REPORT

[시스템프로그래밍

Assignment 3-3]

학과	컴퓨터정보공학부

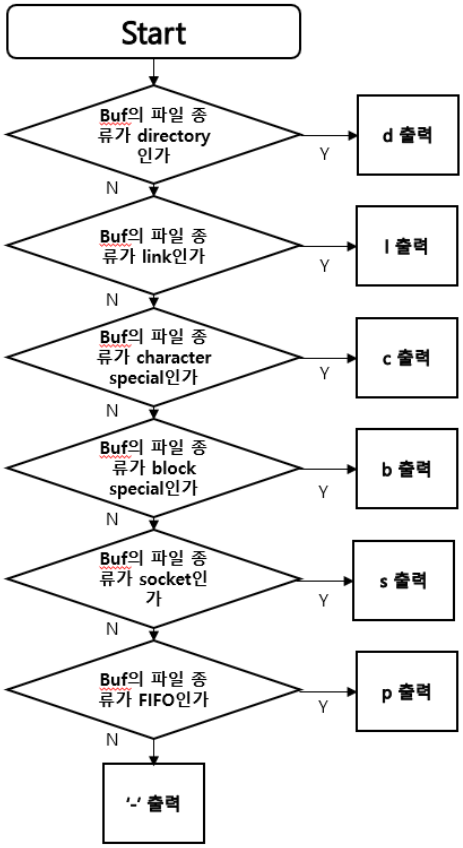
학번	2018202018
이름	유승재
담당 교수	김태석 교수님
제출일	2023.05.31

1. Introduction

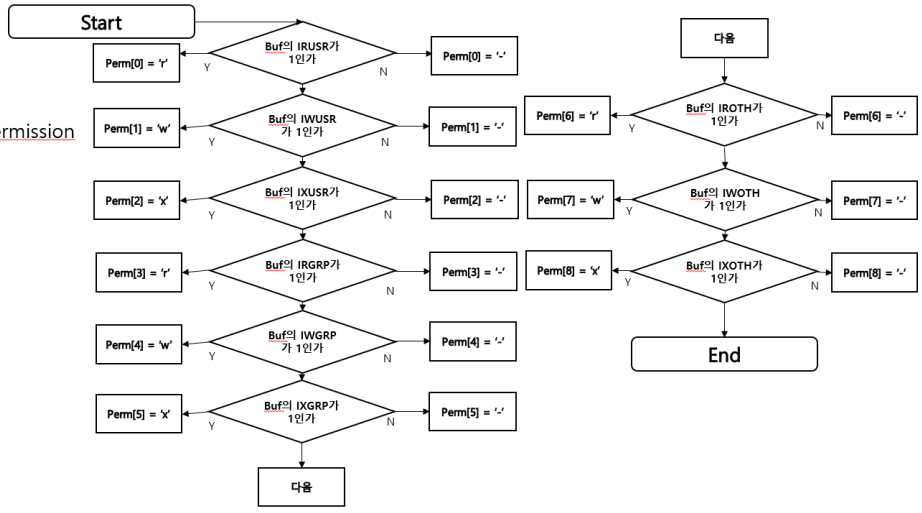
이번 과제는 이전 과제에서 client 가 요청한 경로(URL)과 Client 와 Server 가 연결된 지속 시간을 추가로 출력해주는 것과 지금까지 터미널에만 출력하던 Server 와 Client 간의 연결 시에 출력되는 정보들을 server_log.txt 파일에 추가로 작성해주는 기능을 구현하는 것이 구현의 목표입니다. 또한 server_log.txt 파일에 여러 thread 가 동시에 접근할 경우 파일의 동기화 문제가 발생할 가능성이 생기므로 semaphore 를 통해 한 번에 한 thread 만 접근이 가능하도록 설정하여 파일이 의도한 대로 작성되도록 구현해야합니다.

2. Flow Chart

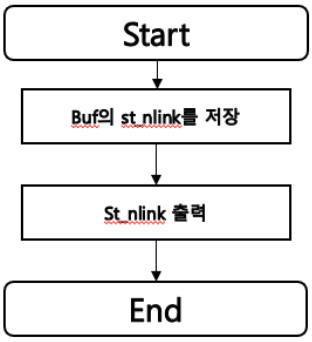
File_check



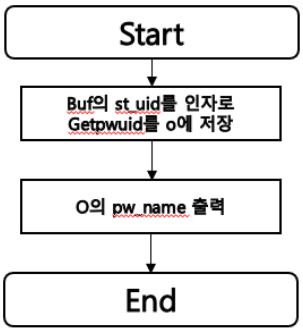
File_permission



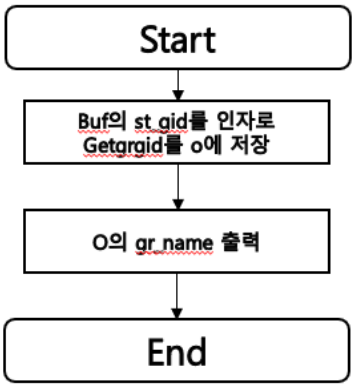
File_link



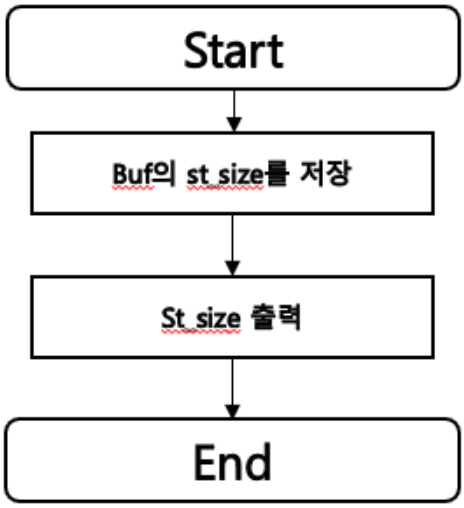
File_owner_name



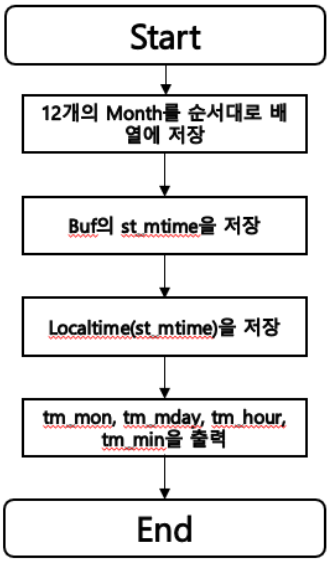
File_group_name



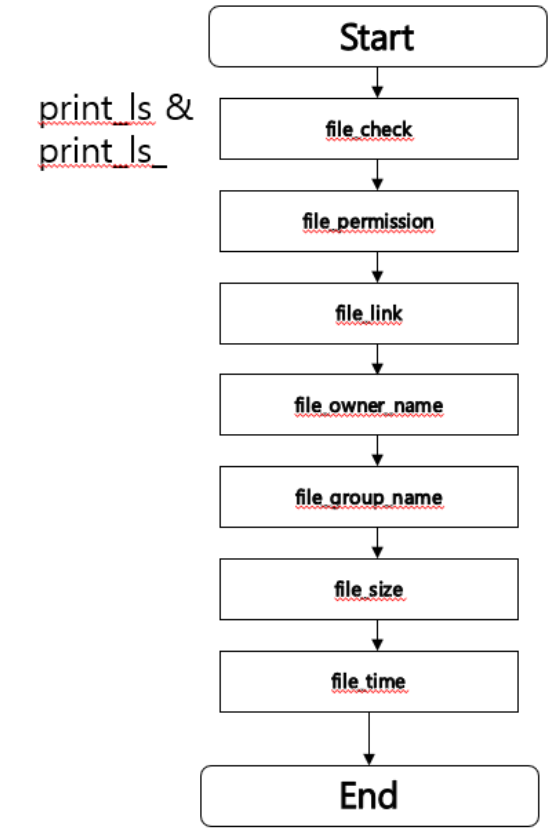
File_size



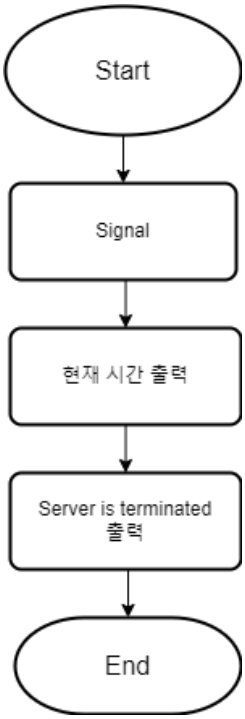
File_time



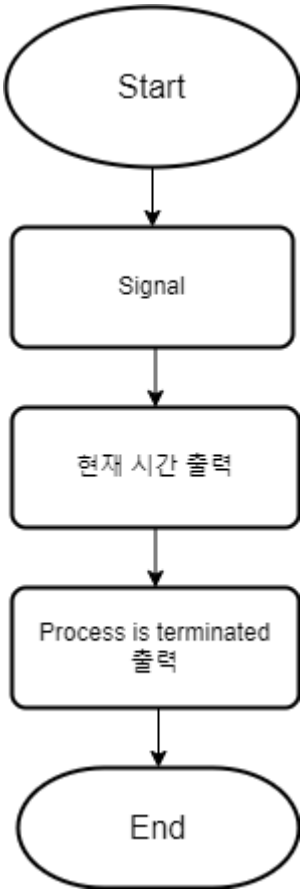
print_ls &
print_ls_



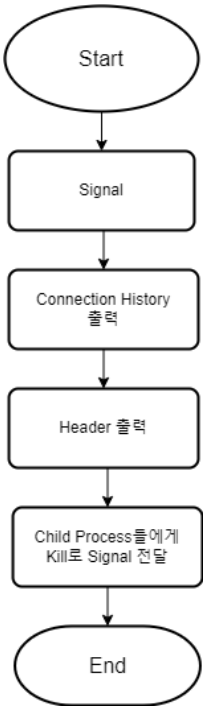
SignalHandler



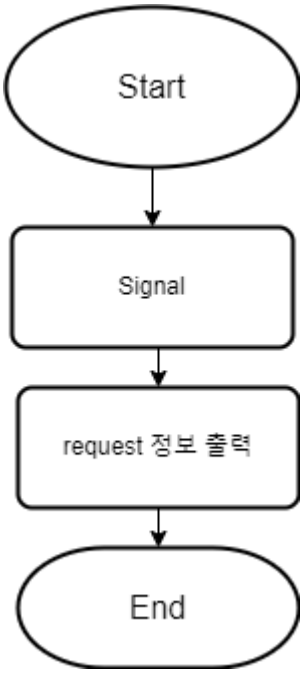
SignalHandler2



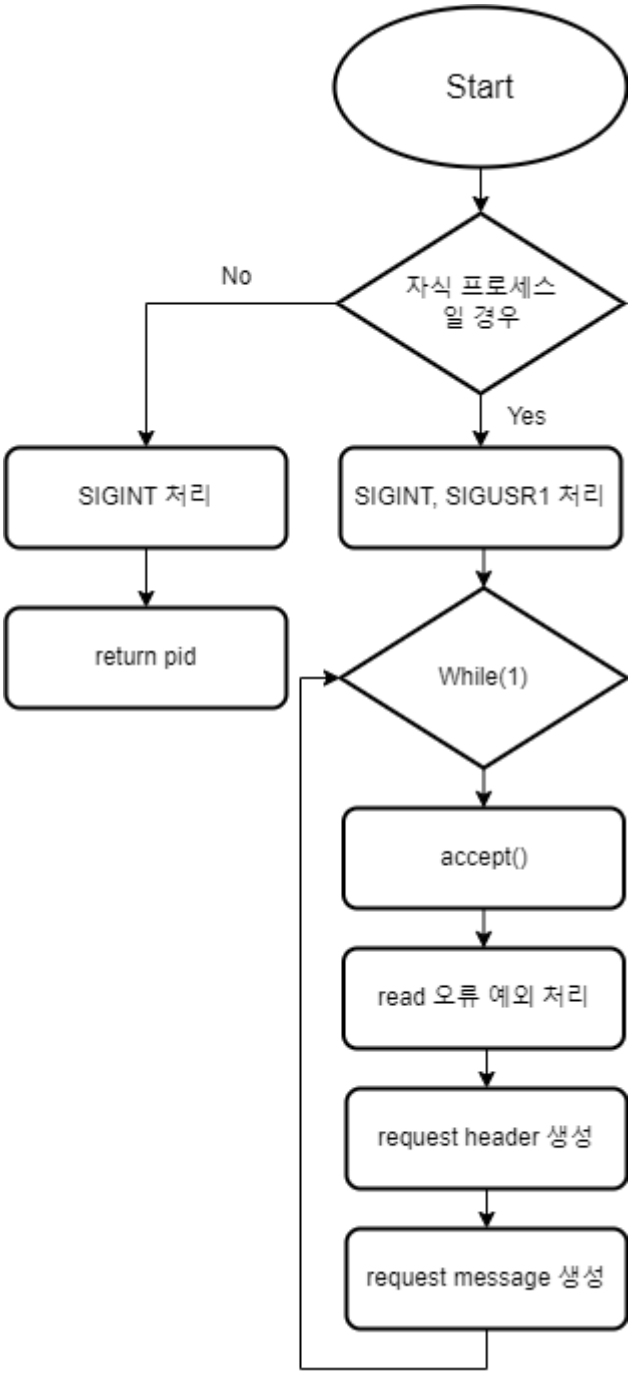
SignalHandlerChild



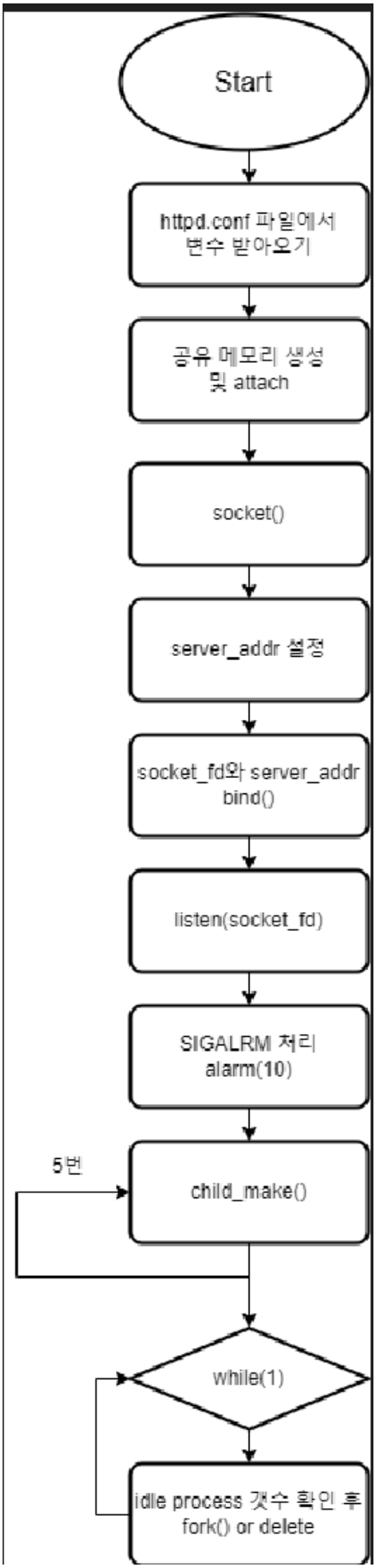
SignalHandlerParent



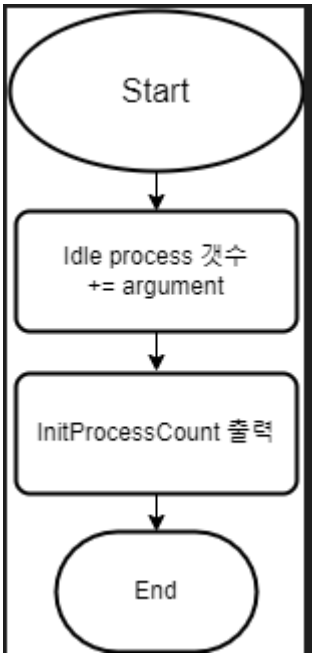
Child_make


Start
자식 프로세스
일 경우
No
Yes
SIGINT 처리
SIGINT, SIGUSR1 처리
return pid
While(1)
accept()
read 오류 예외 처리
request header 생성
request message 생성

Main

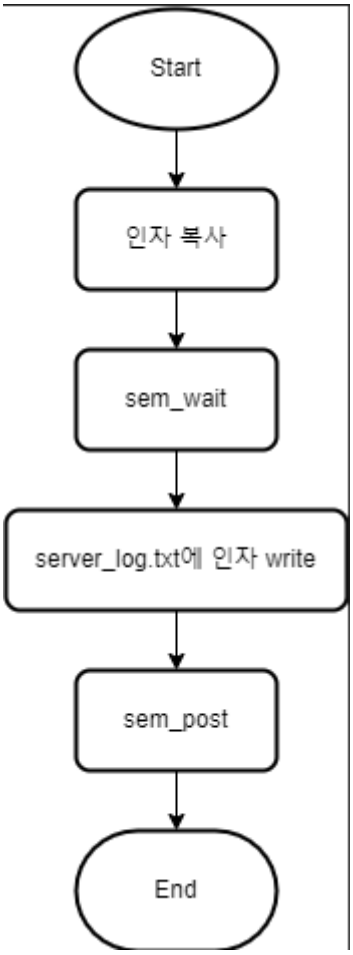

Start
httpd.conf 파일에서
변수 받아오기
공유 메모리 생성
및 attach
socket()
server_addr 설정
socket_fd와 server_addr
bind()
listen(socket_fd)
SIGALRM 처리
alarm(10)
5번
child_make()
while(1)
idle process 갯수 확인 후
fork() or delete

Threadtest



Threadtest2

Threadwrite


Start
인자 복사
sem_wait
server_log.txt에 인자 write
sem_post
End

3. Pseudo Code

```
void file_check(struct stat ){
    char 선언;
    table 행 시작;
    if (디렉토리){
        d 출력
    }
    else if (링크){
        l 출력
    }
    else if (character special){
        c 출력
    }
    else if (block special){
        b 출력
    }
    else if (socket){
        s 출력
    }
    else if (FIFO){
        p 출력
```

```
    }
    else if (regular){
        - 출력
    }
}
void file_permission(struct stat ){
    Perm[10];
    Stat 의 st_mode 를 가져와 권한에 맞게 perm 의 자리에 char 배치
    Perm 출력
}
void file_link(struct stat ){
    Stat 의 st_nlink 를 가져와 link 개수 확인
    Link 개수 출력
}
void file_owner_name(struct stat ) {
    Stat 의 getpwuid 로 st_uid 를 가져와 struct passwd 형태에 저장
    Struct passwd 의 pw_name 출력
}
void file_group_name(struct stat ){
    Stat 의 getgrgid 로 st_gid 를 가져와 struct group 형태에 저장
    Struct group 의 gr_name 출력
```

```
    }
void file_size(struct stat ){
    Off_t 변수 0 으로 초기화;
    변수에 st_size 로 대입;
    St_size 출력
}
void file_time(struct stat ){
      12 개의 month 를 배열 크기 12 개의 문자열 배열에 1 월부터 12 월까지 저장;
      Struct stat 의 st_mtime 을 localtime 함수를 이용해 struct tm 에 저장;
      Struct tm 형식의 tm_mon, tm_mday, tm_hour, tm_min 출력
      Table 행 끝
}
void print_ls(struct stat ){
      File_check, file_permission, file_link, file_owner_name, file_group_name,
      File_size, file_time 한번에 struct stat 을 넣어줘 출력
}
void SignalHandler(int Sig) {
      현재 시간 출력;
      Server is terminated 출력;
}
void SignalHandler2(int Sig) {
```

```
        현재 시간 출력;
        현재 Process is terminated 출력;
}
void SignalHandlerChild(int Sig) {
        Connection History 출력;
        출력 내용 목록 출력;
        모든 child process 에게 SIGUSR1 신호 전달;
        Alarm(10);
}
void SignalHandler(int Sig) {
        현재 시간 출력;
        Server is terminated 출력;
}
void* threadtest(void * num) {
    mutex lock;
    usleep(10000);
    인자 int 형으로 변형;
    Idle process 개수 인자만큼 더해줌;
    현재 시간 출력;
    Idle process 개수 출력;
    mutex unlock;
```

```
}
void* threadtest2(void * presenta) {
    mutex lock;
    usleep(10000);
    현재 시간 출력;
    History 공유 메모리 struct 배열에 저장;
    History 개수 증가;
    mutex unlock;
}
Pid_t child_make(int I, int socket_fd){
      Fork();
      If (부모 프로세스일 경우) {
            Signal(SIGINT, SignalHandler);
            Return pid;
      }
      Signal(SIGINT, SignalHandler2);
      Signal(SIGUSR1, SignalHandlerParent);
      Threadtest(1) 실행;
      While (1) {
      Client_fd = Accept(socket_fd);
      read(client_fd 를 읽을 경우 잘못 읽었을 때){
```

while 문 continue;

}

Threadtest(-1) 실행;

Favicon 입력 예외 처리;

Client connect 시 Time, Url, Ip, Port, Pid 터미널에 출력;

위의 정보 로그 파일에 작성;

if (자식 프로세스일 경우){

Client_fd 를 read 해서 request message 출력;

Request message 에서 url 확인;

url 이 directory 일 경우

header_tag = text/html;

url 이 이미지파일 일 경우

header_tag = image/*

그 외일 경우

Header_tag = text/plain

Threadtest2(accept history) 실행;

if (url 이 "/" (cwd)일 경우){

cwd 의 non-hidden 파일들 배열에 저장;

파일 배열 정렬;

파일배열 출력 형식을 HTML_ls 형식에 맞추어 response_message 에 저장;

}

else if (url 이 cwd 가 아닌 경우) {

if (url 이 directory 인 경우){

url 에 위치한 파일들 전부 배열에 저장;

파일 배열 정렬;

파일 배열 출력 형식을 HTML_ls 형식에 맞추어 response_message 에 저장;

}

else if (url 이 파일인 경우){

url 의 파일 정보 전부 read;

read 한 데이터 전부 response_message 에 저장;

response_header client_fd 에 출력;

}

response_message client_fd 에 출력;

else (존재하지 않는 파일인 경우){

404 Error 양식에 맞추어 출력;

url 이 directory 이거나 image 파일 인 경우

response_header 출력;

이전에 저장한 response_message 출력;

}

}

}

else (부모 프로세스일 경우) {

```
            if (url 이 /favicon.ico 일 경우) (favicon 예외 처리) {
                continue;
            }
               if (IP 가 허가되지 않은 IP 일 경우) {
                continue;
            }

            if (abs_url 이 존재하지 않는 경로일 경우(404 Error)) {
                continue;
            }
            if (이외의 경우) {
                struct Point 에 현재 access 한 client 의 정보들 저장;
                이를 struct Point 배열에 저장;
                Signal Handler 로 출력;
            }
        }
        Disconnected 시 터미널에 Time, Url, IP, Port, Pid 출력;
        위의 정보 로그 파일에 작성;
        Threadtest(+1) 실행;
}
int main(){
```

```
httpd.conf 파일 읽어와서 변수에 저장;
semaphore 생성;
server_log.txt 파일 open;
공유 메모리 생성;
공유 메모리 attach;
Socket_fd 생성;
Setsockopt 로 socket_fd 설정;
Struct sockaddr_in Server_addr 초기화;
Server_addr 과 socket_fd 바인딩;
Listen(socket_fd);
Alarm(10);
Pids 에 child_make 를 통해 모든 자식 process 의 pid 저장;
Pids 출력;
While (1)
    If (idle process 개수가 Max 보다 클경우) {
        While (idle process 개수 == 5)
            Idle process 종료;
            Process 종료 메시지 출력;
            Threadwrite 로 종료 메시지 로그 파일에 작성;
    If (idle process 개수가 Min 보다 작을 경우) {
        While (idle process 개수 == 5)
```

새로운 process 생성;

Process 생성 메시지 출력;

Threadwrite 로 생성 메시지 로그 파일에 작성;

}

4. 결과화면

```
kw2018202018@ubuntu:~/ls_a$ ./test
[Wed May 31 16:33:31 2023] Server is started.
[Wed May 31 16:33:31 2023] 54153 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 1
[Wed May 31 16:33:31 2023] 54156 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 2
[Wed May 31 16:33:31 2023] 54159 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 3
[Wed May 31 16:33:31 2023] 54162 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 4
[Wed May 31 16:33:31 2023] 54165 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 5
^C[Wed May 31 16:33:33 2023] 54153 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 4
[Wed May 31 16:33:33 2023] 54156 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 3
[Wed May 31 16:33:33 2023] 54159 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 2
[Wed May 31 16:33:33 2023] 54165 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 1
[Wed May 31 16:33:33 2023] 54162 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 0
[Wed May 31 16:33:33 2023] Server is terminated.
```

```
[Wed May 31 16:33:31 2023] 54153 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 1
[Wed May 31 16:33:31 2023] 54156 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 2
[Wed May 31 16:33:31 2023] 54159 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 3
[Wed May 31 16:33:31 2023] 54162 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 4
[Wed May 31 16:33:31 2023] 54165 process is forked.
[Wed May 31 16:33:31 2023] IdleProcessCount : 5
[Wed May 31 16:33:33 2023] 54153 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 4
[Wed May 31 16:33:33 2023] 54156 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 3
[Wed May 31 16:33:33 2023] 54159 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 2
[Wed May 31 16:33:33 2023] 54165 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 1
[Wed May 31 16:33:33 2023] 54162 process is terminated.
[Wed May 31 16:33:33 2023] InitProcessCount : 0
[Wed May 31 16:33:33 2023] Server is terminated.
```

프로그램을 시작한 후 바로 종료했을 때 예시입니다. 로그 파일에 터미널과 같이 입력된 것을 확인할 수 있습니다.

```
================= New Client =================

Time : [Wed May 31 16:39:34 2023]
URL : /aub/
IP : 127.0.0.1
Port : 50381
PID : 54235
=============================================

[Wed May 31 16:39:36 2023] IdleProcessCount : 4
================= New Client =================

Time : [Wed May 31 16:39:34 2023]
URL : /aub/
IP : 127.0.0.1
Port : 50893
PID : 54238
=============================================

[Wed May 31 16:39:36 2023] IdleProcessCount : 3
[Wed May 31 16:39:36 2023] 54278 process is forked.
```

```
=============== Disconnected client ==============
Time : [Wed May 31 16:41:05 2023]
URL : /aub/
IP : 127.0.0.1
Port : 52429
PID : 54373
Connecting Time : 4011(us)
=================================================

[Wed May 31 16:41:05 2023] IdleProcessCount : 6
=============== Disconnected client ==============
Time : [Wed May 31 16:41:05 2023]
URL : /aub/
IP : 127.0.0.1
Port : 52941
PID : 54376
Connecting Time : 3473(us)
=================================================
```

Client 가 연결되고 연결이 끊기는 경우 이러한 형식으로 출력되는 것을 확인할 수 있습니다.

```
================= New Client =================
Time : [Wed May 31 16:42:08 2023]
URL : /aub/
IP : 127.0.0.1
Port : 53453
PID : 54432
=============================================
[Wed May 31 16:42:09 2023] IdleProcessCount : 4
================= New Client =================
Time : [Wed May 31 16:42:08 2023]
URL : /aub/
IP : 127.0.0.1
Port : 53965
PID : 54435
=============================================
[Wed May 31 16:42:09 2023] IdleProcessCount : 3
[Wed May 31 16:42:09 2023] 54462 process is forked.
[Wed May 31 16:42:10 2023] IdleProcessCount : 4
[Wed May 31 16:42:10 2023] 54465 process is forked.
[Wed May 31 16:42:10 2023] IdleProcessCount : 5
=============== Disconnected client ===============
Time : [Wed May 31 16:42:14 2023]
URL : /aub/
IP : 127.0.0.1
Port : 53453
PID : 54432
Connecting Time : 3331(us)
=================================================
[Wed May 31 16:42:14 2023] IdleProcessCount : 6
=============== Disconnected client ===============
Time : [Wed May 31 16:42:14 2023]
URL : /aub/
IP : 127.0.0.1
Port : 53965
PID : 54435
Connecting Time : 2759(us)
=================================================
```

위와 같이 로그 파일에도 입력되는 것을 확인할 수 있습니다.

5. 고찰

이번 과제에서는 구현해야 하는 요소가 크게 두 가지로 분류되었습니다. Client 가 Server 와 연결되어 있는 시간을 출력하는 것과 지금까지 터미널에 출력되던 정보를 로그 파일을 생성하여 똑같이 입력하는 것 두 가지를 구현하는 것이 과제였습니다. Connection Time 을 구현하기 위해서 accept 직후의 시간과 disconnect 직후의 시간을 비교하여 그 차이만큼의 시간을 Connection Time 으로 구할 수 있었습니다. 하지만 이것을 구현함에 있어서 client 의 file descriptor 에 read 하는 과정에 있어서 Connection Time 이 비정상적으로 크게 출력되는 것을 확인할 수 있었습니다. 이 문제는 이전과 다른 웹 페이지의 출력을 request 하게 될 경우 빈번하게 발생했습니다. 웹 페이지의 로딩은 바로 완료가 됐지만 Connection Time 이 4 초가 넘게 출력되는 문제가 발생하였습니다. 이에 저는 server 와 client 간의 정보의 차이가 있어 발생한다고 생각하였지만 문제를 해결하지 못했습니다. 그리고 Client 의 Connection 과 Disconnection 동안 출력되는 정보들을 로그 파일에 write 하는 과정에서 semaphore 를 정확하게 인지하지 못하고 사용하여 terminal 에는 올바른 순서대로 출력되었지만, 로그 파일에는 순서가 뒤죽박죽으로 출력되는 문제도 발생하였습니다. 이러한 문제들은 해결되었지만 마지막 모든 process 가 종료되는 시점에서 동기화가 정확하지 않은 문제가 여전히 존재하고 이를 해결하지 못하여 아쉬웠습니다.