System Programming Report

Assignment 4-1 – Pre-forked Web Server

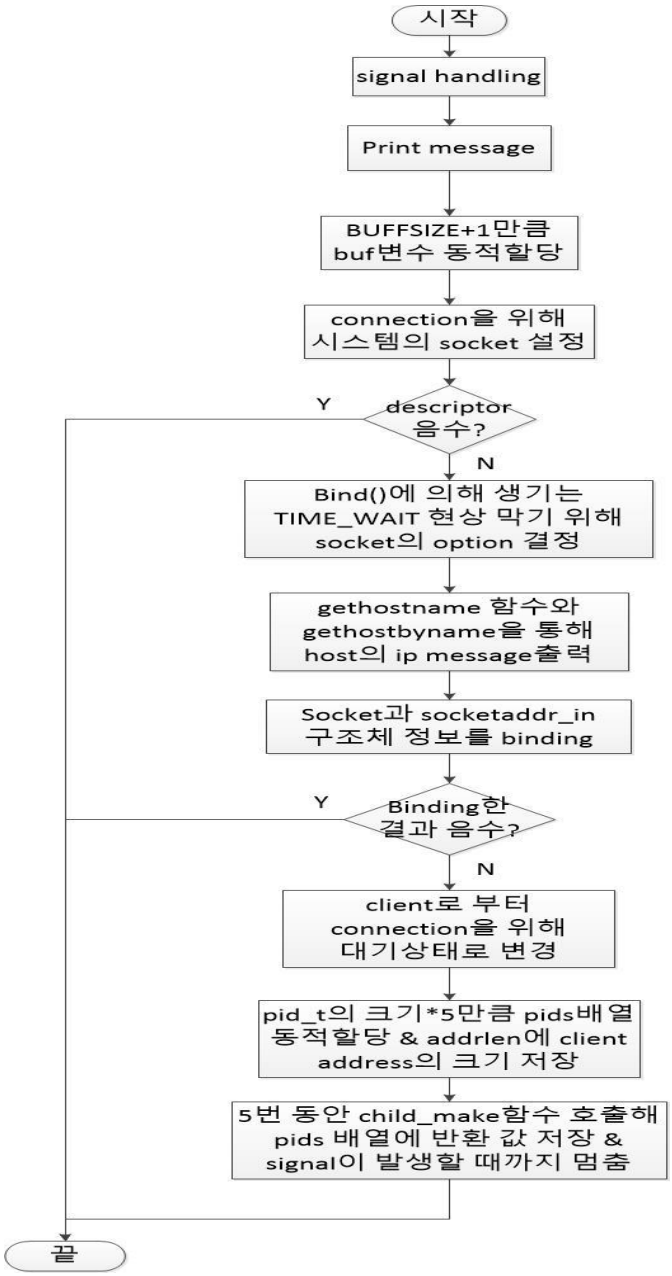
Professor	황호영 교수님
Department	Computer engineering
Student ID	2014722057
Name	김 진아
Class	설계 (화6 목4) / 실습 (금 56)
Date	2016. 5. 27

◆ **Introduction**

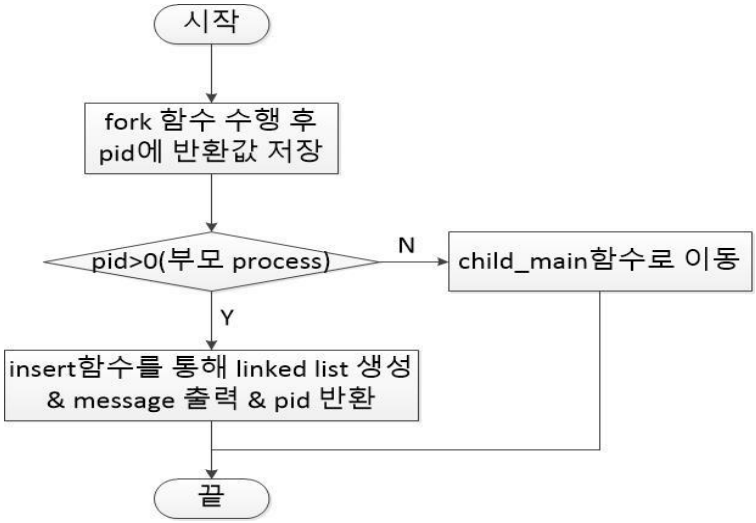
이번 과제는 저번 3-3과제에 조건을 추가해 pre-forked 방식으로 server를 구현하는 것이다. 5개 child process가 생성되도록 하며 child process의 pid를 linked list로 연결시켜 관리한다. SIGINT(Ctrl+c)을 발생시켜 모든 process가 종료되도록 한다. 예전 과제와 마찬가지로 parent process에서 child process를 관리하도록 하고 터미널에 간단한 로그 기록을 출력하도록 한다. 만약에 client process가 disconnected하면 fork를 실행시켜 새로운 child process를 만든다.

◆ **Flowchart**

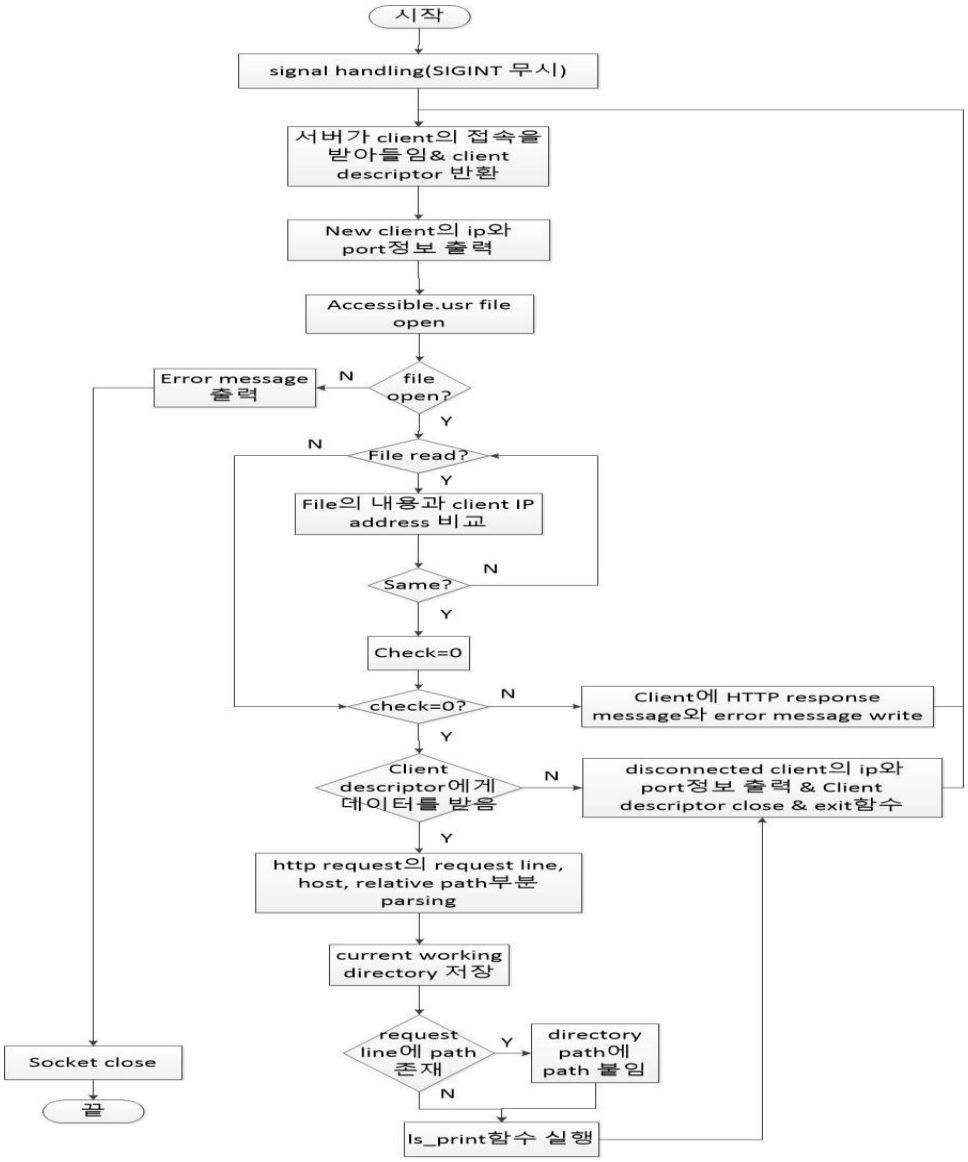
- main 함수



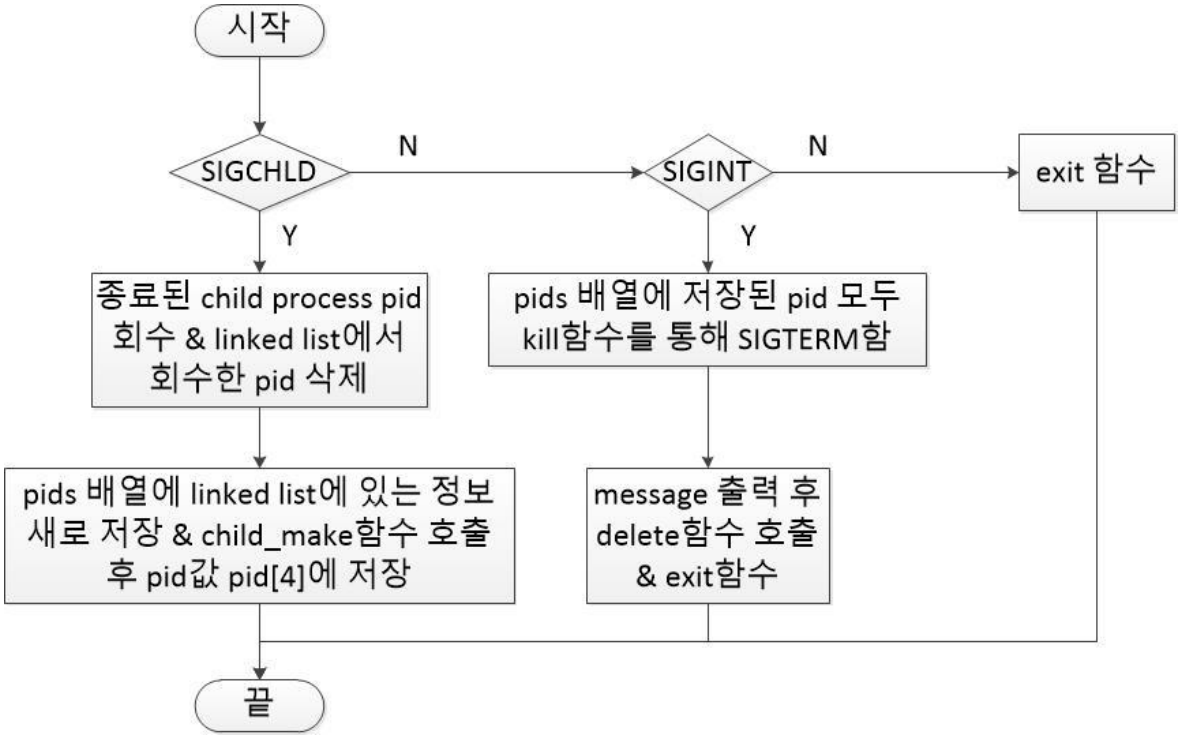
- **child_make 함수**



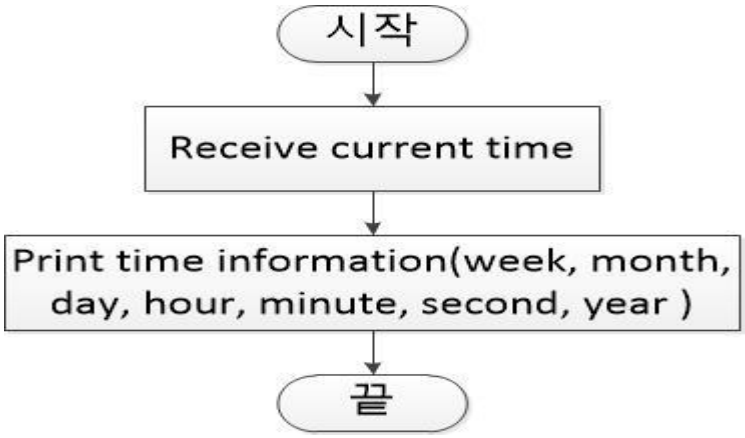
- **child_main 함수**



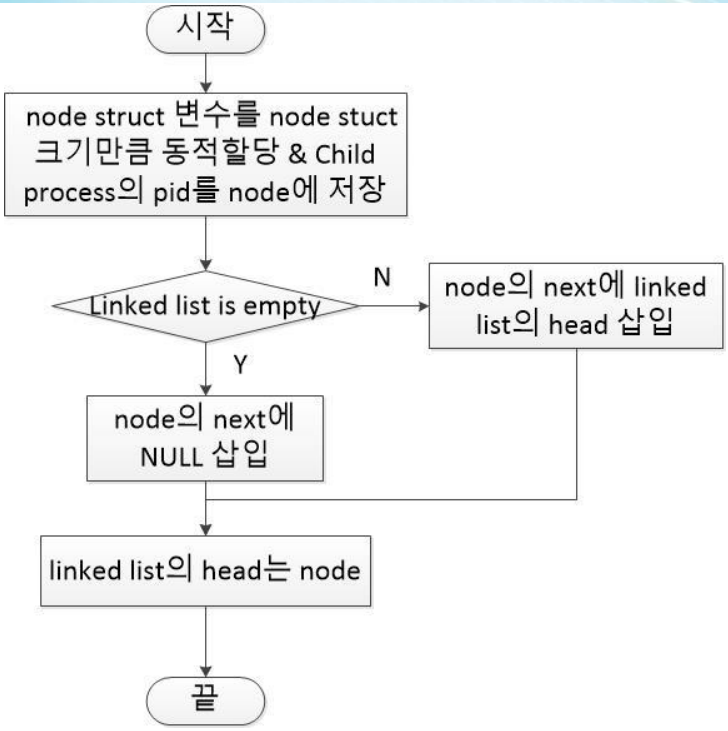
- signal handling 함수



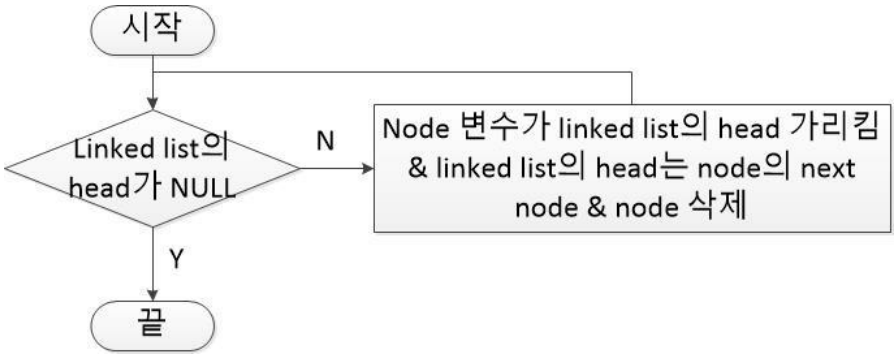
- print_t 함수



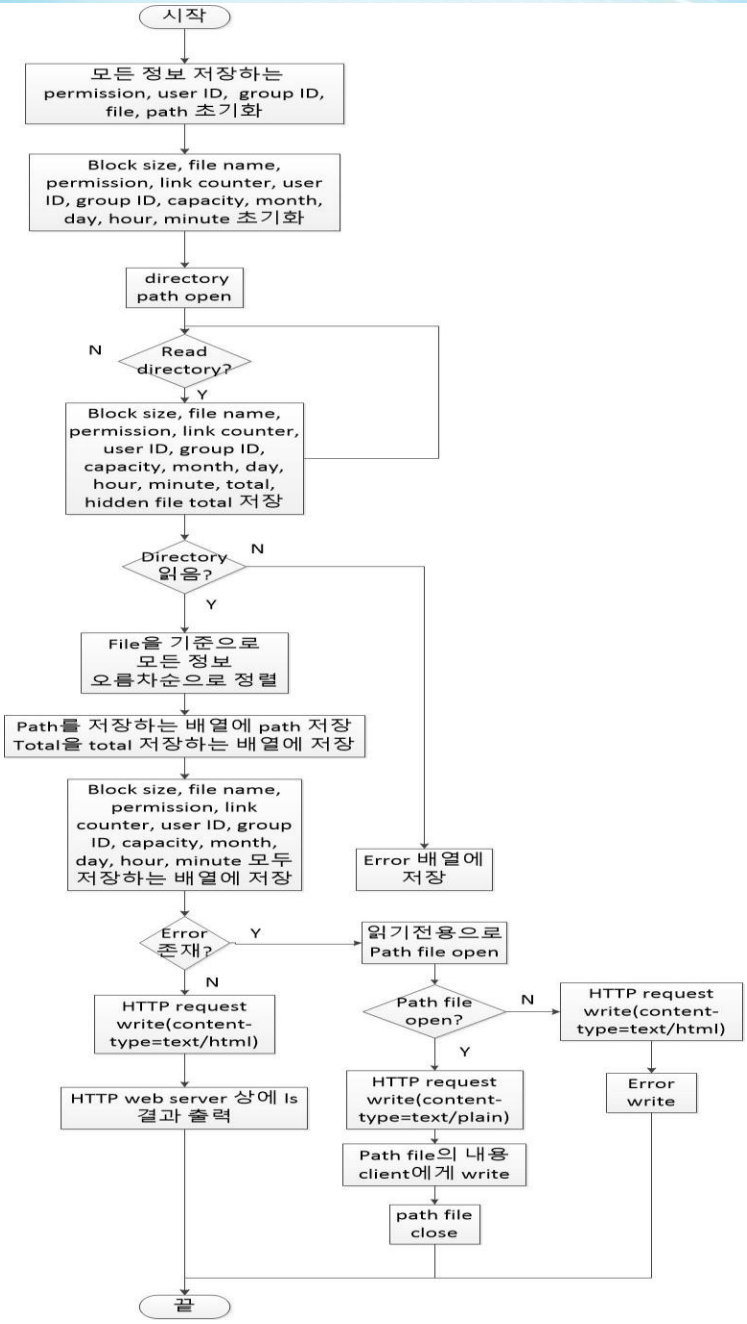
- insert 함수



- delete 함수



- ls_print 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
    maxNchildren is 5
    pids is dynamically allocated by maxNchildren*size of pid_tsave client_address's size
    receive parent process pid
    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)
            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
            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++){  // change new line character to null
                                                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 disconnected client's information
        exit client process
        close client descriptor
    }
close socket descriptor
```

- **signal handling 함수**

```
if child process exits{
        PID has child status by using wait 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
        }
        pNode is pHead
        while(pNode!=NULL){
                save pNode's pid into pids array
                pNode is pNode's next node
        }
        go child_make function and save pid into pids array
}
if ctrl+c is used{
        for(i=4;i>=0;i--){
                use kill function to execute SIGTERM
                print time
                print message(pid process is terminated)
        }
        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 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
```

◆ **Conclusion**

처음 과제에서는 linked list에 status의 정보를 저장해 client가 실행되면 이에 해당하는 child process가 저장된 node의 status가 변화하도록 구현하였다. 이때 부모의 정보와 자식의 정보가 공유가 안 되는데 된다 생각해 linked list의 status를 바뀌도록 했더니 하나도 바뀌지 않았다. 그래서 나중에 부모의 pid를 저장해 kill함수와 SIGUSR1을 이용해 부모 정보를 이용해 linked list의 status 정보가 변화하도록 했다. 그리고 new client와 disconnect client의 정보를 출력하기 위해 SIGUSR2을 이용해 출력되도록 했는데 안돼 고민하다가 status 정보 사용하지 않게 과제가 변해 이 부분을 넘어갔다. disconnected client 정보를 출력하고 다시 fork을 해 새로운 child를 만들어야 하는데 이 부분을 어떻게 해야 할지 몰랐다. 이를 해결하기 위해 client의 실행이 다 끝난 뒤 exit함수를 사용해 SIGCHLD가 발생하도록 했다. 그래서 sig_handler함수의 SIGCHLD부분에서 wait함수를 이용해 pid를 회수한 뒤 pid가 저장된 linked list node를 삭제했다. 그리고 나서 child_make 함수가 수행되도록 해 새로운 child가 발생하도록 했다.