REPORT

[시스템프로그래밍

Assignment 3-1]

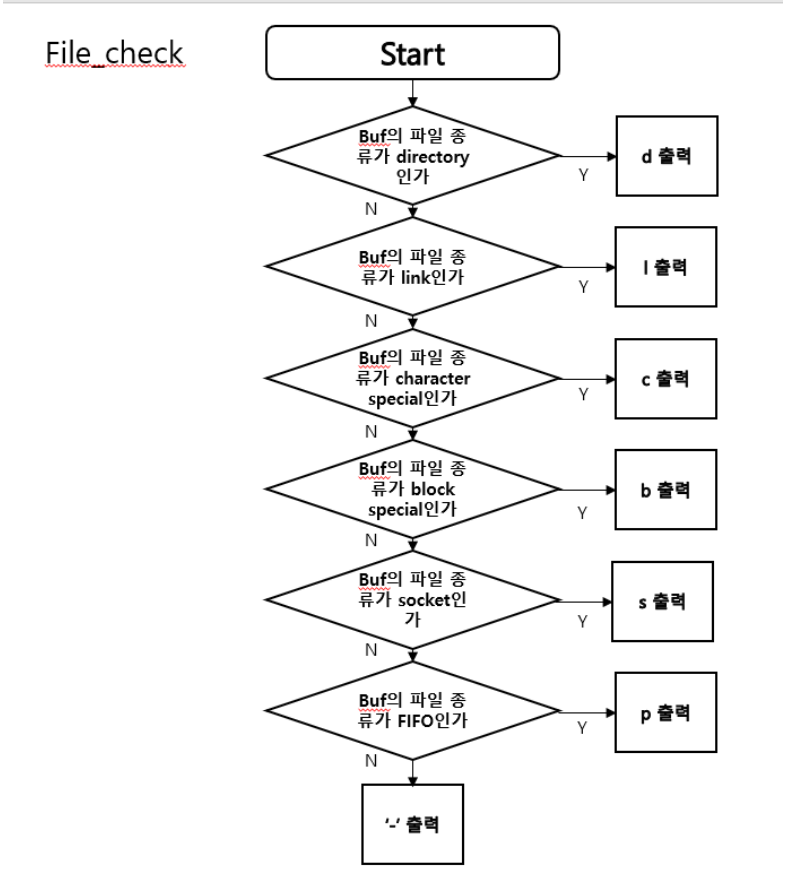
학과	컴퓨터정보공학부

학번	2018202018
이름	유승재
담당 교수	김태석 교수님
제출일	2023.05.10

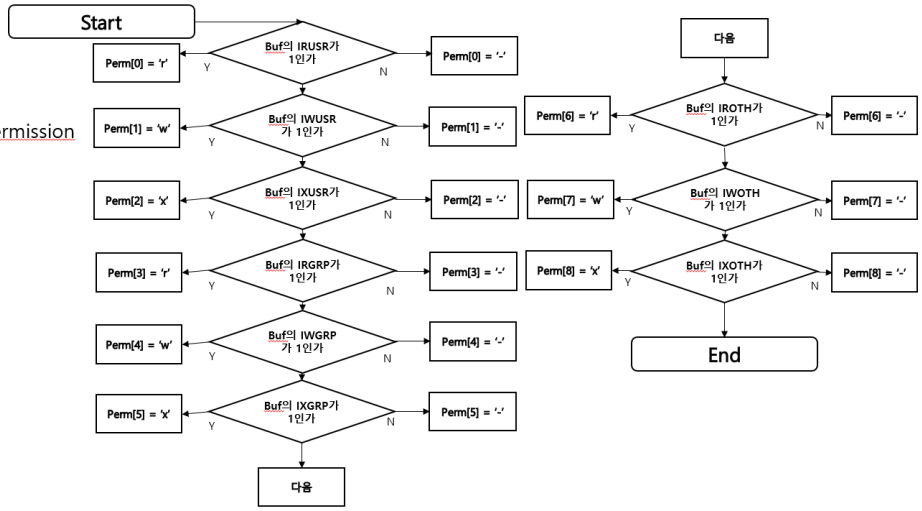
1. Introduction

이번 과제는 저번 주의 web_server 를 활용하여 여러 client 에서 다중 접속이 가능한 web_server 를 구현하는 것입니다. 이번 과제에서는 서버에 access 가능한 ip 를 설정하여 허가되지 않은 ip 에서 접속할 시에 새로운 response message 를 출력하는 기능과 fork 를 이용하여 기존의 기능을 child process 에서 구현하게 되고, parent process 에서는 access 된 기록을 전부 받아서 정해진 시간마다 출력해주도록 하는 기능 두 가지를 추가로 구현해주어야 합니다.

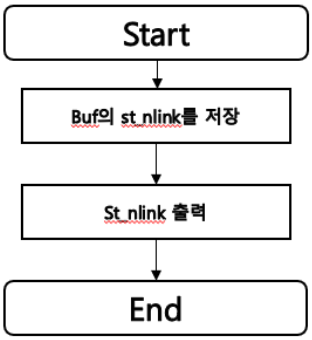
2. Flow Chart



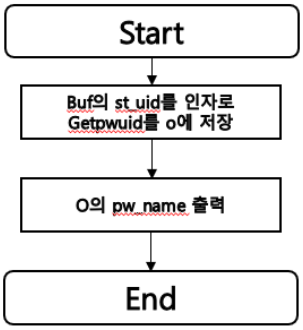
File_permission



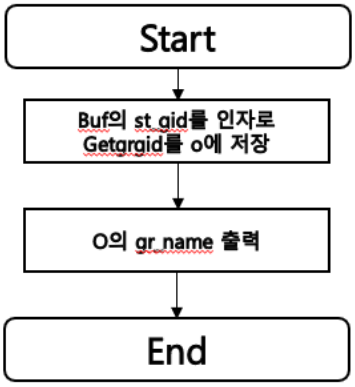
File_link



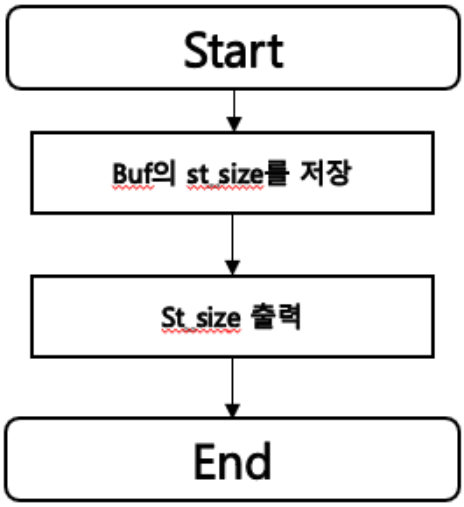
File_owner_name



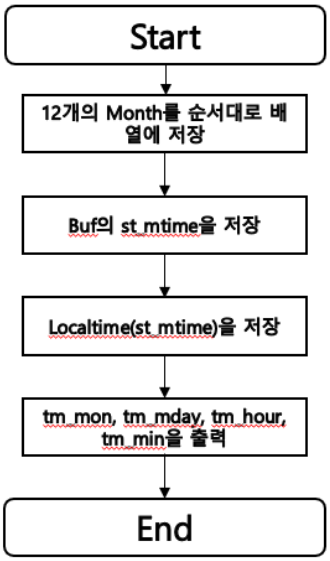
File_group_name



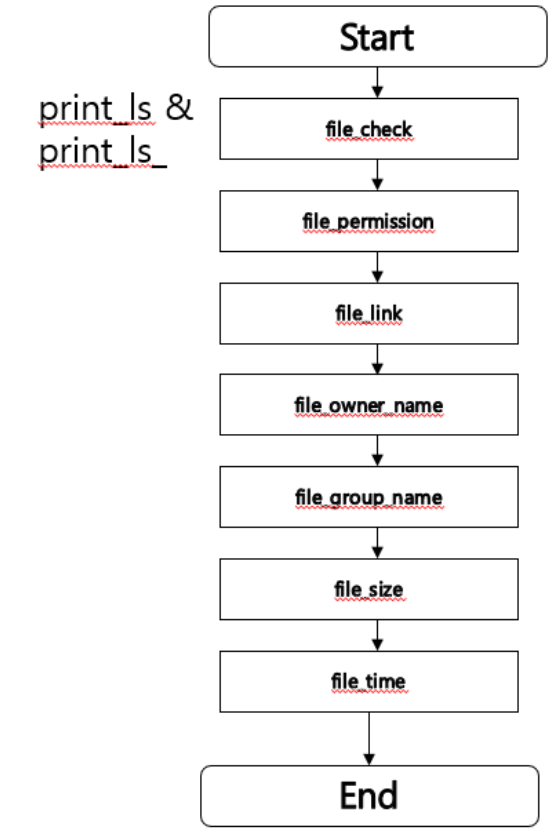
File_size



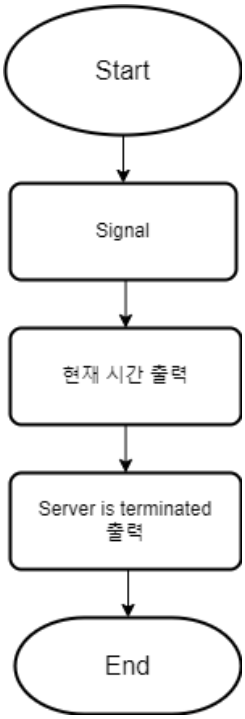
File_time



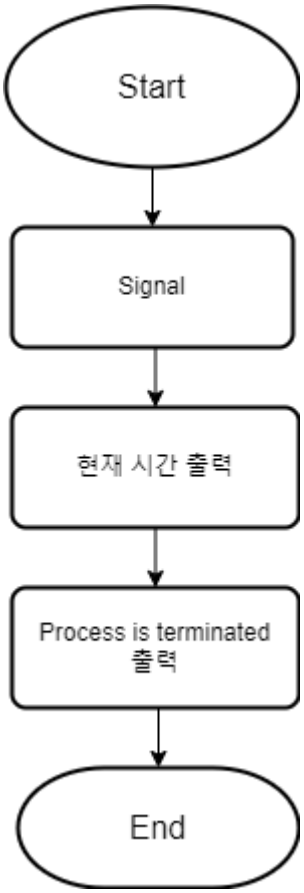
print_ls &
print_ls_



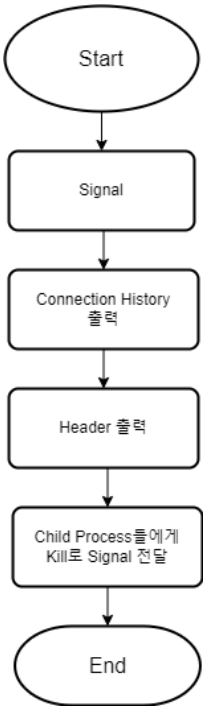
SignalHandler



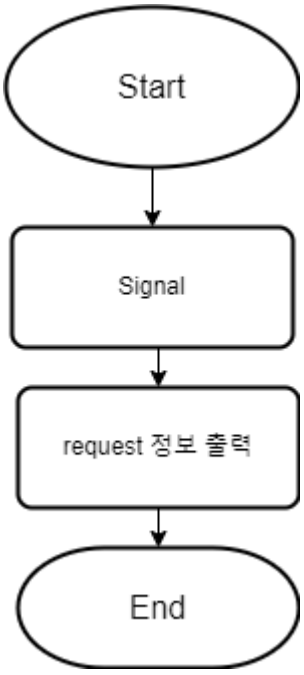
SignalHandler2



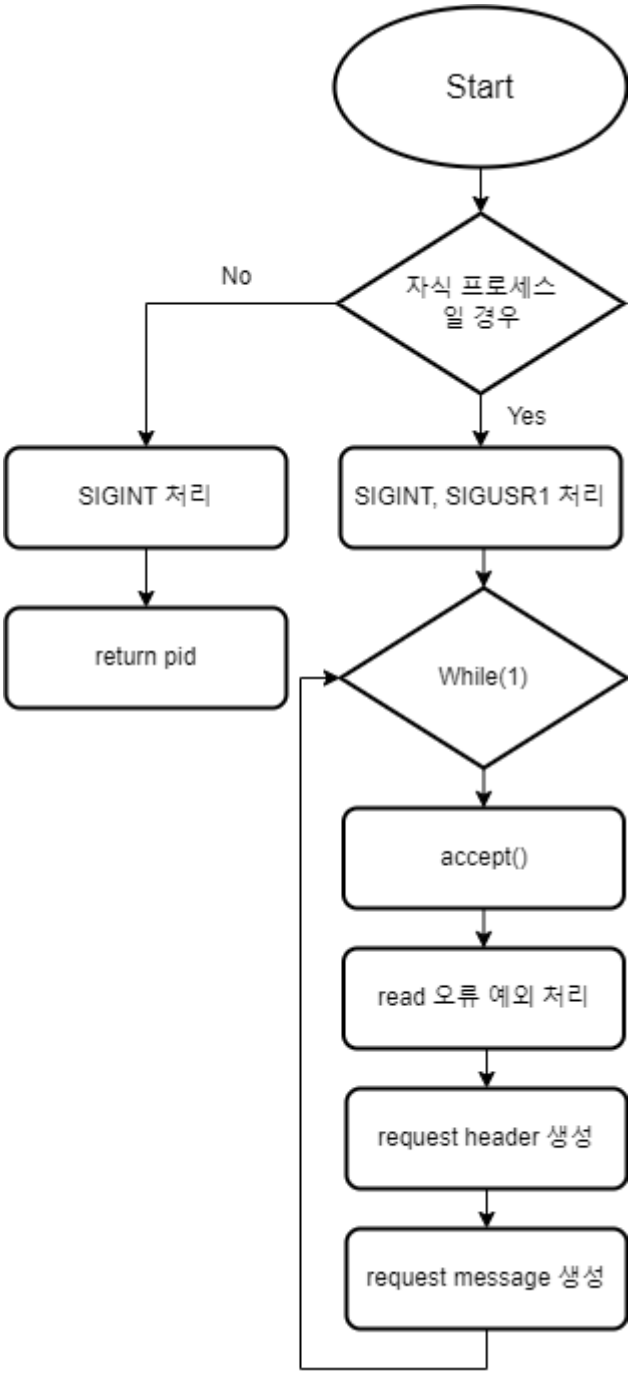
SignalHandlerChild



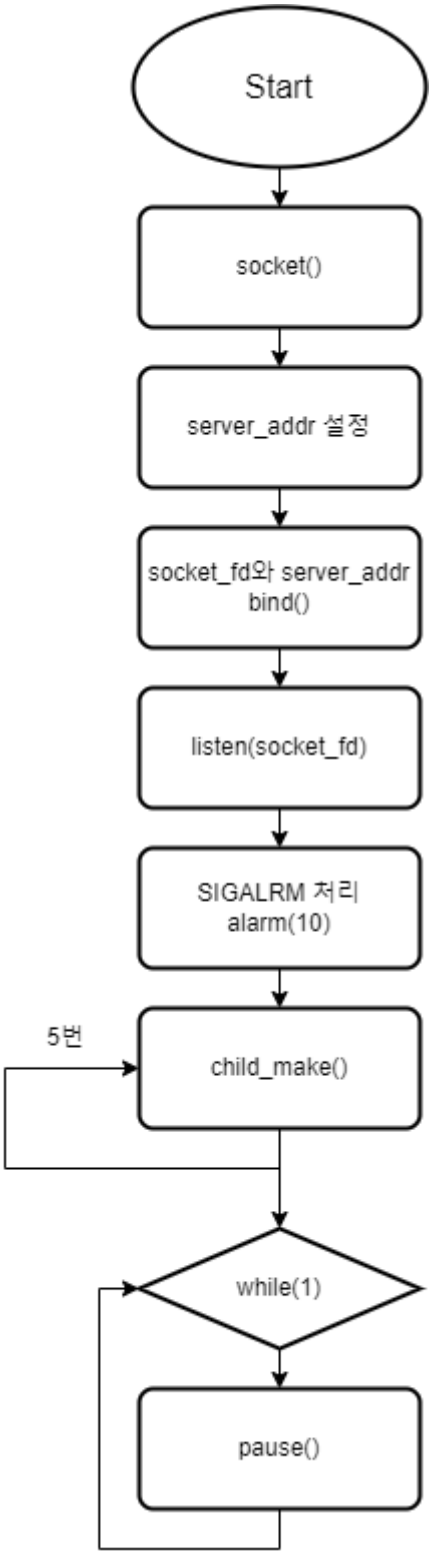
SignalHandlerParent



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
socket()
server_addr 설정
socket_fd와 server_addr
bind()
listen(socket_fd)
SIGALRM 처리
alarm(10)
5번
child_make()
while(1)
pause()

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;
}
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);
```

```
While (1) {
Client_fd = Accept(socket_fd);
read(client_fd 를 읽을 경우 잘못 읽었을 때){
    while 문 continue;
}
    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

        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;
            }
        }
}
int main(){
    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;

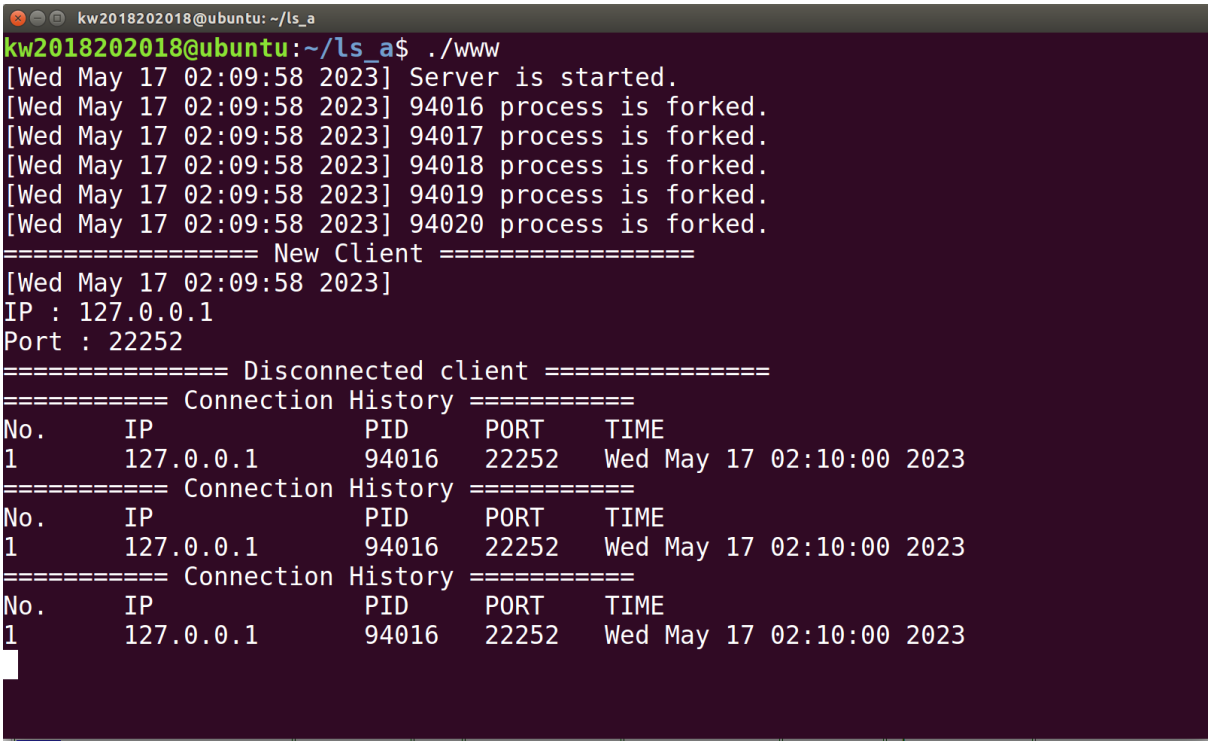
Pids 출력;

While (1)

신호 대기;

}

4. 결과화면



```
================= New Client =================
[Wed May 17 02:09:58 2023]
IP : 127.0.0.1
Port : 23788
=============== Disconnected client ===============
=========== Connection History ===========
No.     IP              PID     PORT    TIME
1       127.0.0.1       94016   22252   Wed May 17 02:10:00 2023
1       127.0.0.1       94020   23788   Wed May 17 02:10:24 2023
=========== Connection History ===========
No.     IP              PID     PORT    TIME
1       127.0.0.1       94016   22252   Wed May 17 02:10:00 2023
1       127.0.0.1       94020   23788   Wed May 17 02:10:24 2023
```

```
================= New Client =================
[Wed May 17 02:09:58 2023]
IP : 127.0.0.1
Port : 24300
=============== Disconnected client ===============
=========== Connection History ===========
No.     IP              PID     PORT    TIME
1       127.0.0.1       94018   24300   Wed May 17 02:10:47 2023
1       127.0.0.1       94016   22252   Wed May 17 02:10:00 2023
1       127.0.0.1       94020   23788   Wed May 17 02:10:24 2023
```

```
=============== Disconnected client ===============
=========== Connection History ===========
No.     IP              PID     PORT    TIME
1       127.0.0.1       95293   37612   Wed May 17 02:20:48 2023
1       127.0.0.1       95297   37100   Wed May 17 02:20:47 2023
=========== Connection History ===========
No.     IP              PID     PORT    TIME
1       127.0.0.1       95293   37612   Wed May 17 02:20:48 2023
1       127.0.0.1       95297   37100   Wed May 17 02:20:47 2023
^C[Wed May 17 02:20:54 2023] 95293 process is terminated.
[Wed May 17 02:20:54 2023] 95294 process is terminated.
[Wed May 17 02:20:54 2023] 95295 process is terminated.
[Wed May 17 02:20:54 2023] 95296 process is terminated.
[Wed May 17 02:20:54 2023] 95297 process is terminated.
[Wed May 17 02:20:55 2023] Server is terminated.
kw2018202018@ubuntu:~/ls_a$
```

client 에서 request 를 전달할 때 마다 계속 새로운 자식 process 에서 출력되는 것을 확인할 수 있습니다. Ctrl + C 로 SIGINT 를 전달해줄 경우 자식 process 가 전부 종료된 후 parent process 가 종료되는 것을 확인할 수 있습니다.

5. 고찰

이번 과제에서는 이전 과제에서 까지는 client 의 request 를 받은 후에 fork 를 진행하는 식에서 미리 fork 를 실행하여 자식 process 에서 그 후에 들어오는

request 를 처리하는 방식으로 코드를 변경하는 것이 중점이었습니다. 처음에는 pre-fork 형식의 process 처리 방식이 이해가 쉽지 않았으나 미리 자식 프로세스를 만들어 연결해주는 과정을 이해하니 어렵지 않게 만들었습니다. 다만 이번 과제에서는 Signal 을 처리하는 과정이 중요한 요소로 작용했다고 생각하는데 마지막 Ctrl + C 를 통해 SIGINT 를 전달하여 이를 처리하는 과정에서 자식 process 가 모두 종료된 후 부모 프로세스가 이를 인지하고 종료되는 과정으로 코드를 구현하는 것이 맞다고 생각했으나 이를 완벽하게 구현하기 어렵다고 판단하여 sleep 을 통하여 자식 프로세스와 부모 프로세스가 동시에 종료되는 과정에서 부모를 강제로 자식 프로세스보다 이후에 종료되도록 설정하여 이 문제를 일시적으로 해결했습니다. 이 과정을 해결한다면 더 좋은 코드가 될 것이라고 생각합니다.

6. Reference