System Programming Report

Assignment 4-3 – Mutual Exclusion

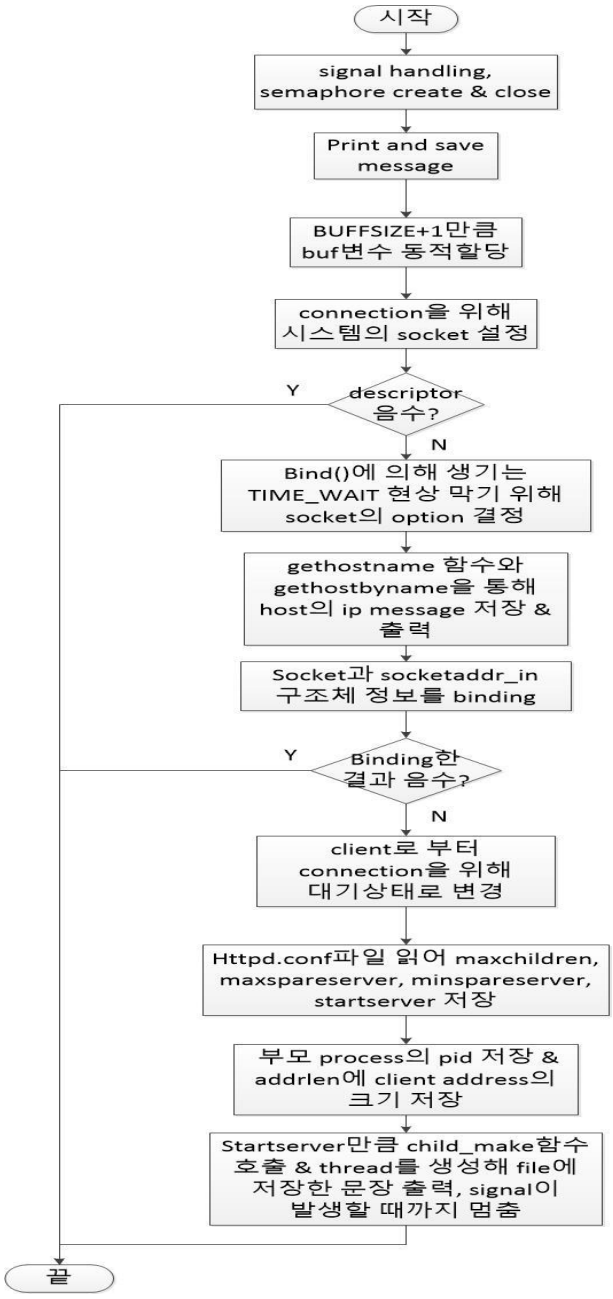
Professor	황호영 교수님
Department	Computer engineering
Student ID	2014722057
Name	김 진아
Class	설계 (화6 목4) / 실습 (금 56)
Date	2016. 6. 10

◆ **Introduction**

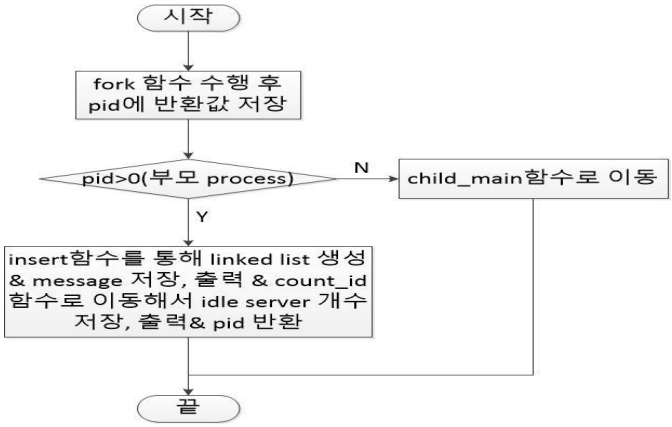
이번 과제는 process 관리 및 client 접속 정보를 파일에 기록하는 프로그램을 설계하는 것이다. 그래서 4-2과제에서 콘솔에 출력한 정보를 access.log 파일을 생성해 여기에 출력하는 것이다. client에 접속할 때 request 경로와 status code를 추가해 기록해준다. 요청이 성공하면 status code는 200 OK, 요청 실패하면 status code는 403 Forbidden이 된다. 동기화 문제 해결하기 위해 mutex 대신 semaphore을 사용한다. semaphore name은 할당된 port number로 한다.

◆ **Flowchart**

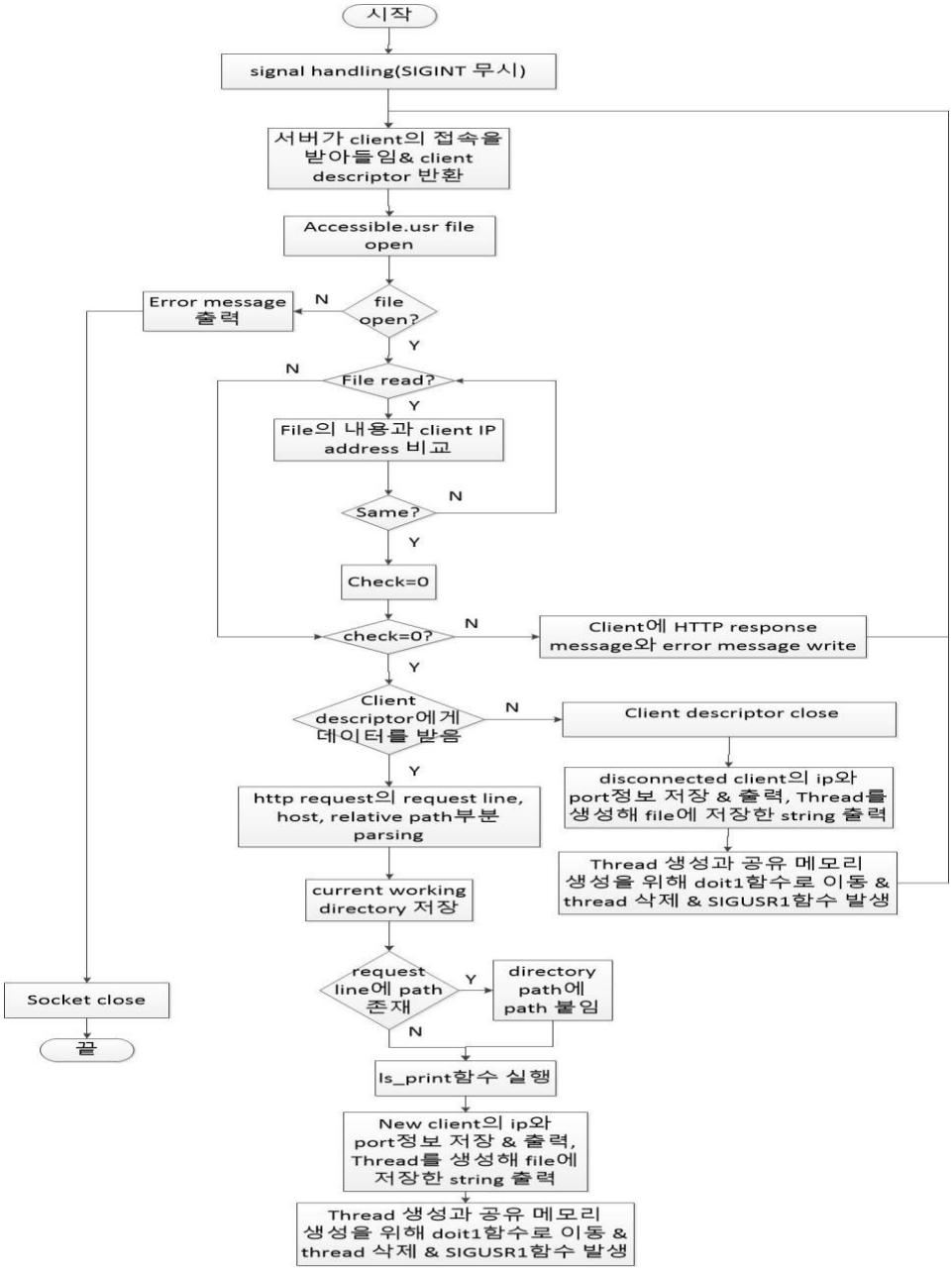
- main 함수



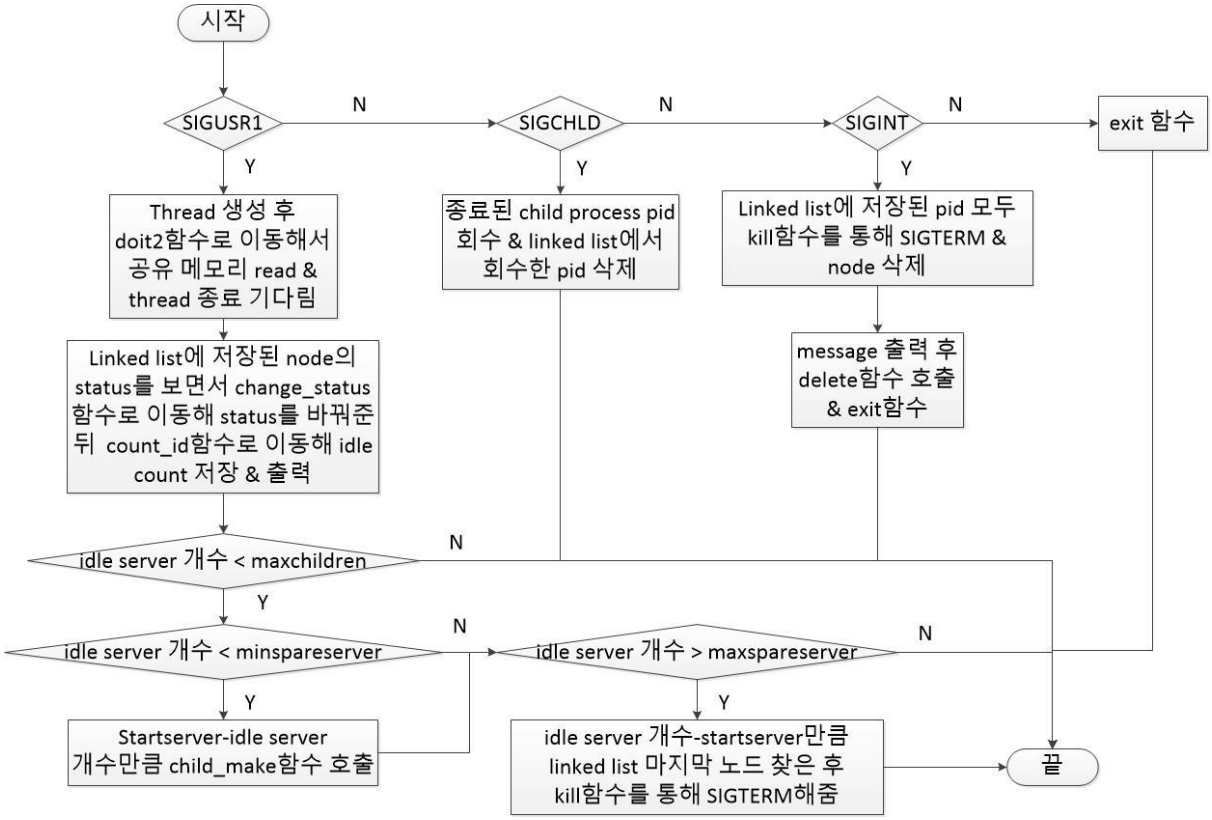
- **child_make 함수**



- **child_main 함수**

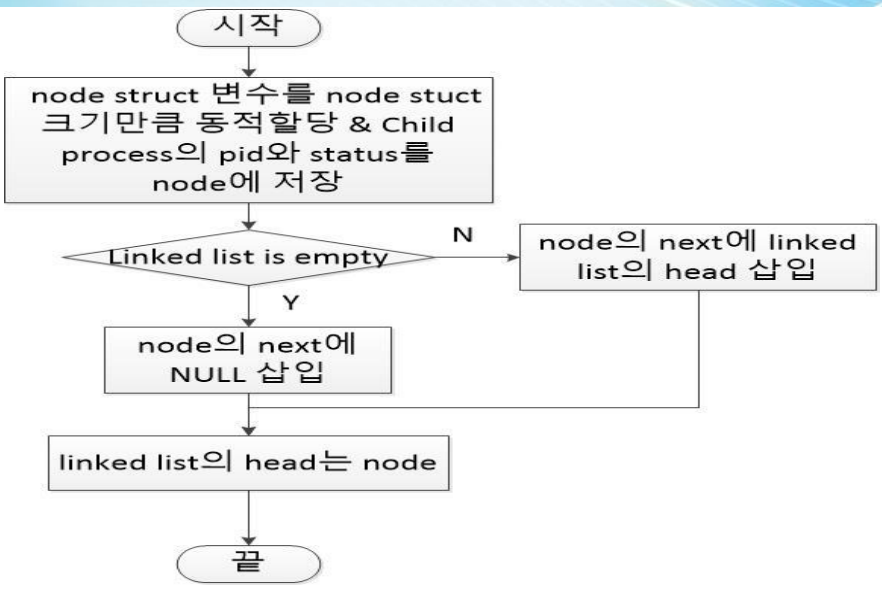


- signal handling 함수

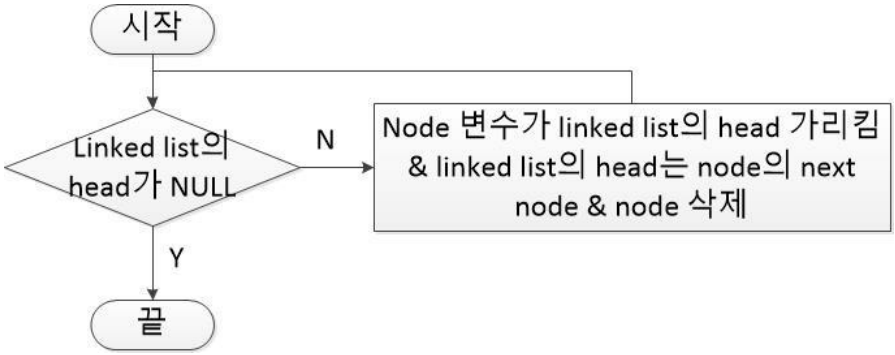


- print_t 함수

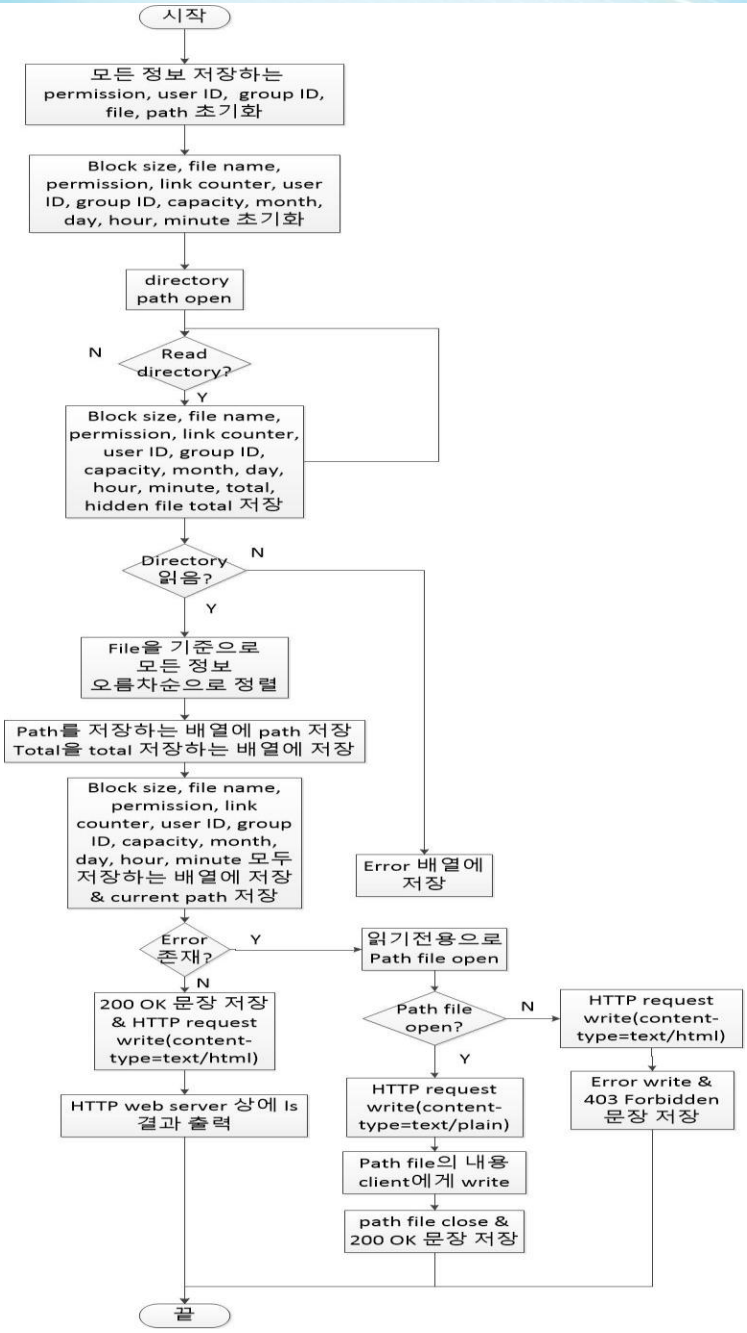
- insert 함수



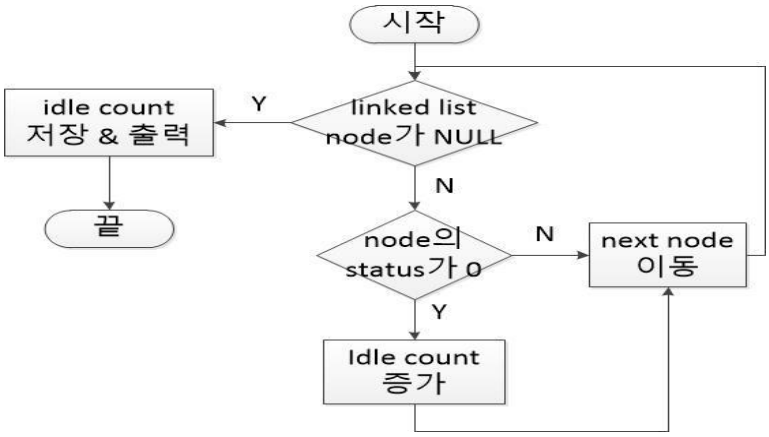
- delete 함수



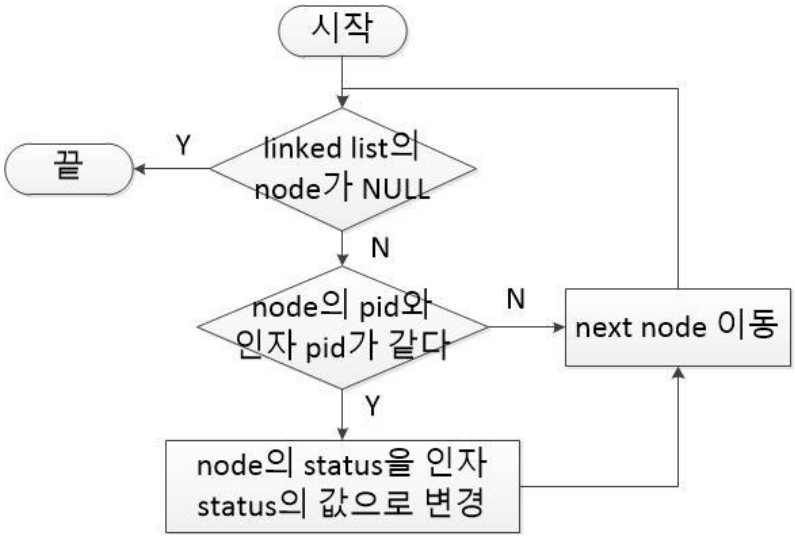
- ls_print 함수


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 & current path 저장
Error 배열에 저장
Error 존재?
Y
N
읽기전용으로 Path file open
200 OK 문장 저장 & HTTP request write(content-type=text/html)
Path file open?
N
Y
HTTP request write(content-type=text/html)
Error write & 403 Forbidden 문장 저장
HTTP web server 상에 ls 결과 출력
HTTP request write(content-type=text/plain)
Path file의 내용 client에게 write
path file close & 200 OK 문장 저장
끝

- count_id 함수

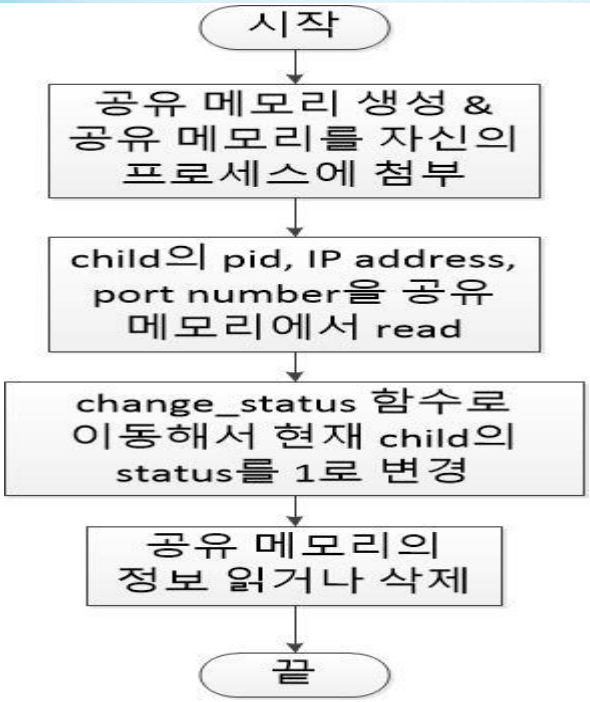

시작
linked list node가 NULL
Y
idle count 저장 & 출력
끝
N
node의 status가 0
N
next node 이동
Y
Idle count 증가

- change_status 함수

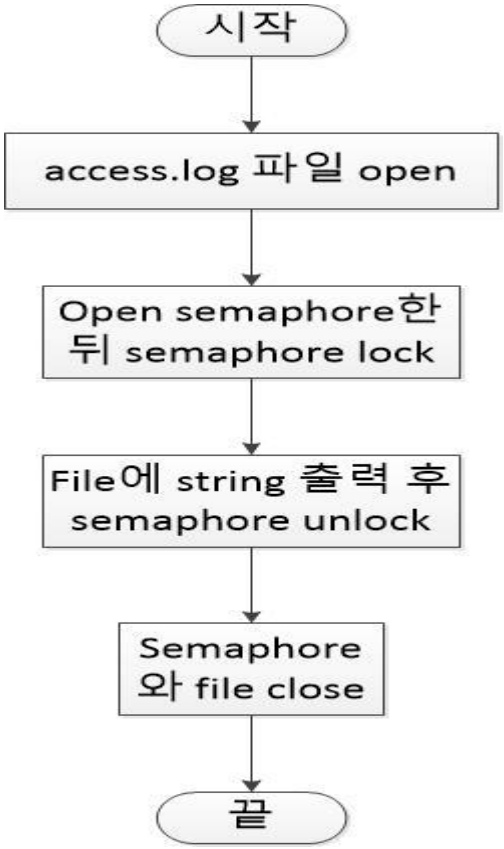


- doit1 함수

- doit2 함수



- doit3 함수



◆ **Pseudo code**

- main 함수

signal handling

open semaphore and close semaphore

print time and message(server is started) and save string

buf is dynamically allocated by BUFFSIZE+1

create a socket

if socket doesn’t create{

    print and save “Server: Can't open stream socket."

    create thread and go to doit3 function to save string in file

    wait for thread to terminate

    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 and save “Server: Can't bind local address.

    create thread and go to doit3 function to save string in file

    wait for thread to terminate

    end of program

}

announce that server is willing to accept connect request

httpd.conf file open

if file doesn't exist{

    print and save no file message

    create thread and go to doit3 function to save string in file

    wait for thread to terminate

```
        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
create thread and go to doit3 function to save string in file
wait for thread to terminate
for(;;)
        pause program until signal is operated
unlink semaphore
end of program
```

- **child_make 함수**

```
if result of fork is parent process{
        insertion function(make linked list)
        use time function to save and print time information
        print and save 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 execute 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
        }
```

```
save client address
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
        print and save new client's information
        create thread and go to doit3 function to save string in file
        wait for thread to terminate
        create thread and go to doit1 function to make shared memory
        wait for thread to terminate
        use kill to make SIGUSR1 signal
}
close client descriptor
```

print and save disconnected client's information

create thread and go to doit3 function to save string in file

wait for thread to terminate

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 and save idle server count

create thread and go to doit3 function to save string in file

wait for thread to terminate

}

if client is disconnected{

go to change_status function to change status of child process by 0

go to count_id function to print and save idle server count

create thread and go to doit3 function to save string in file

wait for thread to terminate

}

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
                        use time function to save and print time information
                        print and save message(pid process is terminated)
```

```
                delete pNode
                end of while statement
            }
            pCur is pNode
            pNode is pNode's next node
        }
        create thread and go to doit3 function to save string in file
        wait for thread to terminate
    }
    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
            go to count_id function
        }
        print time function to save and print time information
        print and save message(Server is terminated)
        create thread and go to doit3 function to save string in file
```

wait for thread to terminate

delete linked list

exit process

}

if process are terminated

exit process

- **print_t 함수**

save current time

print and save 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
    save current path
    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
                    save 403 Forbidden message
            }
            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
                    save 200 OK message
            }
            end of function
    }
    save 200 OK message
    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 and save 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 and save fail message

create thread and go to doit3 function to save string in file

wait for thread to terminate

end of function

}

attach shared memory to process

if it can't attach shared memory to process

print and save fail message

create thread and go to doit3 function to save string in file

wait for thread to terminate

end of function

}

open semaphore

lock semaphore and sleep

write shared memory which is child process information

unlock semaphore and sleep

end of function

- doit2 함수

create shared memory

if it can't create shared memory

print and save fail message

create thread and go to doit3 function to save string in file

wait for thread to terminate

end of function

}

attach shared memory to process

if it can't attach shared memory to process

print and save fail message

create thread and go to doit3 function to save string in file

wait for thread to terminate

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 and save fail message

create thread and go to doit3 function to save string in file

wait for thread to terminate

}

end of function

- doit3 함수

open file and save string

semaphore open

lock semaphore

print string in file

unlock semaphore

close file

close semaphore

◆ **Conclusion**

이번 과제에서 커널 창에 출력했던 모든 내용을 file에 써주기 위해 semaphore을 사용해야 됐는데 이를 위해 print할 때마다 print하는 문장을 저장한 뒤 thread를 생성하고 삭제해줘야 하는지 아니면 여러 개를 묶은 다음에 한번에 할지 고민이 되었다. 결국에는 함수에 있는 모든 print하는 문장을 저장한 뒤 한번에 file에 써지도록 했다. 이때 child_make함수에서 print하는 문장을 저장한 뒤 한번에 출력하게 했더니 fork해서 생성된 child의 pid가 순서대로 나오지 않고 띄엄띄엄 나오게 되었다. 그래서 이를 고치기 위해 child_make함수에서는 print하는 문장을 저장하도록 한 뒤 main함수에 신호가 발생하기 전까지 멈춰있는 구간 바로 위에 child_make함수에서 저장한 문장들 모두 file에 써지도록 했다. status code를 ls_print함수를 통해 저장한 뒤 thread를 이용해 부모와 자식이 정보를 공유하도록 했다. 그래서 sig_handler함수에서 disconnect client 정보를 출력하려 했는데 stack 오류와 abort core dumped 오류가 발생했다. 메모리 문제가 발생하는 것 같아 thread에 status code를 공유하지 않도록 하고 sig_handler에서 disconnected client 정보를 출력 안 하고 child_main함수에서 하도록 했다.