시스템프로그래밍 보고서

실험제목: assignment3-1 과제

제출일자: 2023년 05월 13일 (토)

학 과: 컴퓨터공학과

담당교수: 이기훈 교수님

실습분반: 금요일 5 6

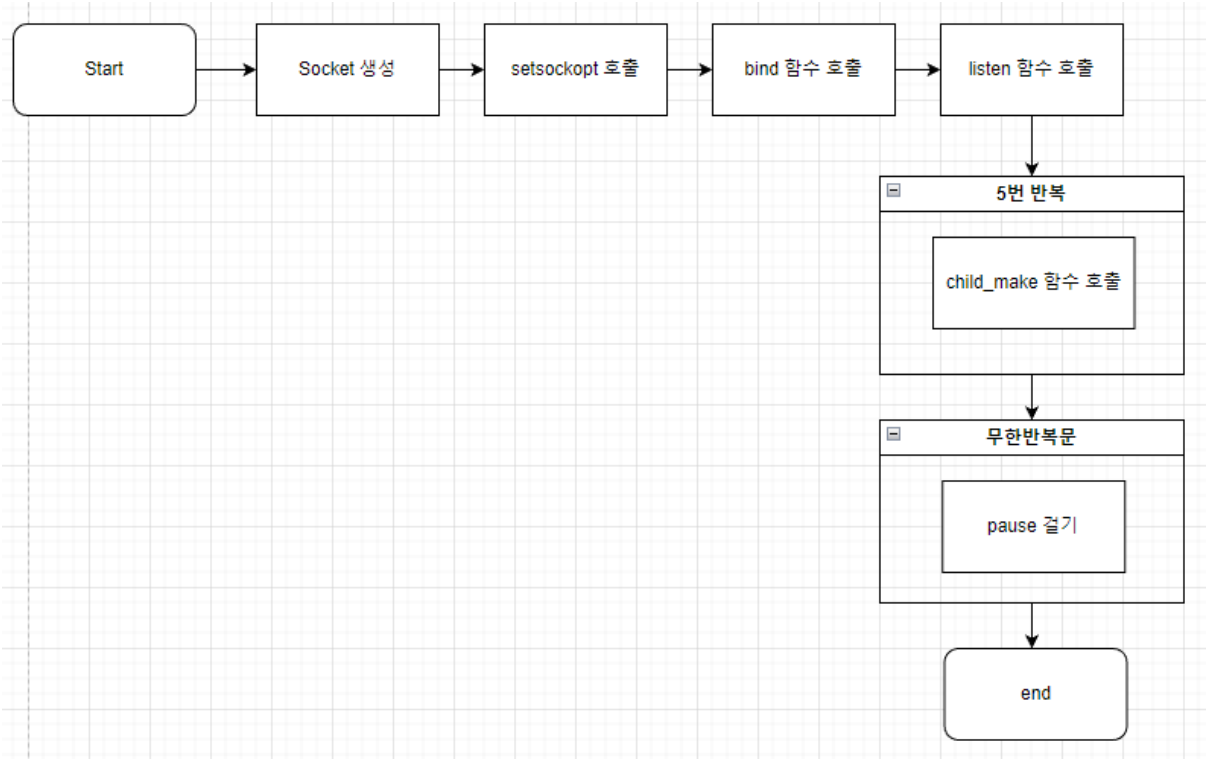
학 번: 2019202031

성 명: 장형범

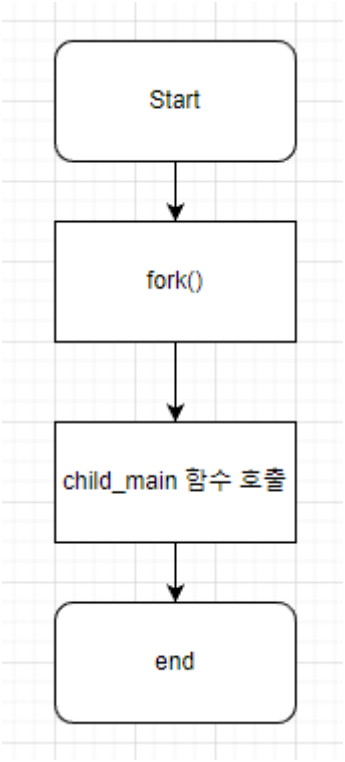
1. Introduction

2-3 에서 구현한 다중 접속을 5 개의 child 프로세스를 만들고, terminal 에 그 정도를 출력한다. SIGINT 발생 시, 모든 프로세스를 완전하게 종료 시켜야 한다. 단 좀비 프로세스가 생성되지 않도록 보장해야 한다. 10 초마다 자식 프로세스에 연결된 client 의 접속 기록을 출력해야 한다. 각 자식 프로세스마다 최신 history 기록을 저장 및 출력한다.

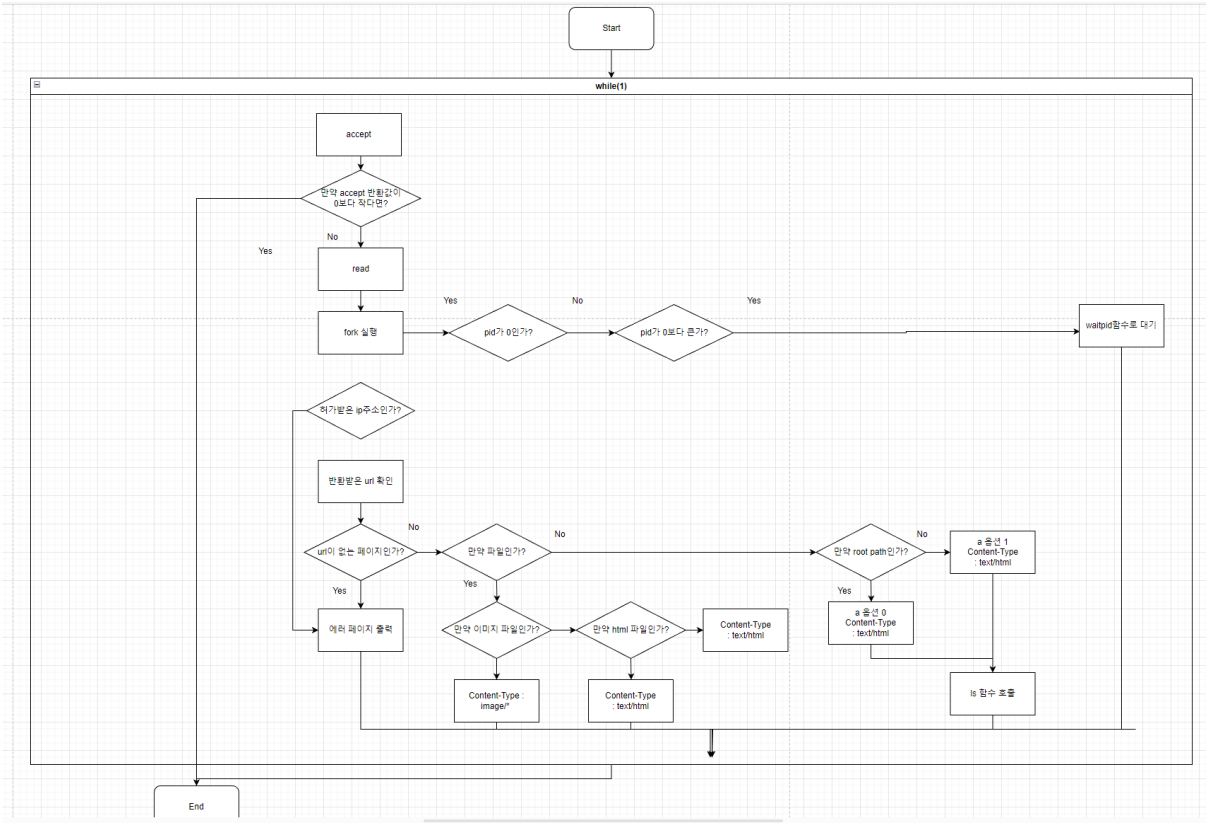
2. Flow chart



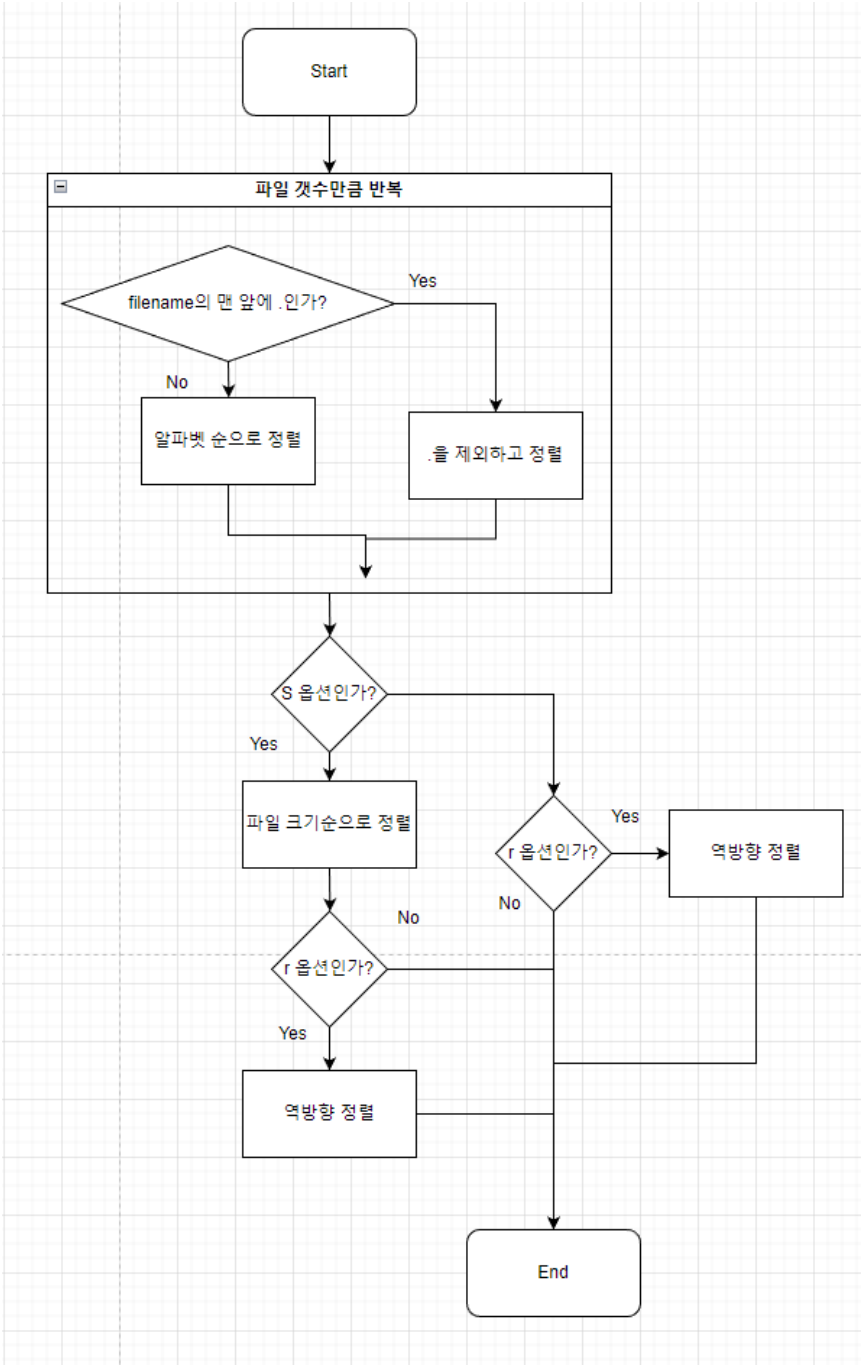
main 함수의 flow chart 이다.



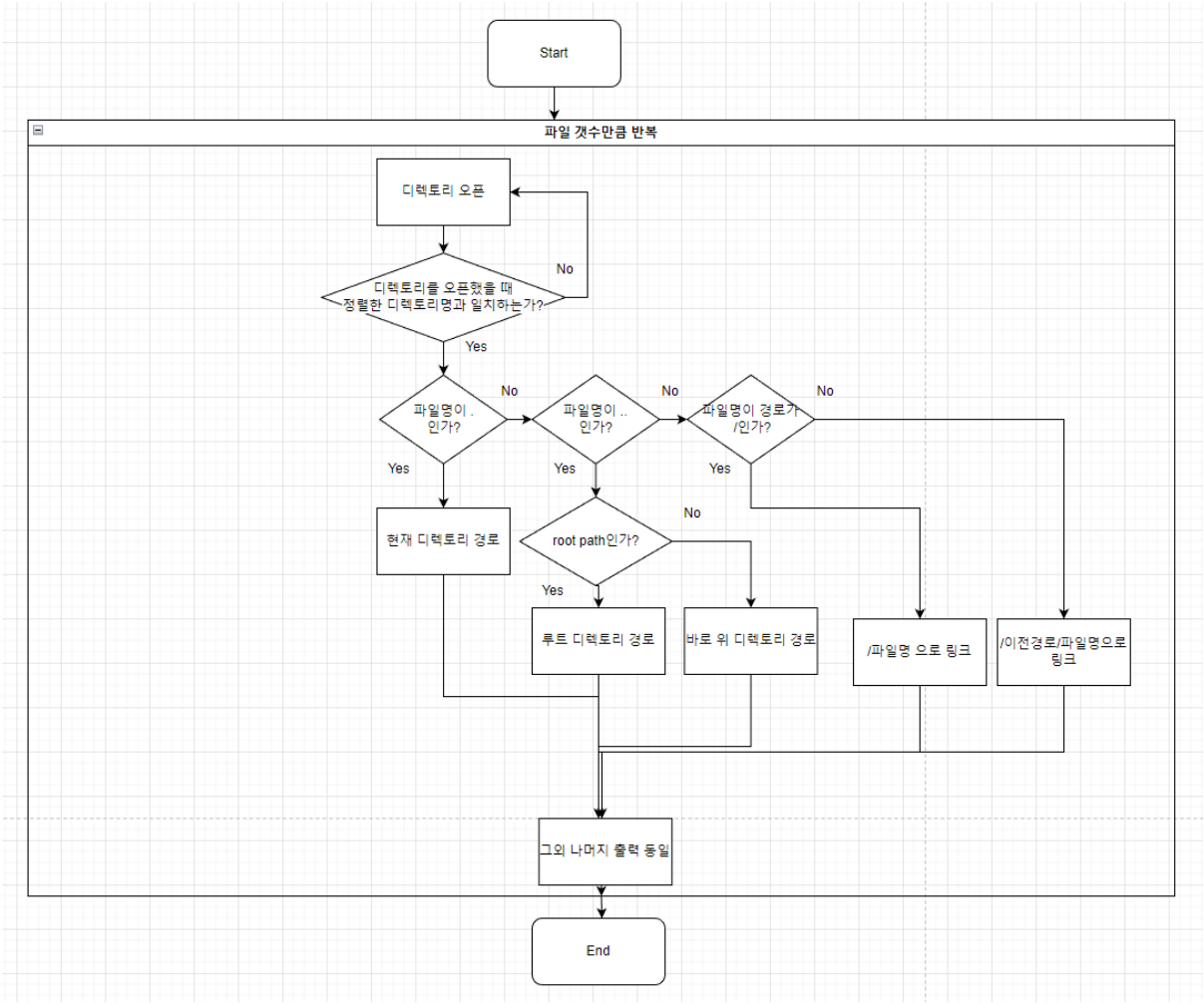
child_make 함수이다.



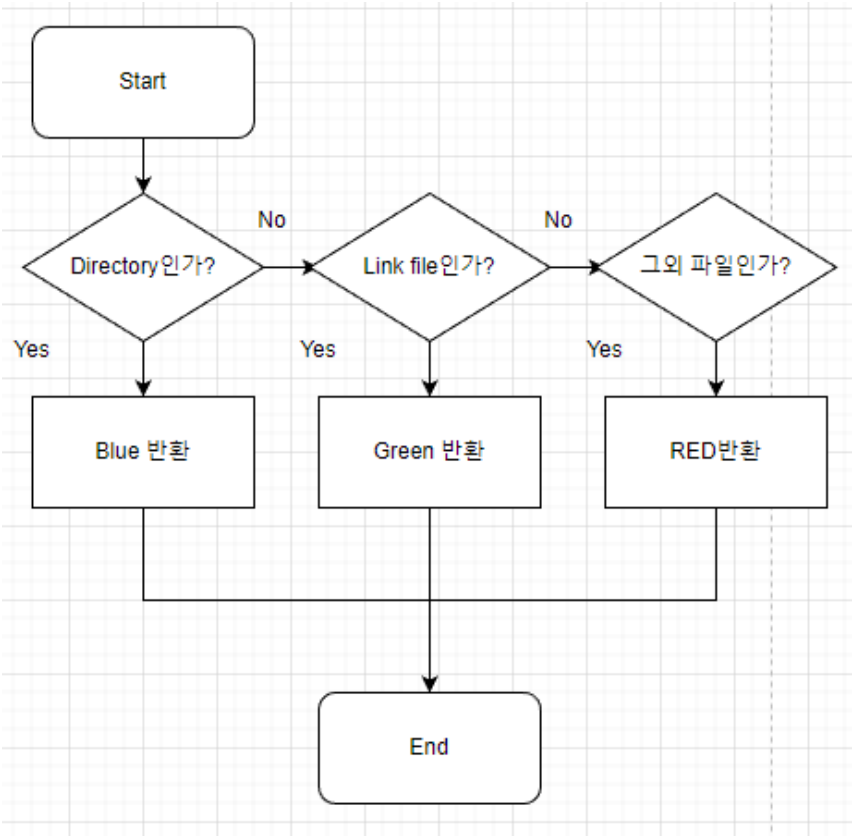
child_main 함수이다



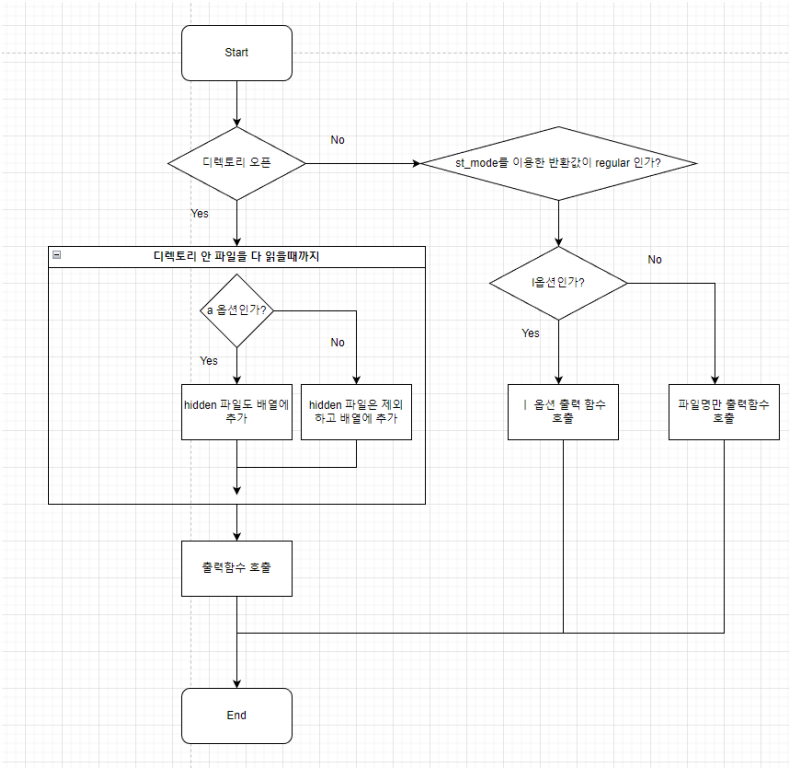
sorting 함수의 순서도이다.



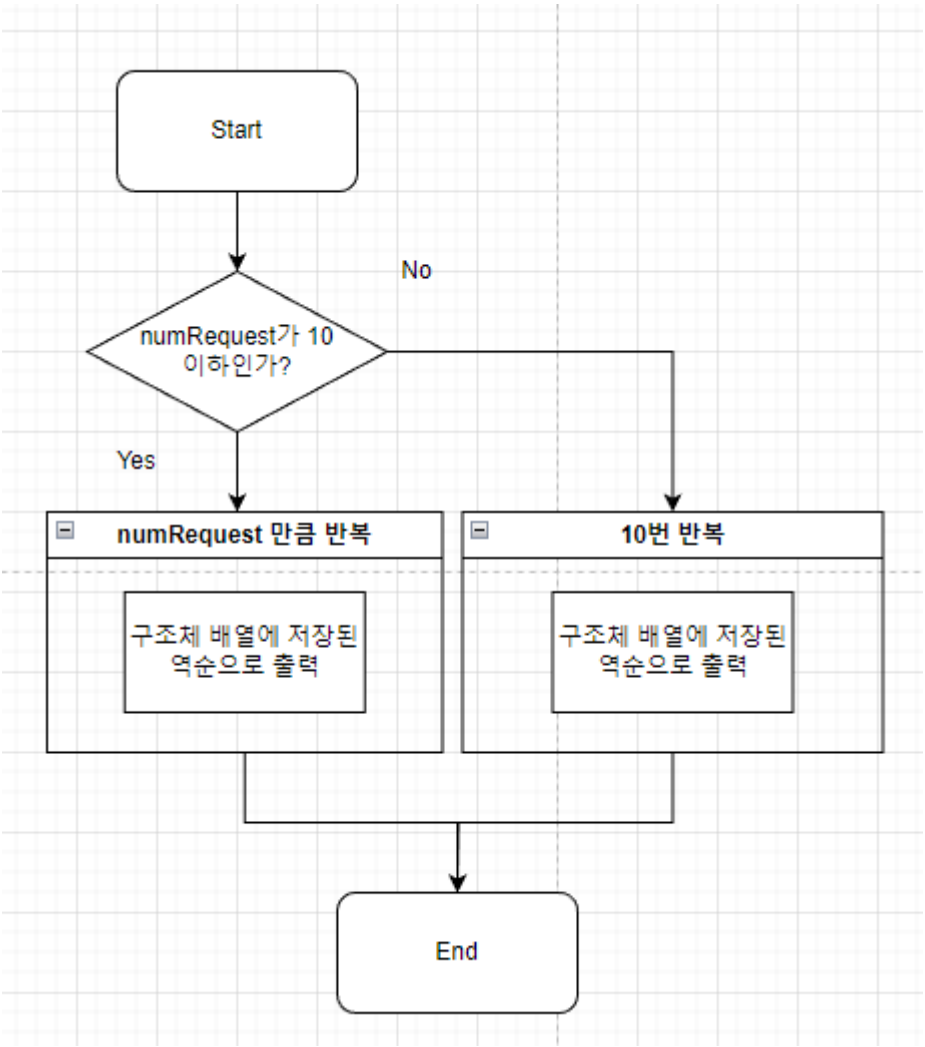
print_file_info 함수의 순서도이다.



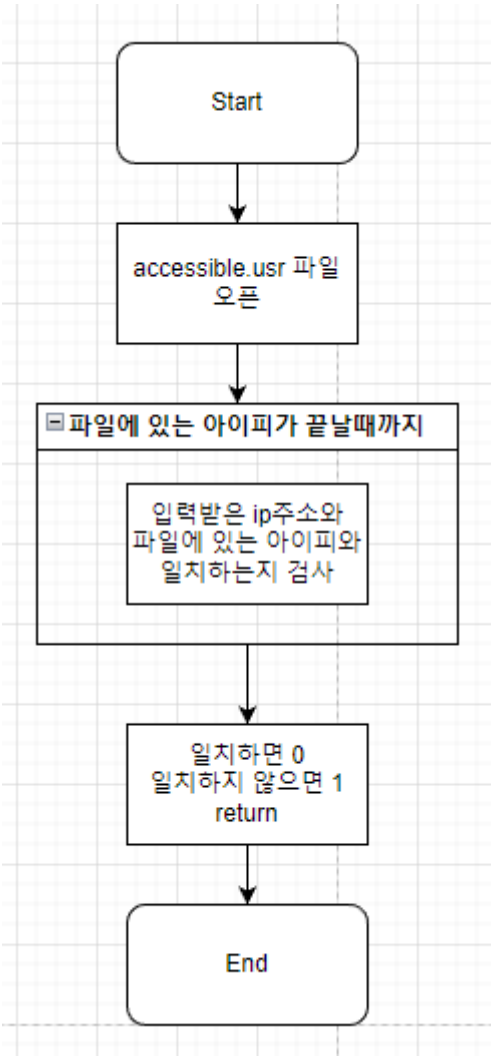
print_filetype 함수의 순서도이다.


Start
디렉토리 오픈
No
st_mode를 이용한 반환값이 regular 인가?
Yes
디렉토리 안 파일을 다 읽을때까지
a 옵션인가?
Yes
No
hidden 파일도 배열에 추가
hidden 파일은 제외하고 배열에 추가
l옵션인가?
No
Yes
l 옵션 출력 함수 호출
파일명만 출력함수 호출
출력함수 호출
End

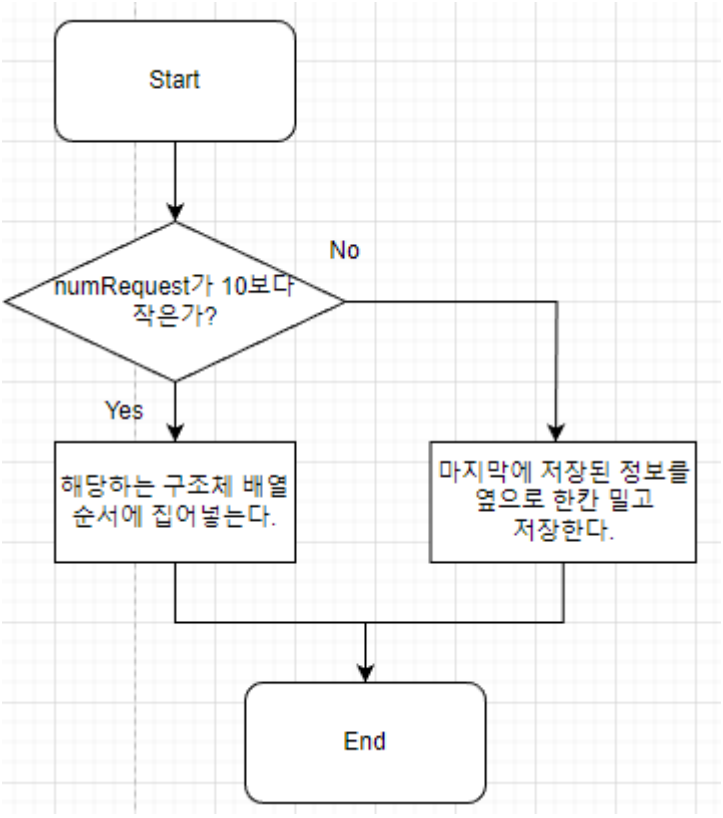
ls 함수의 순서도이다.



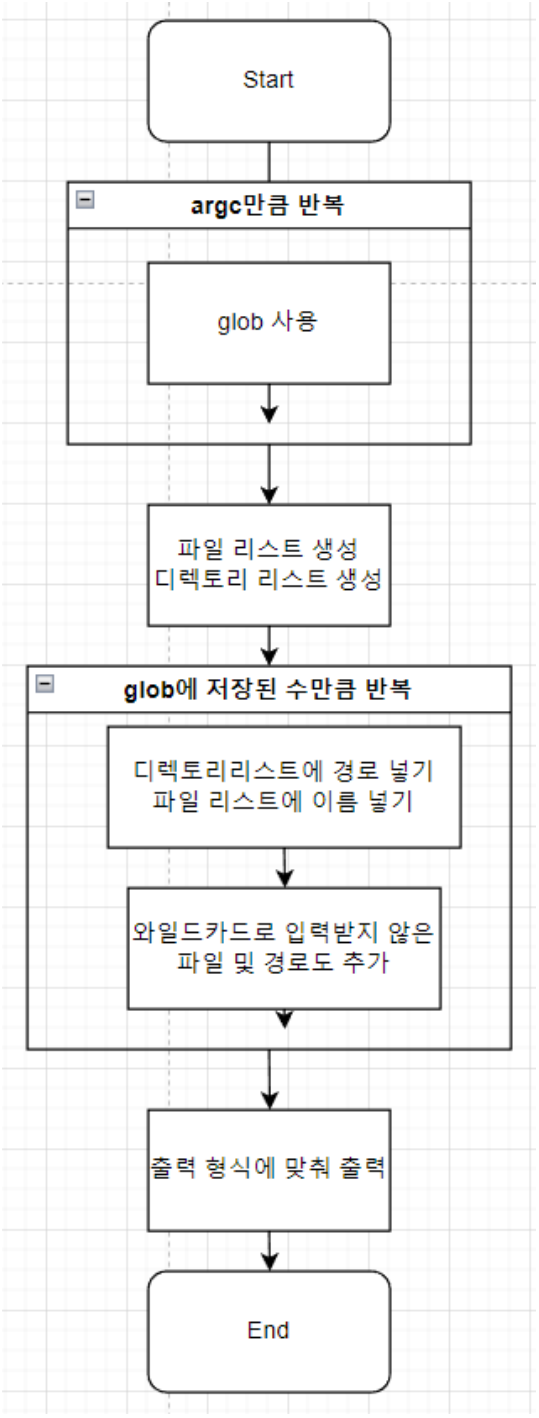
printHistory 함수이다.



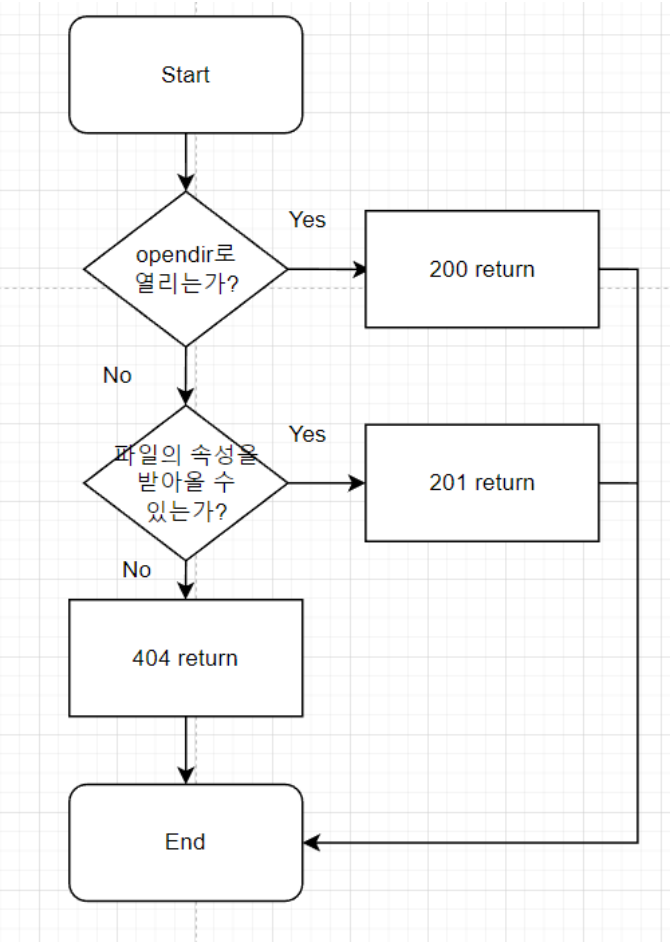
is_ip_allowed 함수이다.



addHistory 함수이다.



wild card 의 순서도이다.



check_404 의 순서도이다.

3. sudo code

```
void alarmHandler()
{
    if (SIGINT 시그널이 들어온다면) // if parent process and signal is SIGINT
    {
        for (5 번 반복)
        {
            kill(자식 프로세스에게 명령 전달);
        }
            부모 프로세스 종료// parent
    }
    else if (SIGCHILD 시그널이 들어온다면)
    {
        자식 프로세스 회수
    }
```

```
    else if (SIGALARM 이라면) // if parent process alarm
    {
        for (5 번 반복)
        {
            kill(자식 프로세스에게 명령어 전달);
        }
    }
    alarm(10); // set alarm time
}
void child_alarmHandler()
{
    if(SIGTERM) //child process exit
    {
            exit(0);
    }
    else if(signum == SIGUSR1)
        printHistory(); //print history
}

int main(int argc, char** argv)
{
       소켓 함수 호출
       setsockopt 로 셋팅
       bind 함수로 소켓과 구조체 정보 binding
       listen 함수로 client 로부터 connection 을 위해 대기상태로 변경
       for(5 번 반복)
       {
              child_make 함수 호출
       }
       for(무한반복)
       {
              pause 걸기
       }
    return 0;
}
pid_t child_make(int i, int socketfd, int addrlen)
{

    if((pid = fork()) > 0)
    return 부모프로세스 돌아가기;

    child_main 함수 호출
}

void child_main()
{
    while(1)
    {
```

```
        자식 프로세스
              {
              10 초마다 출력되는 알람 설정
              accept 로 서버가 클라이언트 접속 허용
              read 로 정보 받음
               fork 함수 사용

              if 허가된 ip list 에 없으면 access denied 출력
              if(없는 페이지)
                     404 에러 출력
              이미지, text, 일반 파일이면 조건에 맞게 출력
              root path 라면 a 옵션 off
              root path 가 아니라면 a 옵션 on
              }
              부모 프로세스
              {
              구조체 배열에 정보 전달
              자식이 종료되기까지 대기 후 closed
              }

    }
}

void alarmHandler(int signum)
{
    if (SIGCHILD 가 신호로오면)
    {
       waitpid 함수를 이용한 자식 프로세스 pid > 0 일때까지 반복
    }
    else if (signal == SIGALRM)
    {
        구조체 배열 선언
    }
    alarm(10); // set 10 second
}

void printHistory()
{
    if (저장된 history 가 10 개 이하라면) { //process number < 10
        저장된 것 만큼 반복해서 역순으로 출력
    }
    else
    {
        for (10 번 반복)
        {
            역순으로 출력
    }
}
```

```
int is_ip_allowed() {
    accessible.usr 파일 오픈
    fnmatch 함수를 이용해 파일에 있는 ip 주소들 검사
    같은게 있다면 거짓 반환 (함수 호출을 위해)
    같은게 없다면 참 반환
}
    char line[256];
    while (fgets(line, sizeof(line), file) != NULL) { //get ip address
        line[strcspn(line, "\n")] = '\0'; // cut \n word
        if (fnmatch(line, client_ip, FNM_CASEFOLD) == 0) { //compare word
            fclose(file);
            return 0;  //return allow ip
        }
    }
    fclose(file);
    return 1;  // return not allow ip
}



void addHistory() {
    if (저장된 개수가 10 개보다 적다면) {
        그다음 위치에 프로세스 정보 저장

    } else {
        기존에 저장된 배열을 한칸 민다.
        밀어서 생긴 공간에 프로세스 정보 저장한다.
    }
    numRequests++;
}

int check_404
{
    디렉토리라면
        return 200; //exist
    }
    파일이라면
    {
        return 201; //exist
    }
    else
        return 404; //not exist
}

void wild_card
{
    argc 만큼 반복문을 사용해 glob 함수 사용
```

```
    파일과 디렉토리 갯수 체크
     파일과 디렉토리 배열 생성
    배열에 정보 저장
    와일드카드가 아니었던 경로나 파일을 배열에 추가
    테이블 생성
    값 write 하기
}

void no_dir
{
    테이블 생성
    디렉토리가 아닌 파일의 출력 -l 옵션일때 사용
    write 하기
}

void ls
{
   디렉토리를 열고 안에 있는 파일 read
   파일의 size 합산, 배열에 파일명 넣기
   옵션에 맞는 파일 정렬
}

void sorting
{
       알파벳 순서로 배열 정렬 만약 파일의 이름 맨 앞이 .이라면 그 다음 알파벳
       부터 비교
       S 옵션이 1 이라면 dir 를 파일의 크기로 정렬
       r 옵션이 1 이랑면 역순으로 파일 정렬

}

char* check_filetype
{
    type 반환
}

void print_file_info
{
       테이블 생성
       l 옵션이라면 파일의 세부 정보 html 테이블에 입력

       옵션이 없다면 파일의 이름만 테이블에 입력
    }

}
```

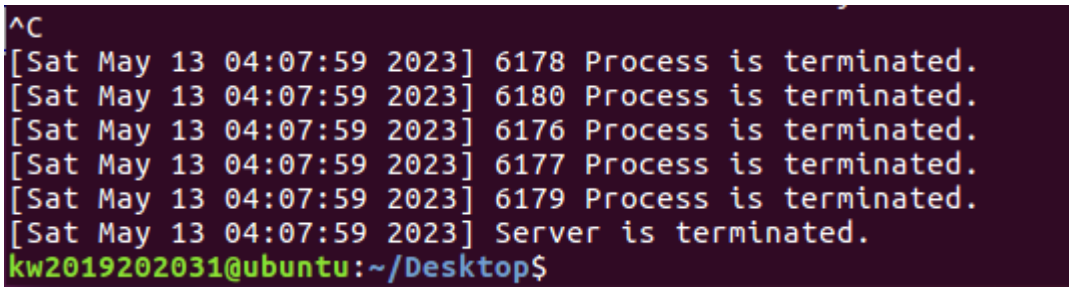
결과화면

```
kw2019202031@ubuntu:~/Desktop$ ./html_ls
[Sat May 13 04:03:29 2023] Server is started.
[Sat May 13 04:03:29 2023] 6150 Process is forked.
[Sat May 13 04:03:29 2023] 6151 Process is forked.
[Sat May 13 04:03:29 2023] 6152 Process is forked.
[Sat May 13 04:03:29 2023] 6153 Process is forked.
[Sat May 13 04:03:29 2023] 6154 Process is forked.
=============== New Client ===============

[Sat May 13 04:03:31 2023]
IP : 192.168.111.138
Port : 1700
=========================================
========== Disconnected Client ===========

[Sun Apr 30 06:42:31 2023]
IP : 192.168.111.138
Port : 1700
=========================================
=============== New Client ===============

[Sat May 13 04:03:31 2023]
IP : 192.168.111.138
Port : 2212
=========================================
========== Disconnected Client ===========

[Sun Apr 30 06:42:31 2023]
IP : 192.168.111.138
Port : 2212
=========================================
=============== New Client ===============

[Sat May 13 04:03:32 2023]
IP : 127.0.0.1
Port : 39146
=========================================
========== Disconnected Client ===========

[Sat May 13 04:03:32 2023]
IP : 127.0.0.1
Port : 39146
=========================================
=============== New Client ===============

[Sat May 13 04:03:32 2023]
IP : 127.0.0.1
Port : 39658
=========================================
========== Disconnected Client ===========

[Sat May 13 04:03:32 2023]
IP : 127.0.0.1
Port : 39658
=========================================
```

서버를 켜면 Server is started 가 나오며 자식 process 의 pid 번호를 보여준다. 이후 연결된 자식 프로세스들을 보여준다.

```
============ Connection History ============================================
No.     IP              PID     PORT    TIME
1       192.168.111.138 6151    2212    Sat May 13 04:03:31 2023
1       127.0.0.1       6153    39658   Sat May 13 04:03:32 2023
1       192.168.111.138 6150    1700    Sat May 13 04:03:31 2023
1       127.0.0.1       6152    39146   Sat May 13 04:03:32 2023
```

연결되었던 기록을 보여준다.



ctrl c 를 누르면 자식 프로세스들이 먼저 종료되면서 마지막에 부모 프로세스가 종료된다.

고찰

종료시킬 때 자식 프로세스를 먼저 종료시키고 부모 클라이언트를 종료시킬 때 약간 출력이 이상해서 애를 먹었다. 출력문의 위치를 살짝 옮기니 정상적으로 작동했다. connection history 를 볼 때 PID 가 섞여서 나왔는데 조교님께 여쭤보니 클라이언트에서 주기적으로 뭔가를 보내는데 그 주기가 우연히 겹쳐 이런 경우가 나온다고 하셨다.

reference

강의자료 참고