과 출력
Path file의 내용 client에게 write
path file close
끝

- count_id 함수

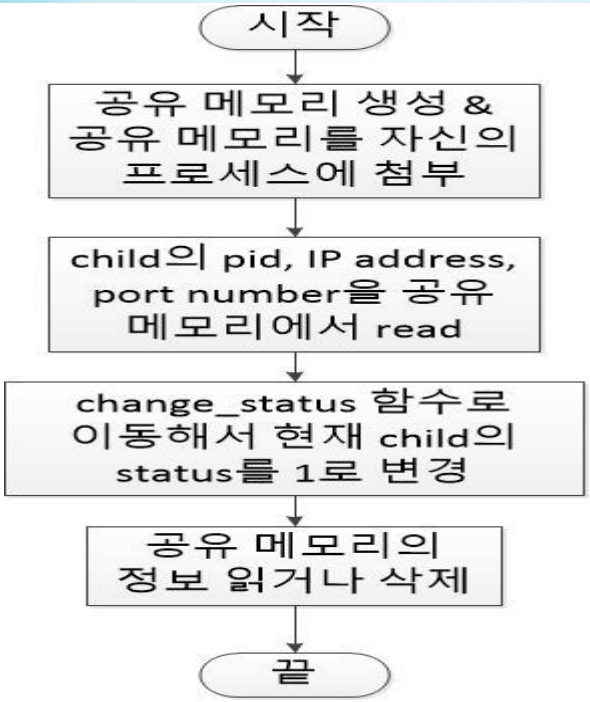

시작
linked list node가 NULL
Y
N
idle count 출력
끝
node의 status가 0
N
Y
next node 이동
Idle count 증가

- change_status 함수



- doit1 함수

- doit2 함수



◆ **Pseudo code**

- **main 함수**

signal handling

print time and message(server is started)

buf is dynamically allocated by BUFFSIZE+1

create a socket

if socket doesn't create{

print "Server: Can't open stream socket."

end of program

}

receive address family, IPv4 address, port number

use setsockopt function to block bind error

get host name and print time and message(socket is creadted. IP: host ip, port: port number)

associate an address with a socket

if socket doesn't bind{

print "Server: Can't bind local address.

```
        end of program
}
announce that server is willing to accept connect request
httpd.conf file open
if file doesn't exist{
        print no file message
        end of program
}
if file exists{
        while(read file){
                for(i=0;f_str[i]!='₩0';i++){
                        if new line character exists{
                                change new line character to null
                                stop for statement
                        }
                }
                token file's information
                if file's information is maxchildren
                        receive maxNchildren
                if file's information is maxspareserver
                        receive maxNspareserver
                if file's information is minspareserver
                        receive minNspareserver
                if file's information is startserver
                        receive startNserver
        }
```

```
        close file
    }
    receive parent process pid
    save client_address's size into addrlen
    save socket file descriptor
    save client's address length
    for(i=0;i<maxNchildren;i++)
        parent returns to use child_make function
    for(;;)
        pause program until signal is operated
    end of program
```

- **child_make 함수**

```
if result of fork is parent process{
        insertion function(make linked list)
        print time function
        print message(pid process is forked)
        go to count_id function to print idle child process's count
        parent move out
    }
    go to child process to excute child process
```

- **child_main 함수**

```
signal handling to ignore SIGINT
while(1){
        save client_address's size into clilen
        accept a connect request from client
        if it cannot accept{
```

```
        print "Server: accept failed.
        end of program
}
print new client's information
save client address
create thread and go to doit1 function to make shared memory
wait for thread to terminate
use kill to make SIGUSR1 signal
accessible.usr file open
if file doesn't exist{
        print no file message
        end of program
}
if file exists{
        while(read file){
                for(i=0;f_str[i]!='₩0';i++){
                        if new line character exists{
                                change new line character to null
                                stop for statement
                        }
                }
                compare client's IP address and IP address in file
                if client's IP address and IP address in file are same{
                        stop for statement
                }
        }
```

```
                close file
        }
        if client's IP address and IP address in file are different{
                write HTTP response message and error message at client descriptor(content
type is text/html)
                close client descriptor
                continue
        }
        if it can read HTTP request message{
                close socket descriptor
                if favicon.ico message operates{
                        close client file descriptor
                        continue
                }
                initialize host, version, temp
                write HTTP request message
                find GET / HTTP/1.1 in HTTP request message
                find Host in HTTP request message
                find temp in HTTP request message
                if temp's last letter is '/'
                        temp's last letter is NULL
                get current working directory path
                if exist relative path
                        add relative path to current working directory path
                go to ls_print function
        }
```

```
        close client descriptor
        create thread and go to doit1 function to make shared memory
        wait for thread to terminate
        use kill to make SIGUSR1 signal
    }
close socket descriptor
```

- **signal handling 함수**

```
if SIGUSR1 is operated{
        create thread and go to doit1 function to make shared memory
        wait for thread to terminate
        if client is created{
                go to change_status function to change status of child process by 1
                go to count_id function to print idle server count
        }
        if client is disconnected{
                print disconnected client's information
                go to change_status function to change status of child process by 0
                go to count_id function to print idle server count
        }
        if the # of idle child process is less than the maximum # of child process{
                if the # of idle child process is less than the minimum # of idle child process{
                        save result of startNserver-idle to e
                        for(s=0;s<e;s++)
                                go child_make function
                }
                if the # of idle child process is more than the maximum # of idle child
```

```
process{
                        save result of startNserver-idle to e
                        for(s=0;s<e;s++){
                                find last node stored linked list
                                kill process to make SIGTERM signal
                                use usleep function
                        }
                }
        }
}
if child process exits{
        PID has child status by using waitpid function
        pNode and pCur are pHead
        while(pNode!=NULL){
                if pNode's pid and PID are same{
                        if pNode is pHead
                                pHead is pNode's next node
                        if pNode isn't pHead
                                pCur's next node is pNode's next node
                        print time
                        print message(pid process is terminated)
                        delete pNode
                        end of while statement
                }
                pCur is pNode
                pNode is pNode's next node
```

```
        }
    }
    if ctrl+c is used{
        pNew and pDel are pHead
        while(linked list's node isn't NULL){
            change node's status by 0
            pNew is pNew's next node
        }
        while(linked list's node isn't NULL){
            kill process to make SIGTERM signal
            print time and message
            pNew is pDel
            pDel is pDel's next node
            change linked list's head by pDel
            delete node
            go to count_id function
        }
        print time
        print message(Server is terminated)
        delete linked list
        exit process
    }
    if process are terminated
        exit process
```

- **print_t 함수**

```
    save current time
```

print time information(week, month, day, hour, minute, second, year)

- **insert 함수**

store child process id and child process status into node

if pHead(linked list's head) is NULL

node's next pointer is NULL

if pHead isn't NULL

pNode's next is pHead

pHead is pNode

- **delete 함수**

while(linked list's head isn't NULL){

pNode is pHead

pHead is pNode's next node

delete pNode

}

- **ls_print 함수**

open directory

if directory can read

while(read information in opened directory){

receive file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute, the number of 1K blocks, total

}

}

if directory read

for(k1=0;k1<index1;k1++){

for(k2=0;k2<index1-1;k2++){

receive two files' name

```
                    if files' name are same
                        continue for statement
                    calculate files' length
                    receive letters of files' name while two letter are different
                    if first file's character > second file's character
                        change file's position, permission's position, linkcounter's position, user ID's position, group ID's position, capacity's position, month's position, day's position, hour's position, minute's position, block's position
                }
            }
            save directory path and total
            save beginning information
            for(i=0;i<index1;i++,s_index++)
                save file, permission, user ID, group ID, block size, linkcounter, capacity, month, day, hour, minute
            save ending information
        }
        if directory path doesn't exist
            save current path into error array
        close directory
        if error exists{
            path file open for read only
            if path file doesn't open{
                write HTTP response message at client descriptor(content type is text/html)
                write error message at client descriptor
            }
```

```
        if path file opens{
                write HTTP response message at client descriptor(content type is text/plain)
                read path file's content and write file's content at client descriptor
                close path file
        }
        end of function
    }
    write HTTP response message at client descriptor(content type is text/html)
    write title and head at client descriptor
    for(i=0;i<s_index;i++)
        write result of 'ls –al' at client descriptor
```

- count_id 함수

```
    while(linked list's node isn't NULL){
        if pNodes's status is 0 (pNode is idle server)
            increase idle process counter
        pNode is pNode's next node
    }
    print information of idle server count
    save the # of idle process to global variable
```

- change_status 함수

```
    while(linked list's node isn't NULL){
        if pNode's pid and Pid are same
            change status
        pNode is pNode's next node
    }
```

- doit1 함수

create shared memory

if it can't create shared memory

print fail message

end of function

}

attach shared memory to process

if it can't attach shared memory to process

print fail message

end of function

}

lock mutex and sleep

write shared memory which is child process information

unlock mutex and sleep

end of function

- doit2 함수

create shared memory

if it can't create shared memory

print fail message

end of function

}

attach shared memory to process

if it can't attach shared memory to process

print fail message

end of function

}

read shared memory and token shared memory(pid of child process, IP address, port number)

go to change_status function to change status

read shared memory or remove shared memory

if it can't read shared memory or remove shared memory

print fail message

end of function

◆ **Conclusion**

이번 과제에서 과제에 대한 이해가 안돼 무엇을 shared memory에 써야 하는지 몰랐다. 그러다가 child process의 정보를 공유한다고 해 child process의 정보가 저장된 linked list을 shared memory로 하려고 했다. 그랬더니 NODE가 공유가 안됐다. 나중에는 child process의 pid, ip address, port number을 공유하도록 했다. disconnected client 정보를 어디다가 출력해야 되는지 몰라 못하다가 친구가 조교님께 질문해 disconnected client 정보를 연결된 client를 close한 뒤 출력하거나 출력을 안 해도 된다는 것을 알았다. 또한 client가 연결되면 status가 0에서 1로 변하고 close하면 1에서 0으로 바뀌어 계속 idle server count가 4, 5가 반복된다는 것을 알게 되었다. 이렇게 고친 뒤 봤더니 minspareserver보다 idle server count가 작거나 maxspareserver보다 idle server count가 클 때 process를 생성하거나 종료해야 되는데 이 부분이 필요치 않게 되었다. 나중에 이 조건이 잘되는지 확인하기 위해 close한 뒤 status가 1에서 0으로 바뀌는 부분을 주석처리하고 돌렸더니 조건이 만족됨을 볼 수 있었다.