System programming

Assignment 4-3 Mutual Exclusion

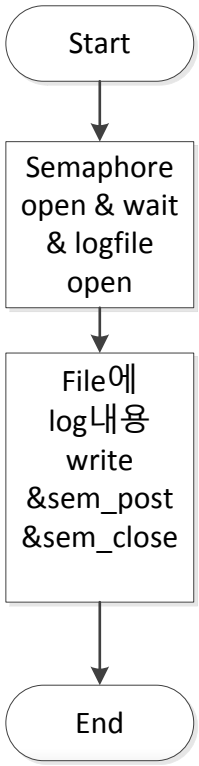
Professor	이기훈 교수님
Department	Computer engineering
Student ID	2014722046
Name	유지현
Class	설계(화6목5) /실습(목34)
Date	2016. 6. 10

1.Introduction

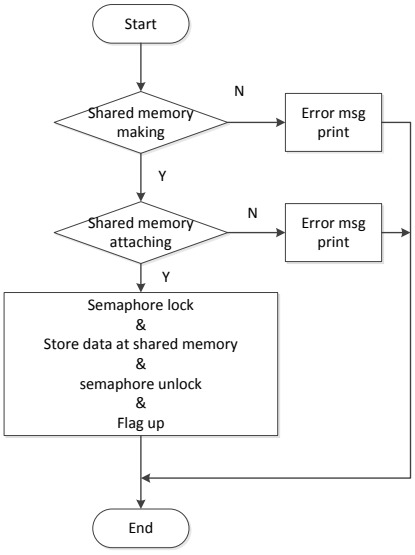
이번 과제는 지난번 과제에서 구현했던 것에서 printf를 이용하여 log내용을 출력했던 것을 logfile을 만들고 이에 log내용을 쓰는 것으로 변경하는 것이다. logfile에 log내용을 쓸 때 동기화 문제를 해결하기 위해 thread를 생성하고 thread함수에서 mutex semaphore를 이용하여 process 하나만 file에 쓰는 것을 수행 할 수 있도록 구현해야 한다.

2.flow chart

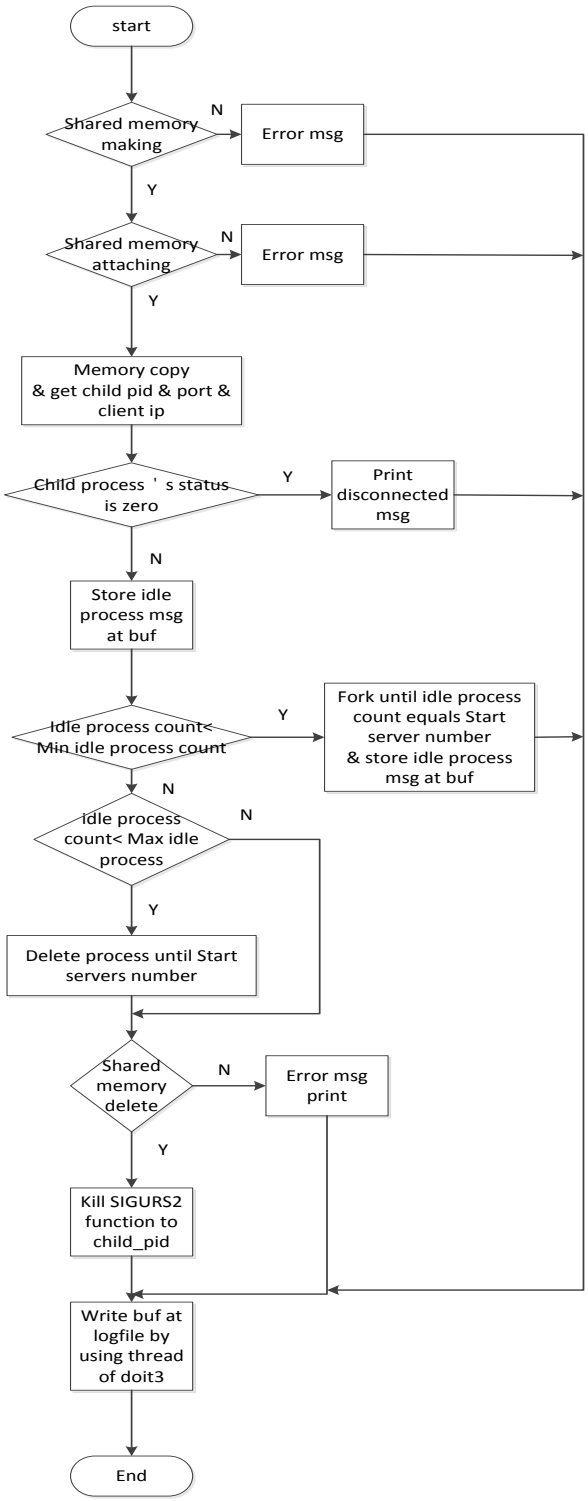
■doit3



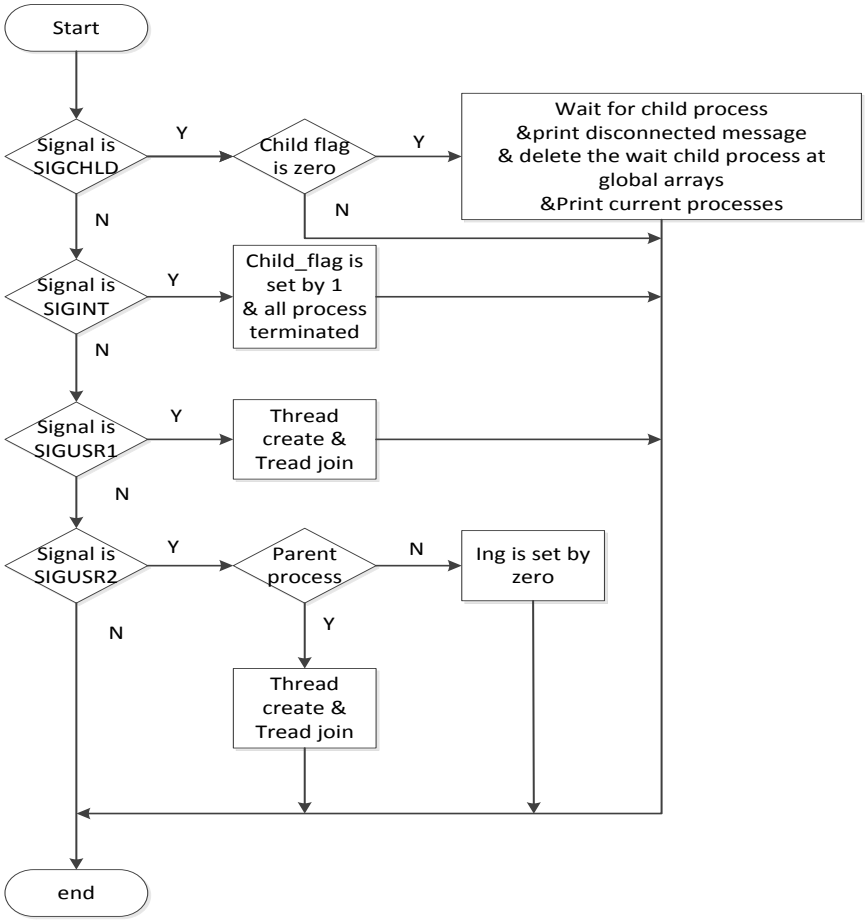
■doit1



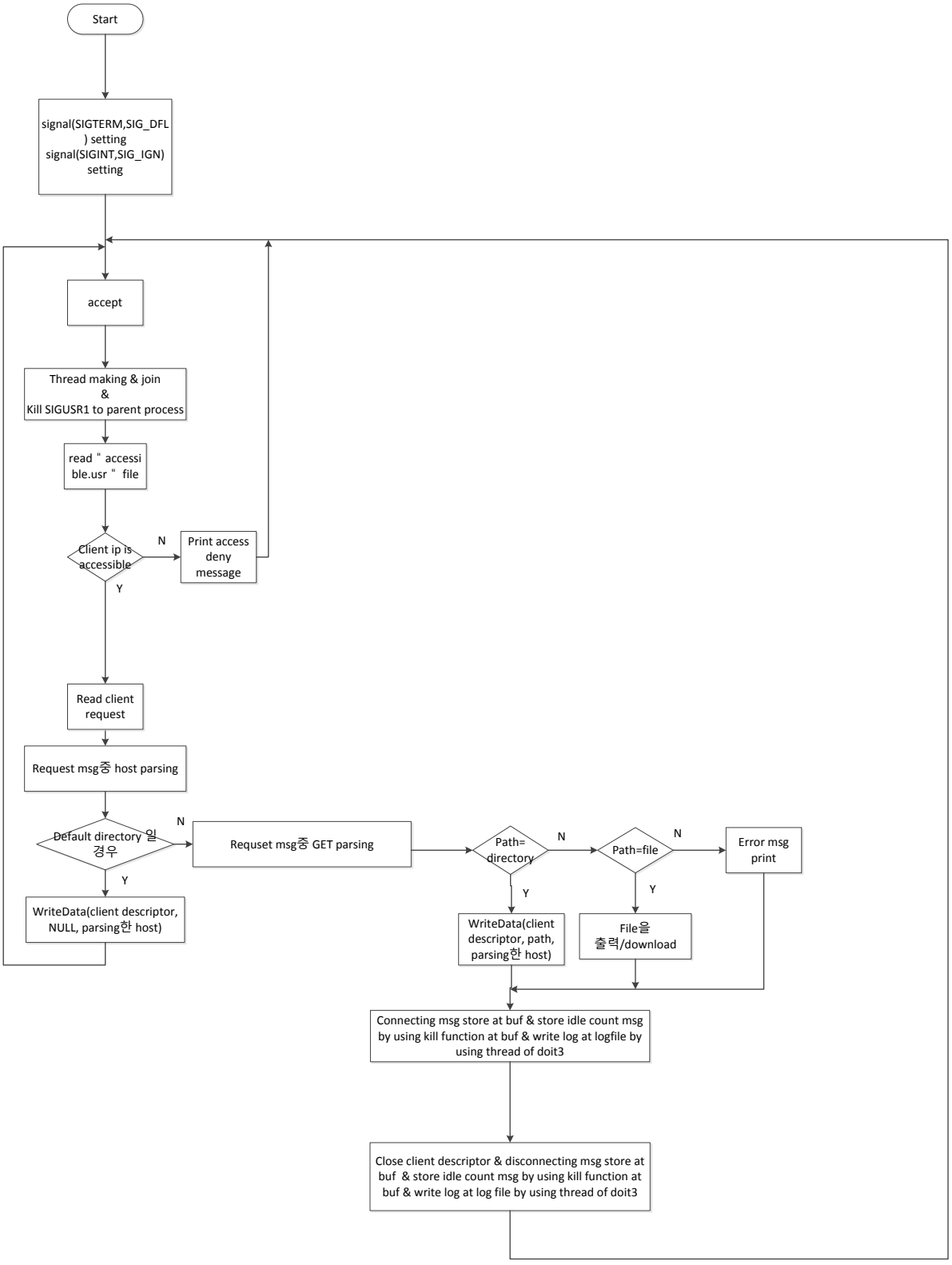
■doit2


start
Shared memory making
N
Error msg
Y
Shared memory attaching
N
Error msg
Y
Memory copy & get child pid & port & client ip
Child process ' s status is zero
Y
Print disconnected msg
N
Store idle process msg at buf
Idle process count< Min idle process count
Y
Fork until idle process count equals Start server number & store idle process msg at buf
N
idle process count< Max idle process
N
Y
Delete process until Start servers number
Shared memory delete
N
Error msg print
Y
Kill SIGURS2 function to child_pid
Write buf at logfile by using thread of doit3
End

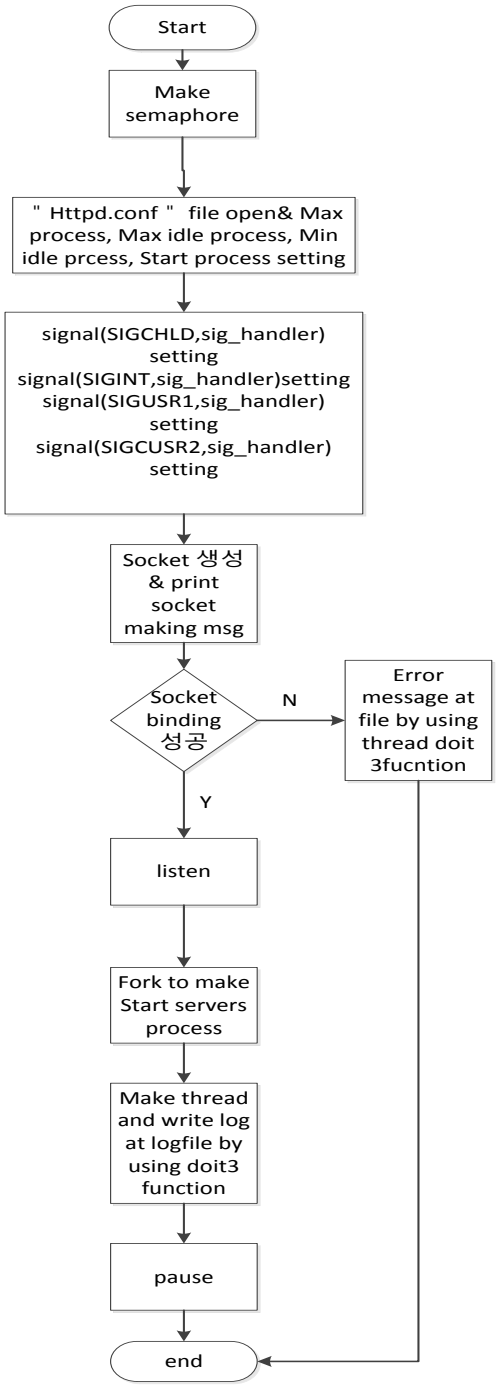
■sig_handler



■child_main


Start
signal(SIGTERM,SIG_DFL) setting signal(SIGINT,SIG_IGN) setting
accept
Thread making & join & Kill SIGUSR1 to parent process
read " accessible.usr " file
Client ip is accessible
N
Print access deny message
Y
Read client request
Request msg중 host parsing
Default directory 일 경우
N
Requset msg중 GET parsing
Path= directory
N
Path=file
N
Error msg print
Y
WriteData(client descriptor, NULL, parsing한 host)
Y
WriteData(client descriptor, path, parsing한 host)
Y
File을 출력/download
Connecting msg store at buf & store idle count msg by using kill function at buf & write log at logfile by using thread of doit3
Close client descriptor & disconnecting msg store at buf & store idle count msg by using kill function at buf & write log at log file by using thread of doit3

■main



3.Pseudo

■ doit3(void* buf)

semaphore open

semaphore lock

logfile open

wirte log at logfile

close file

semaphore unlock

close semaphore

■doit1(void *vptr)

if shmget fail
{
print error msg
return NULL
}

if shmat is fail
{
print error msg
return NULL
}

semaphore open
semaphore lock function
store child process pid & client_ip & port at shm_addr
semaphore unlock function
close semaphore
ing flag set
return null

■doit2

if shmet error
{
print error msg
return NULL
}

if shmat error
{
print error msg
return NULL
}

shared memory copy

for(i=0; memory[i]!='/' ;i++)
temp[i] is set by memory[i];

```
temp[i] is set by '₩0';
store i at temp_i

store child_pid

p is set by pHead
while(p)
{
        if p's pid equal child_pid
                break

        p is set by p's next node
}
 if p's stauts is zero
p's status is changed to 1
if   p's status is 1
p's status is changed to zero
temp reset
for(i=temp_i ; memory[i]!='/'; i++)
        temp[i-temp_i] is set by memory[i];
temp[i-temp_i] is set by '₩0';


temp_i is set by i+1;
for(i=0; temp[i]!='₩0'; i++)
        p->ip[i] is set by temp[i];
p->ip[i] is set NULL;

memset temp
for(i=temp_i ; memory[i]!='₩0'; i++)
        temp[i-temp_i] is set by memory[i];
temp[i-temp_i] is set by NULL
store port num
p's port is set by child_pid



idle count setting
time setting
```

```
print idle node count
temp_i is set by zero
p is set by pHead
while(p)
{
        p is set by p's next node
        temp_i is increased
}
if temp_i is smaller than Max child process && idle count is smaller than Min idle server
{
        num is set by S_server-cnt;
        for(i=0; i<num; i++)
        {
                child_pid is set by child_make return value
                insert
                store "process is forked" msg at buf
                store idleserver count msg at buf
                cnt is increased
        }

}
if temp_i is larger than Max_server
{
        cnt-=S_server;//cnt   is set by cnt - S_server
        for(i=0; i<temp_i; i++)
        {
                p is set by pHead;
                while(p)
                {
                        if p's status is zero
                        exit

                        pPrev is set by p
                        p is p's next

                }
                if p is pHead
                {
                        pHead is set by p's next
```

```
                        p's next is NULL
                }
                if p is not head
                {
                        pPrev's next is set by p's next
                        p's next is NULL

                }
                kill p node
                free p node
                idle count setting
                print idle count


        }


}


if   shtml error
{
        print error msg
}
make thread of doit3
join thread
kill SIGUSR2 signal to child process

return NULL
```

■sig_handler

```
if signal number is SIGCHLD
{
        if child_flag is zero
        {
                memset(print_time,0,100)
                store time at now
                wait child process
                print "process is terminated"
```

```
                print idle process number
                p is set by pHead
                i is set by zero
                while(p)
                {
                        p's pid is equal PID
                        break
                        pPrev is set by p
                        p is set by p's next
                        i is set by i+1
                }
                if p is pHead
                pHead is set by p's next
                if p isn't pHead
                pPrev's next is set by p's next

        }
}
if sig is SIGINT
{
        flag setting for SIGCHLD
        while(p){
                p is set by idle process
                p is set by p's next
        }
        while(p){
                send SIGTERM signal at
                wait
                time store at now
                store "process terminated" msg at buf
                p is set by p's next node
        }
        delete linkedlist
        time store at now
        currnet time copy at print_time
        store"Server is terminated" msg at buf
        store idle process number msg at buf
        make thread of doit3
        join
```

close socket
exit parent process
}
■child_main(int i, int socketfd, int addrlen)

set SIGTERM default
set SIGINT signal

while(1){

clilen is set by addrlen
accept
store client IP
store client port

"accessible.usr" file open

if file open fail
{
error message print
exit child process
}

while((end=fgets(file_buf,256,file))!=NULL)
{

for(b=0; b<file_buf[b]!='₩0'; b++)
{

if file_buf[b] is '₩n'
file_buf[i] is set by NULL
}

if file_buf and client IP match
break

```
            if not match
                        continue

}

if end is NULL, in short, client IP is denied IP
{
store response message of requestof client at client_buf
store response message of requestof client at client_buf
print client_buf at client descriptor
make html
print error message
store error IP at client_buf
print error IP
print error message
close client_fd
close file
continue
}

close file
time stroe


if read is larger than zero
            if error request
            {
                        close client_fd
                        continue
            }

            for(b=0; buf[b]!='/'; b++)
            reset host
            for(j=0; ; j++)
            {
                        for host parsing
                        {
                        for(a=0; buf[a+j+6]!='₩n';a++)
```

```
                {
                        store host parsing at host
                }
                host[a] is set by '₩0'
                break
                }


        }

        if not link
        {
                store response message of request of client at client_buf
                print client_buf at client descriptor
                reset client_buf
                Write function

        }
        if link exist
        {

                for(k=b+1; buf[k]!=' ' ;k++)
                        client_buf[k-i-1] is set by buf[k];
                client_buf[k-i-1] is set by ₩0
                reset client_buf
                get current working directory at client_buf
                client_buf + '/'
                client_buf + temp

                if client_buf is directory
                {
                        store response message of request of client at
client_buf

                        print client_buf at client descriptor
                        Write function
                }
                if client_buf is file
                {
```

store response message of requestof client at client_buf

print client_buf at client descriptor

while(fgets(file_buf,256,file)!=NULL)
{
write entity at client descriptor
}
close file
}
error path
{

write client_buf at client descriptor
make html
store error message at client_buf
write client_Buf at client descriptor

}

}

}
store new client msg at buf
thread making of doit3
thread join

kill SIGUSR1 signal to parent
close client descriptor
store disconnecting msg at buf
thread making of doit3
thread joining

kill SIGUSR2 signal to parent

```
        }
}
```

■main

```
        child flag is set by zero
        SIGCHLD signal setting
        SIGINT signal setting
        SIGUSR1 signal setting
        SIGUSR2 signal setting

        file open "httpd.conf"
        if file open error
        {
                print error msg
                return 0
        }

        while((end=fgets(file_buf,256,file))!=NULL)
        {

                for(i=0; file_buf[i]!=':' ; i++)
                        temp[i] is set by file_buf[i];
                temp[i] is set by '₩0';
                if Maxchilds
                {
                        reset temp array
                        for(i++ ; file_buf[i]!=' ' && file_buf[i]!='₩t'; i++)
                        a is set by i
                        for( ; file_buf[i]!='₩0' && file_buf[i]!='₩n'; i++)
                                temp[i-a] is set by file_buf[i]
                        temp[i-a] is set by '₩0'
                        M_child is set by atoi(temp)
                }
                if MaxSpareServers
                {
                        reset temp array
                        for(i++ ; file_buf[i]!=' ' && file_buf[i]!='₩t'; i++);
                        a is set by i
                        for( ; file_buf[i]!='₩0'&& file_buf[i]!='₩n'; i++)
                                temp[i-a] is set by file_buf[i]
```

```
                temp[i-a] is set by '₩0'
                Max_server is set by atoi(temp)
        }
        if MinSpareServers
        {
                reset temp array
                for(i++ ; file_buf[i]!=' ' && file_buf[i]!='₩t'; i++)
                a is set by i;
                for( ; file_buf[i]!='₩0'&& file_buf[i]!='₩n'; i++)
                        temp[i-a] is set by file_buf[i]
                temp[i-a] is set by '₩0'
                Min_server is stored by atoi(temp)
        }
        if StartServsers
        {
                reset temp array
                for(i++ ; file_buf[i]!=' ' && file_buf[i]!='₩t'; i++)
                a is set by i
                for( ; file_buf[i]!='₩0'&& file_buf[i]!='₩n'; i++)
                        temp[i-a] is set by file_buf[i]
                temp[i-a] is set by '₩0'
                S_server is set by atoi(temp);
        }
}

time setting & print
store "Server is started" msg at buf
PPID is set parent PID
pHead is set by NULL

if socket open error
{
        print error message
        return 0
}
get host name
get host ip

time setting
```

```
store Socket IP & port at buf
socket reset
set address format
s_addr setting by host to network
port seeting by host to network
opt is set by 1
 blocking bind error
if socket bind error, print error message
        return 0
listen
addrlen setting
for(i=0; i<S_server; i++){
        time setting
        pids is set by child_make return value
        store "process is forked" at buf
        store idle process count at buf
        insert
}

make thread of doit3
join thread
pause
```

4.Conclusion

이번에 과제를 구현할 때 어려웠던 점은 thread에 대한 함수를 하나 더 만들어서 thread내에서도 또 thread를 만드는 등 process의 흐름이 나뉘는 분기점이 많다는 것이었다. 흐름이 나뉘기 때문에 어느 시점에서 어떤 코드들이 수행될지 이해하는 것이 중요했다. 그리고 thread를 생성하여 file에 log내용을 기록해야 하는데 한줄씩 내용을 file에 그때그때 쓰기에는 분기점이 너무 많이 생기고 thread를 너무 많이 생성해야해서 복잡하고 어려워서 대신 각 함수마다, 혹은 나만의 단위를 설정하여 그 단위별로 buf에 log기록을 저장한 후 thread를 생성하여 logfile에 log기록을 저장하도록 구현했다.

그리고 file에 계속해서 이어 써야 하기 때문에 file을 오픈할 때 option "a"를 주어 file에 계속해서 이어쓸 수 있도록 구현하였다.

하지만 처음에는 process의 흐름을 명확하게 이해하지 못하여 logfile에 기록되는 순서가 올바르지 못하고 이상했는데 다시 차근차근 분기점부터 흐름을 파악하여 이 부분을 수정했다.