시스템프로그래밍 보고서

실험제목: assignment3-3 과제

제출일자: 2023년 05월 27일 (토)

학 과: 컴퓨터공학과

담당교수: 이기훈 교수님

실습분반: 금요일 5 6

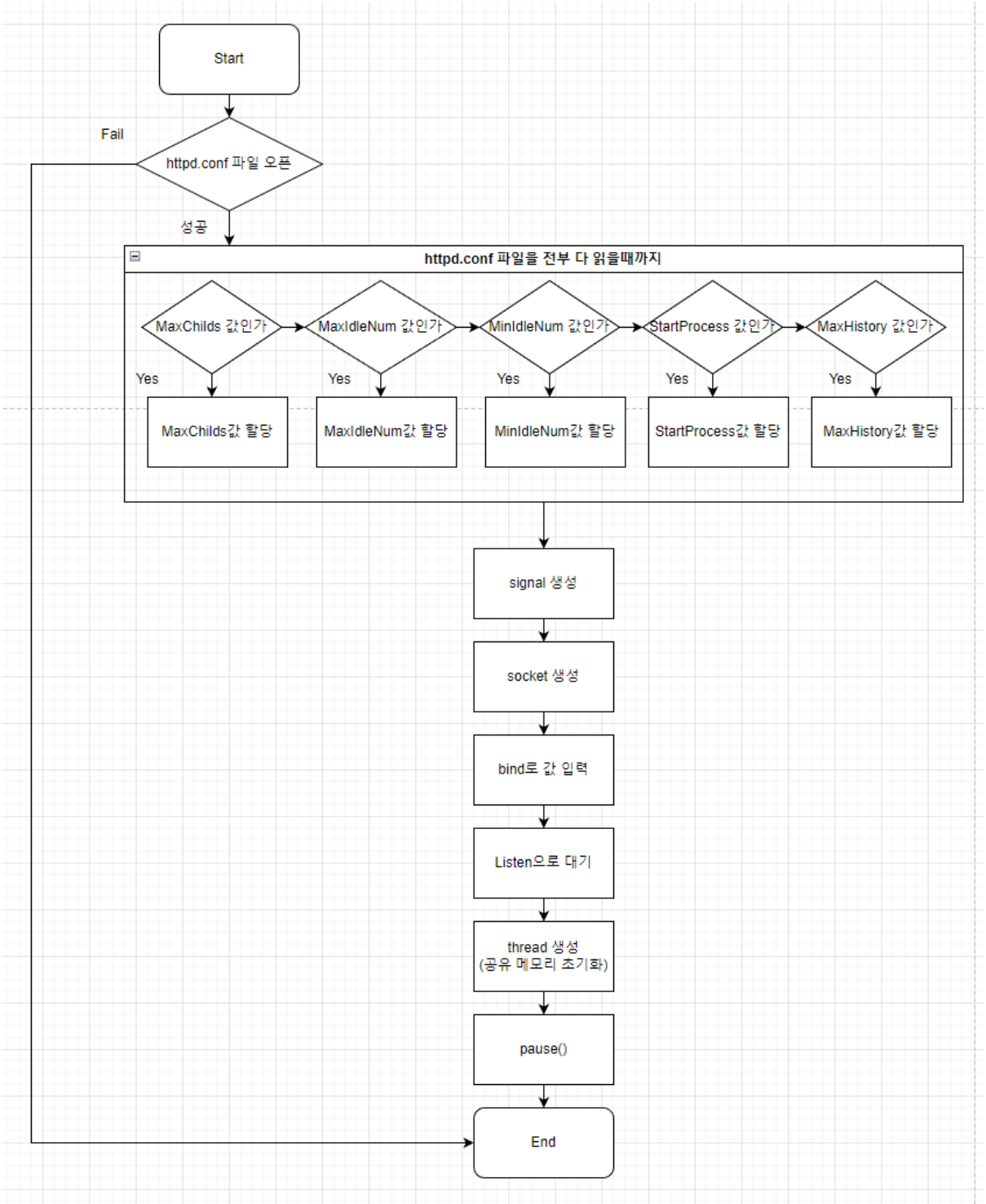
학 번: 2019202031

성 명: 장형범

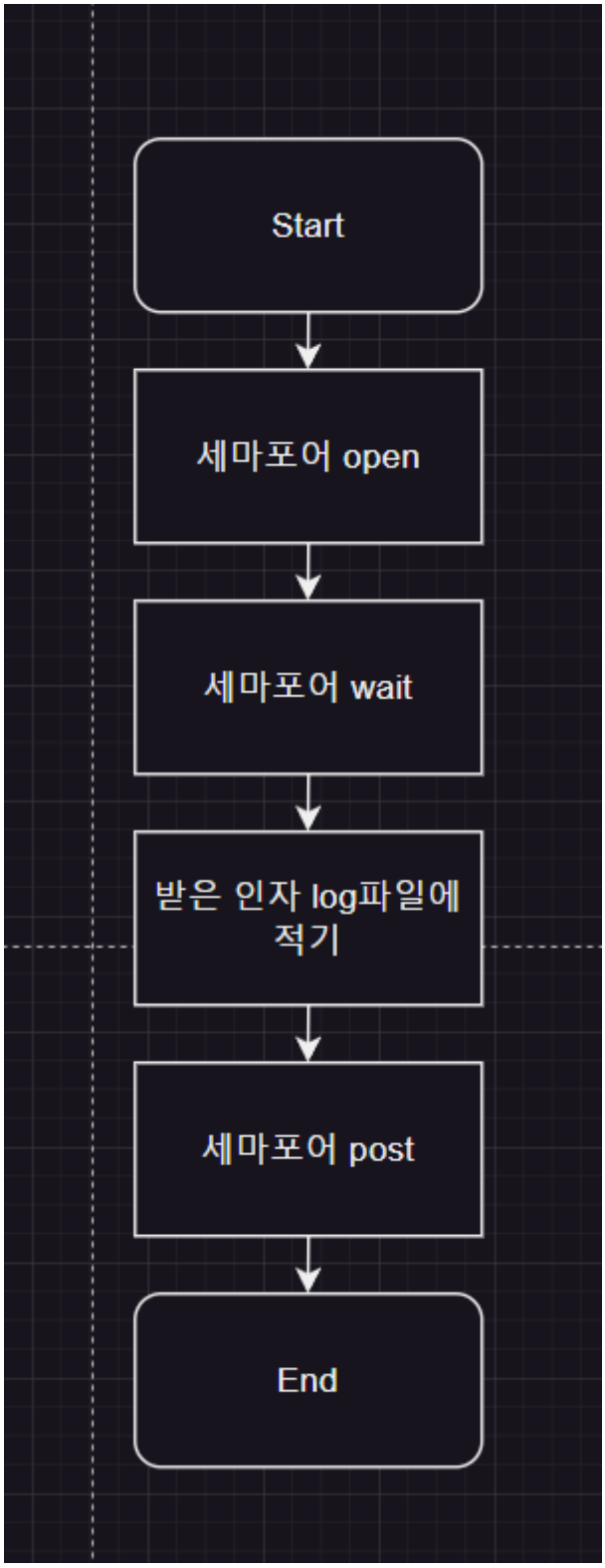
1. Introduction

3-2 에서 구현했던 기능에서 Client 가 요청한 경로 부분을 출력에 추가하고 연결 지속 시간도 추가한다. 클라이언트의 연결이 disconnected 될 때 출력한다. 단위는 마이크로 초(us)이며 클라이언트 연결 종료 시간 까지만 출력한다. 즉 sleep(5)의 시간은 제외한다. 로그 파일 작성시, child 프로세스마다 생성한 thread 를 이용하여 기록한다. 즉 thread 구동 함수가 string 을 parameter 로 받아 이를 file 에 기록하게끔 구현한다.

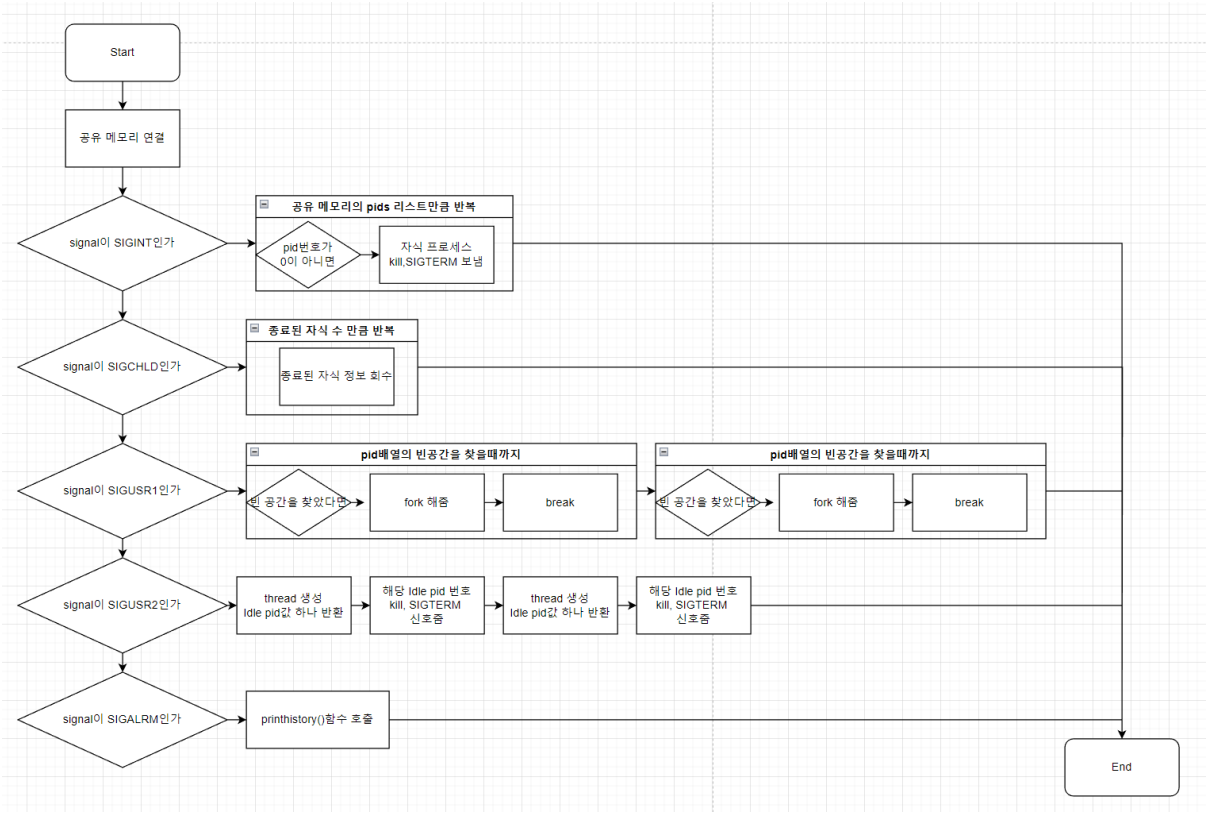
2. Flow chart



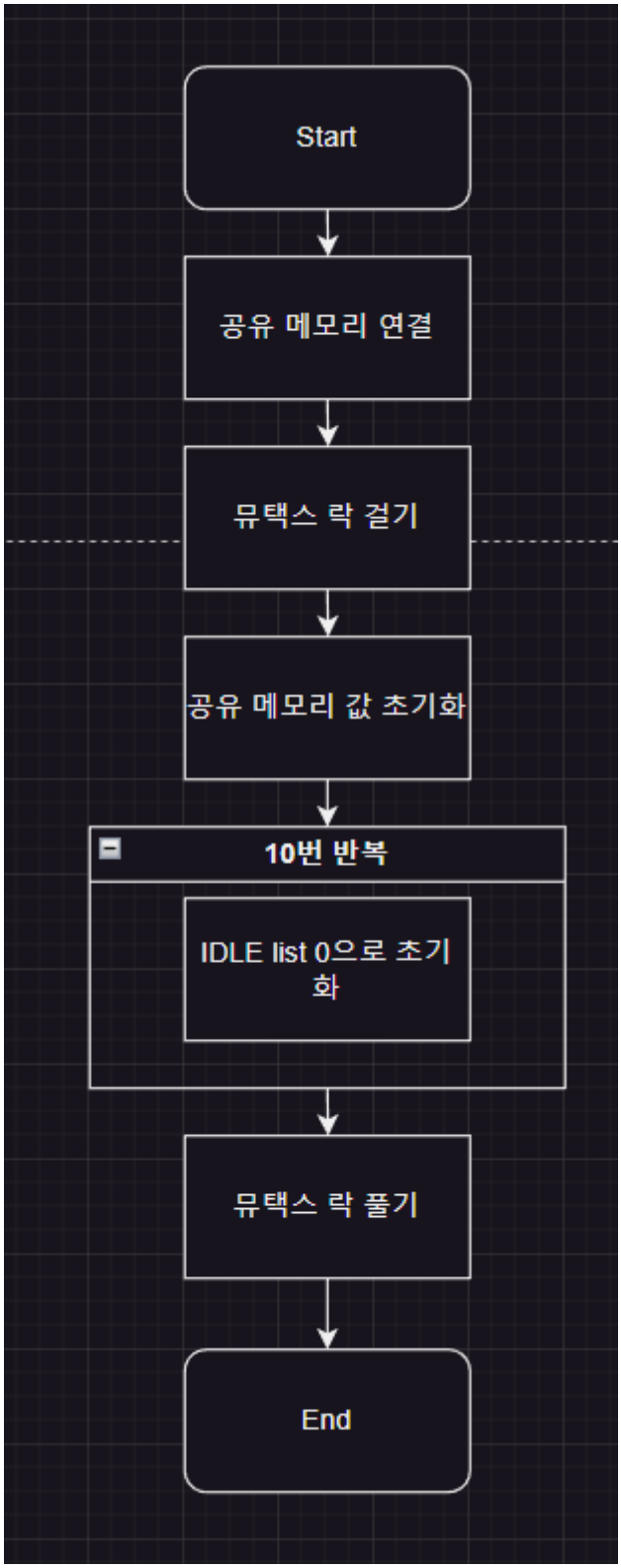
main 함수의 flow chart 이다.



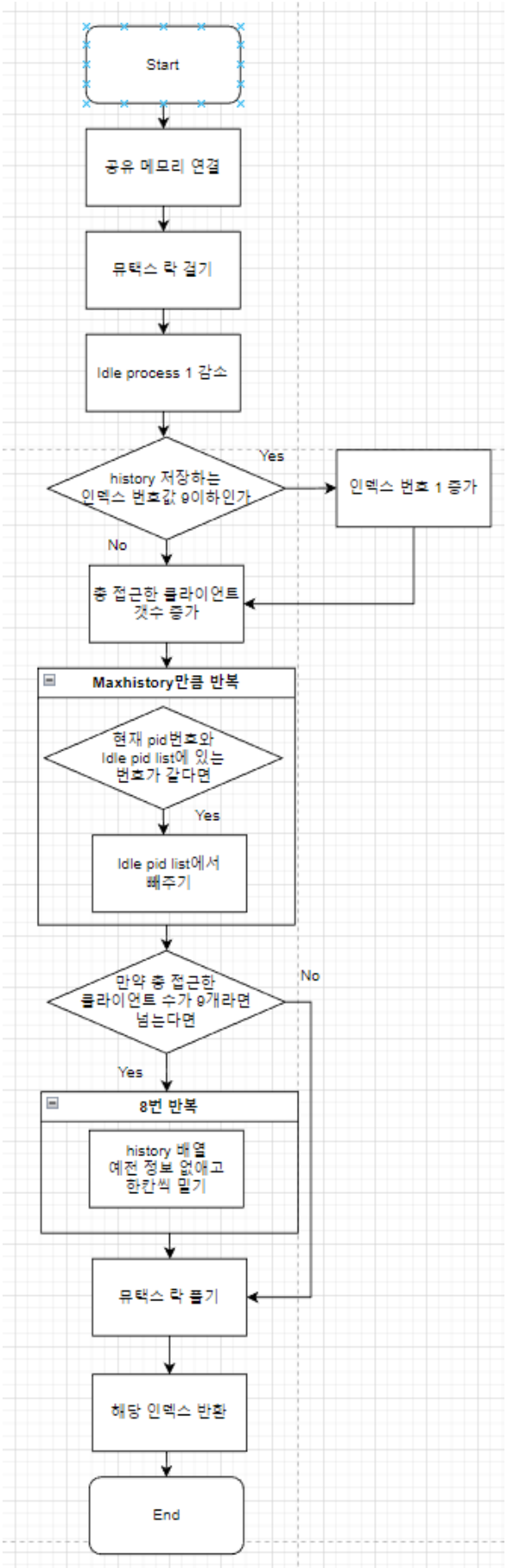
print_log 함수이다.



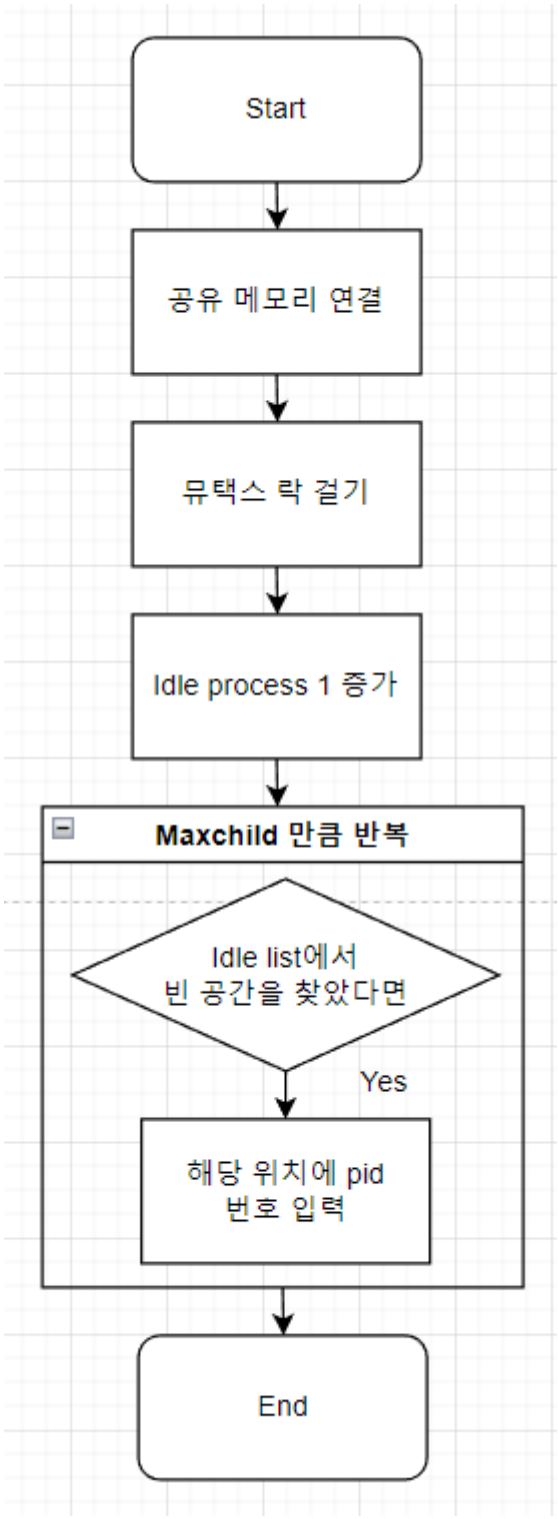
alarmHandler 함수이다


Start
공유 메모리 연결
뮤택스 락 걸기
공유 메모리 값 초기화
10번 반복
IDLE list 0으로 초기화
뮤택스 락 풀기
End

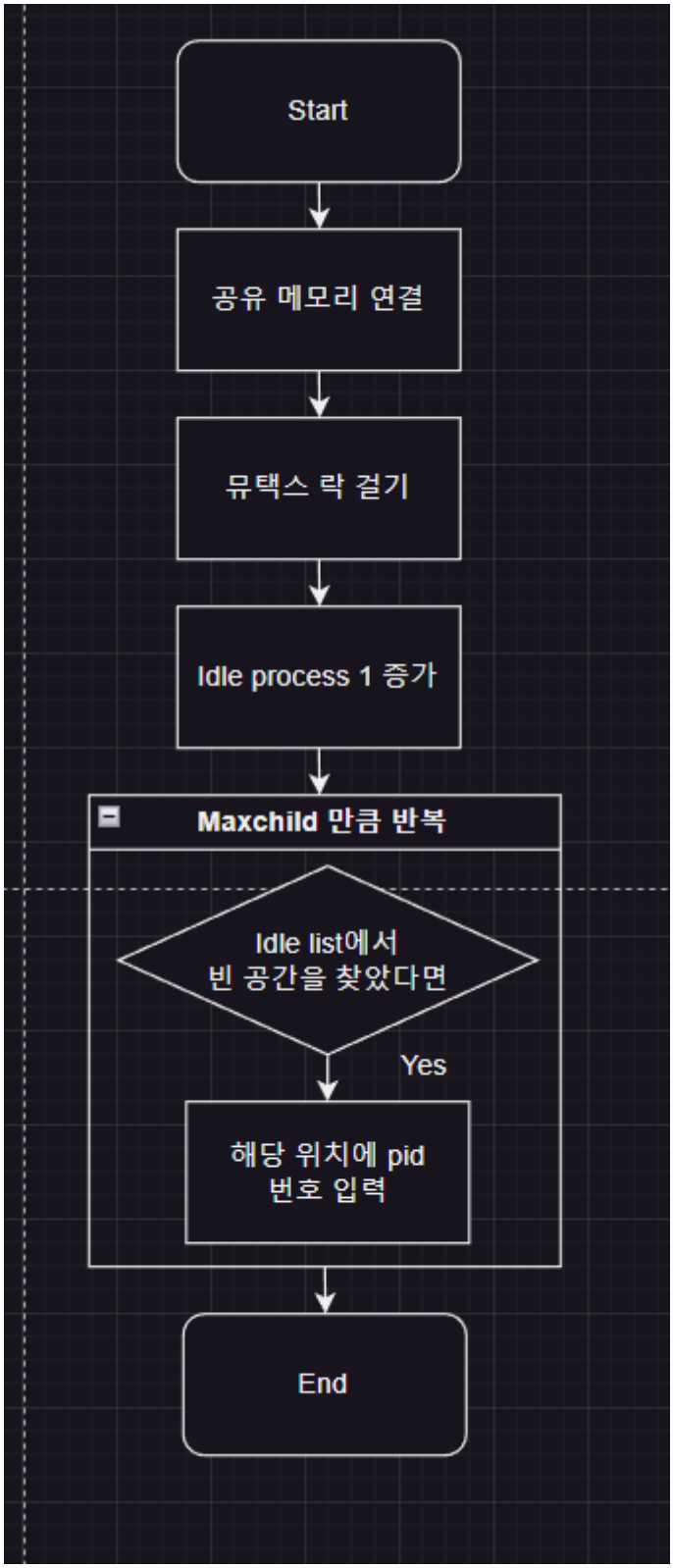
doit1 함수이다.



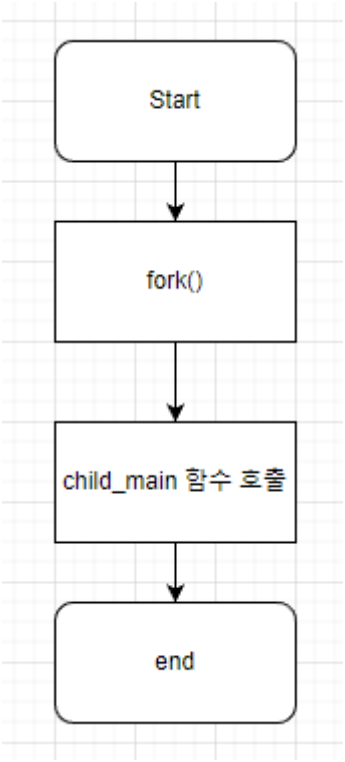
doit_dec 함수이다.



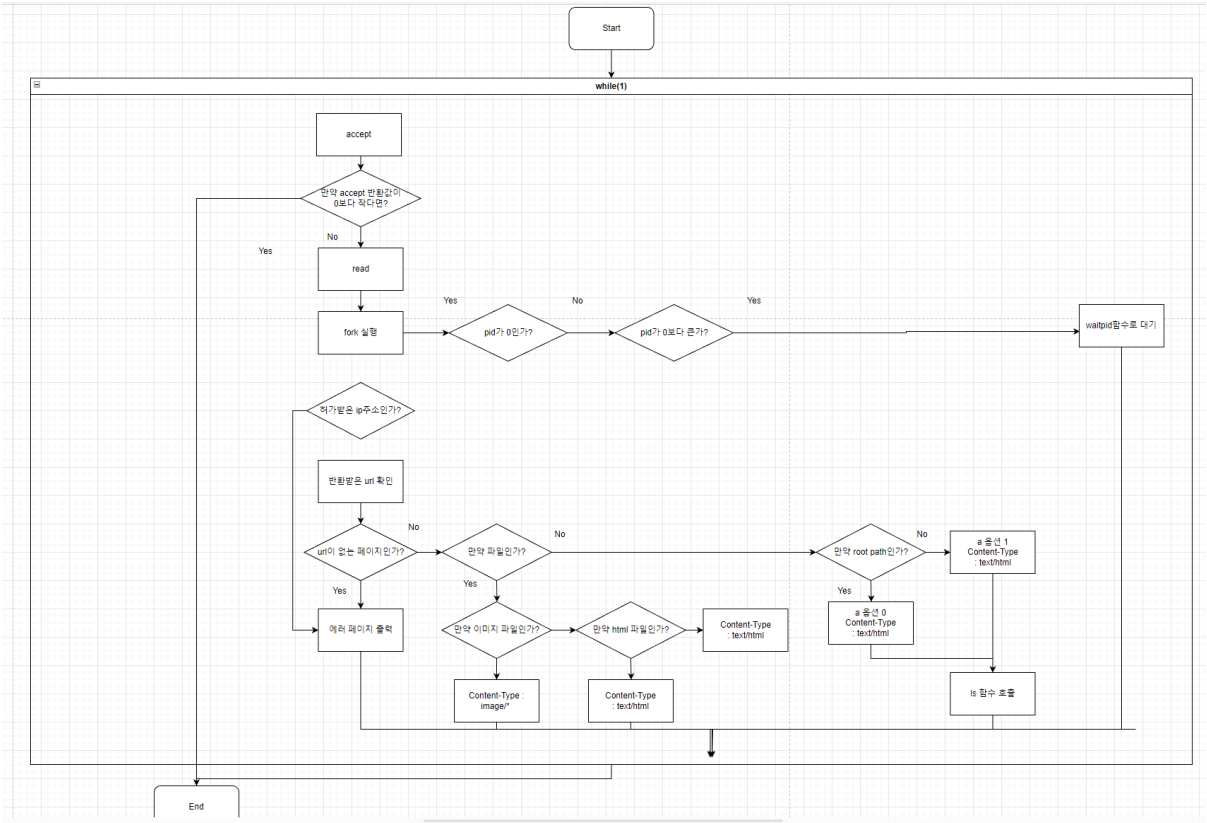
doit_disconnect 함수이다. disconnect 되었을 때의 Idle process 를 찾는 것은 doit_count 와 기능이 똑같고 print 문만 다를 뿐입니다.


Start
공유 메모리 연결
뮤택스 락 걸기
Idle process 1 증가
Maxchild 만큼 반복
Idle list에서
빈 공간을 찾았다면
Yes
해당 위치에 pid
번호 입력
End

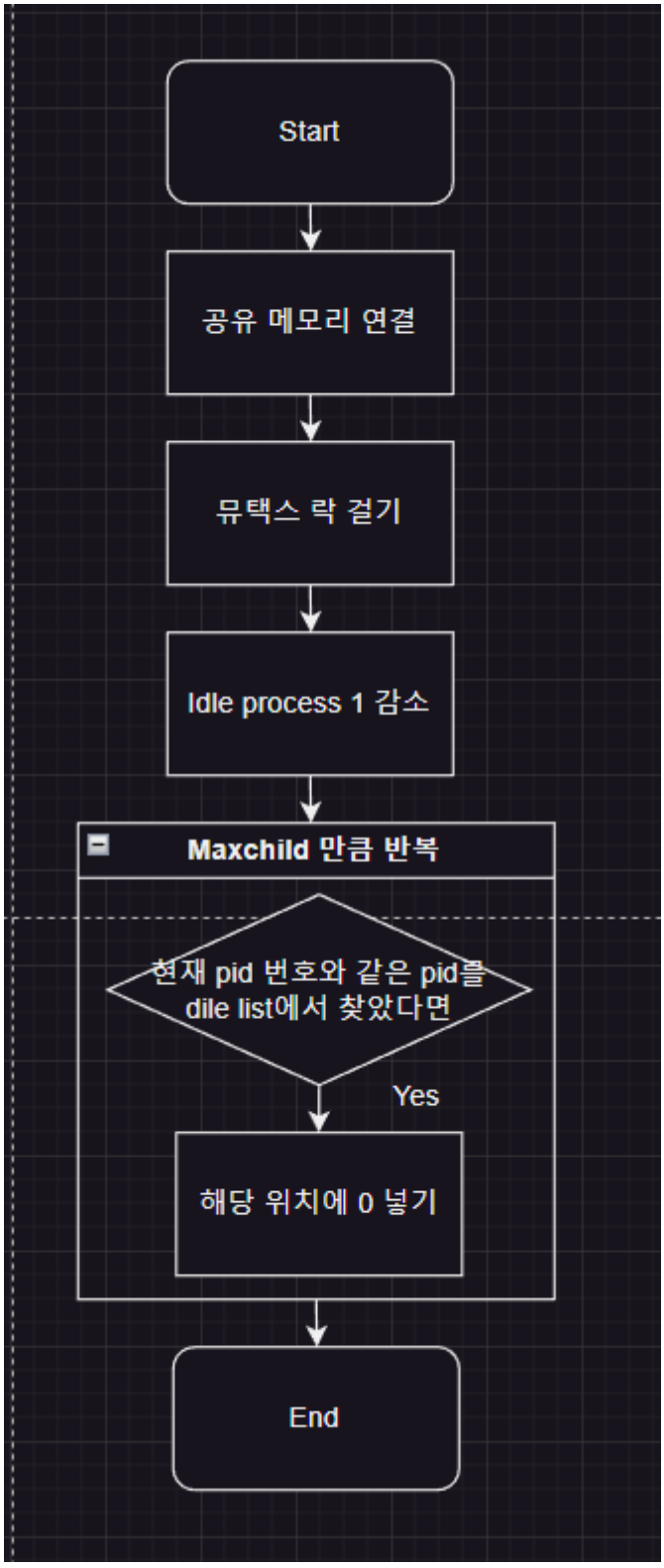
doit2 함수이다.



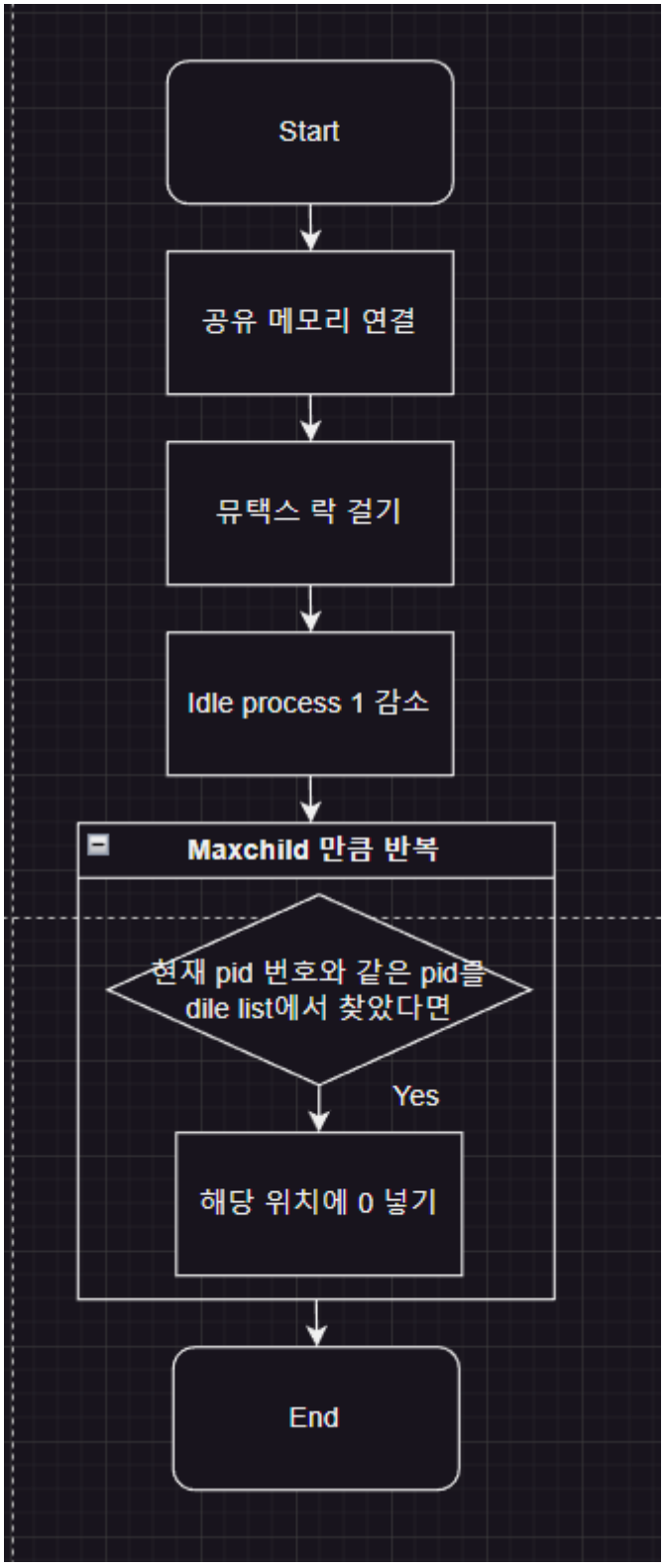
child_make 함수이다.



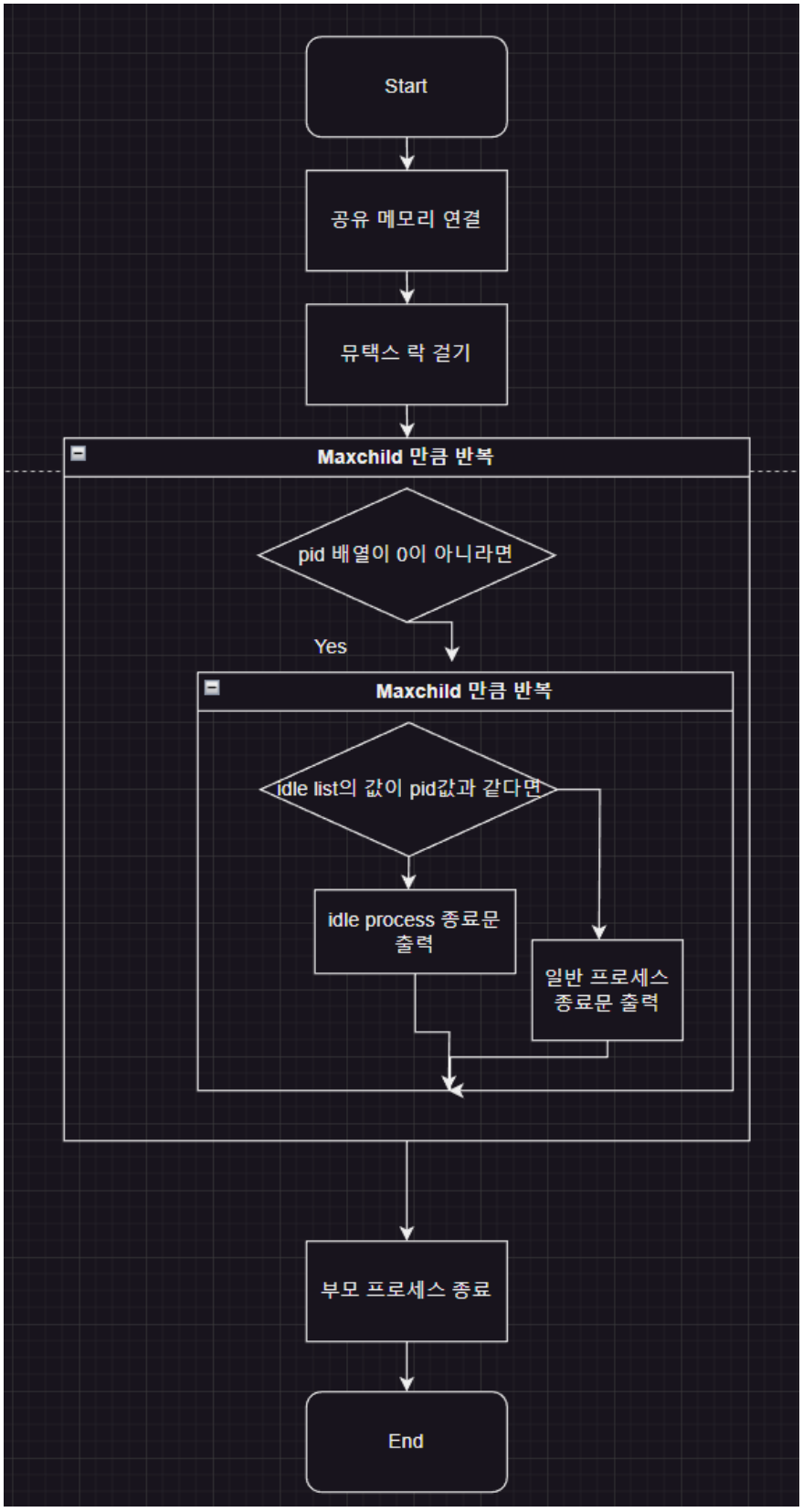
child_main 함수이다



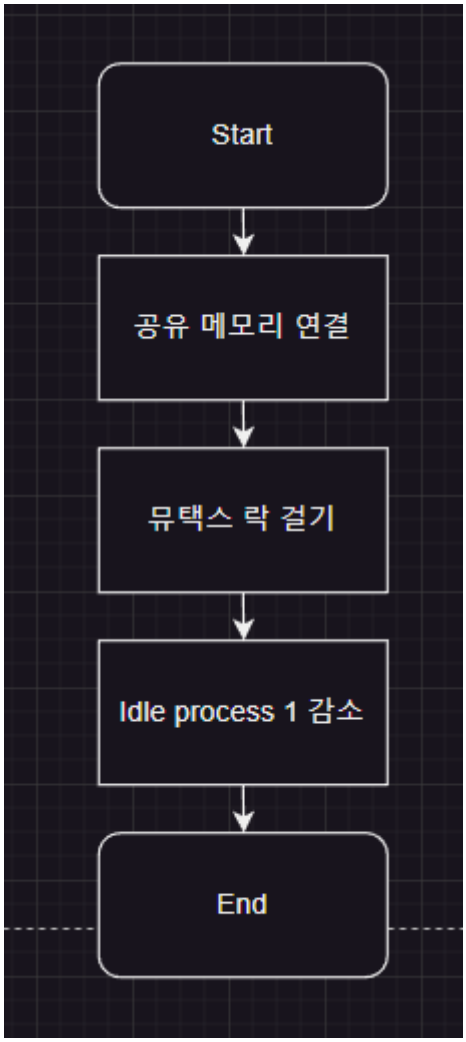
doit3 함수의 순서도이다.



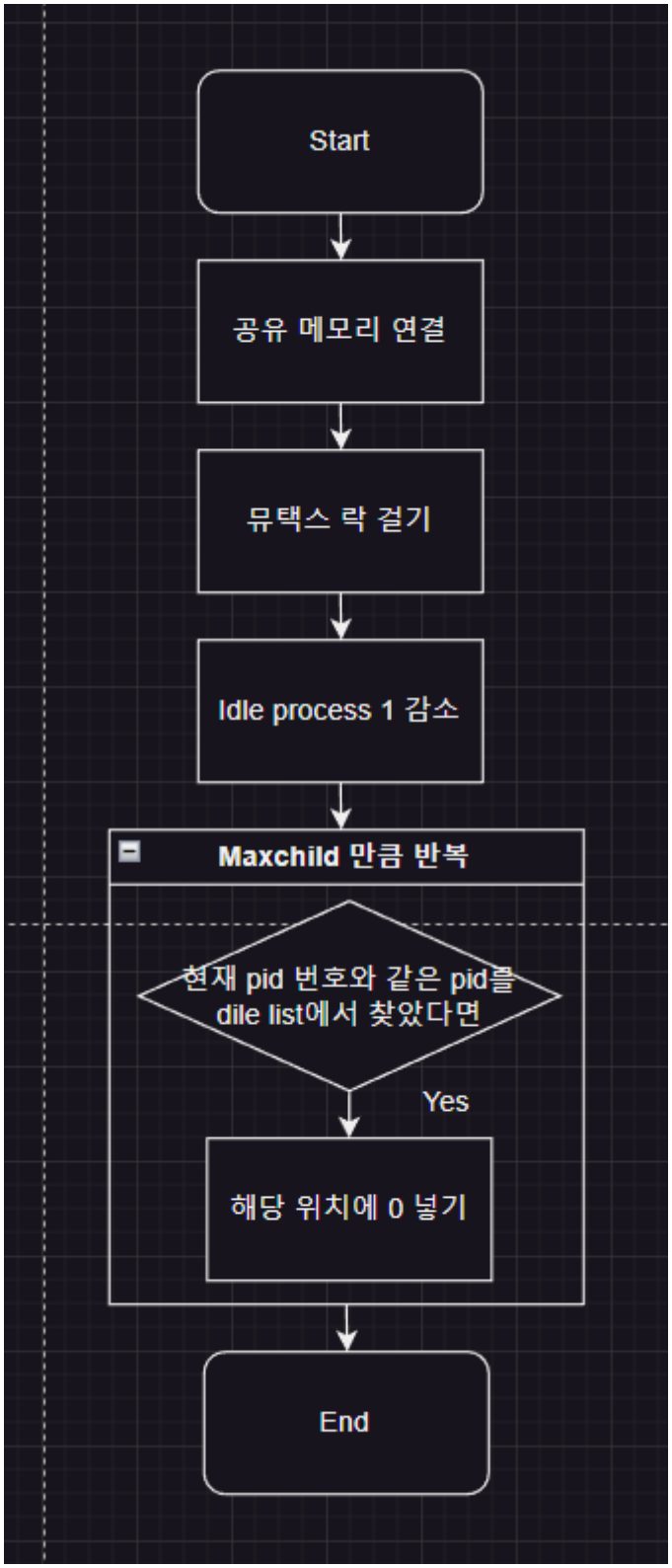
doit4 함수의 순서도이다. doit3 과 다른점은 print 문만 다르다.



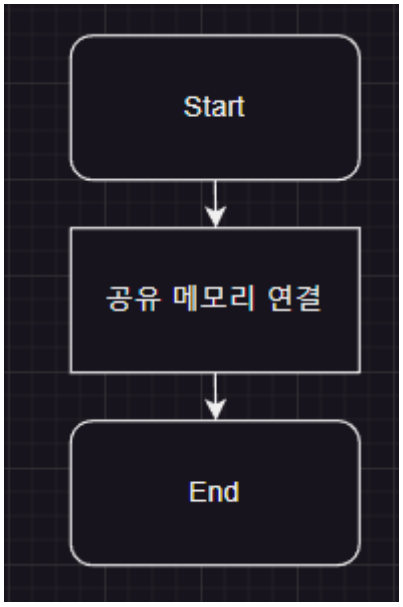
doit5 의 순서도이다.



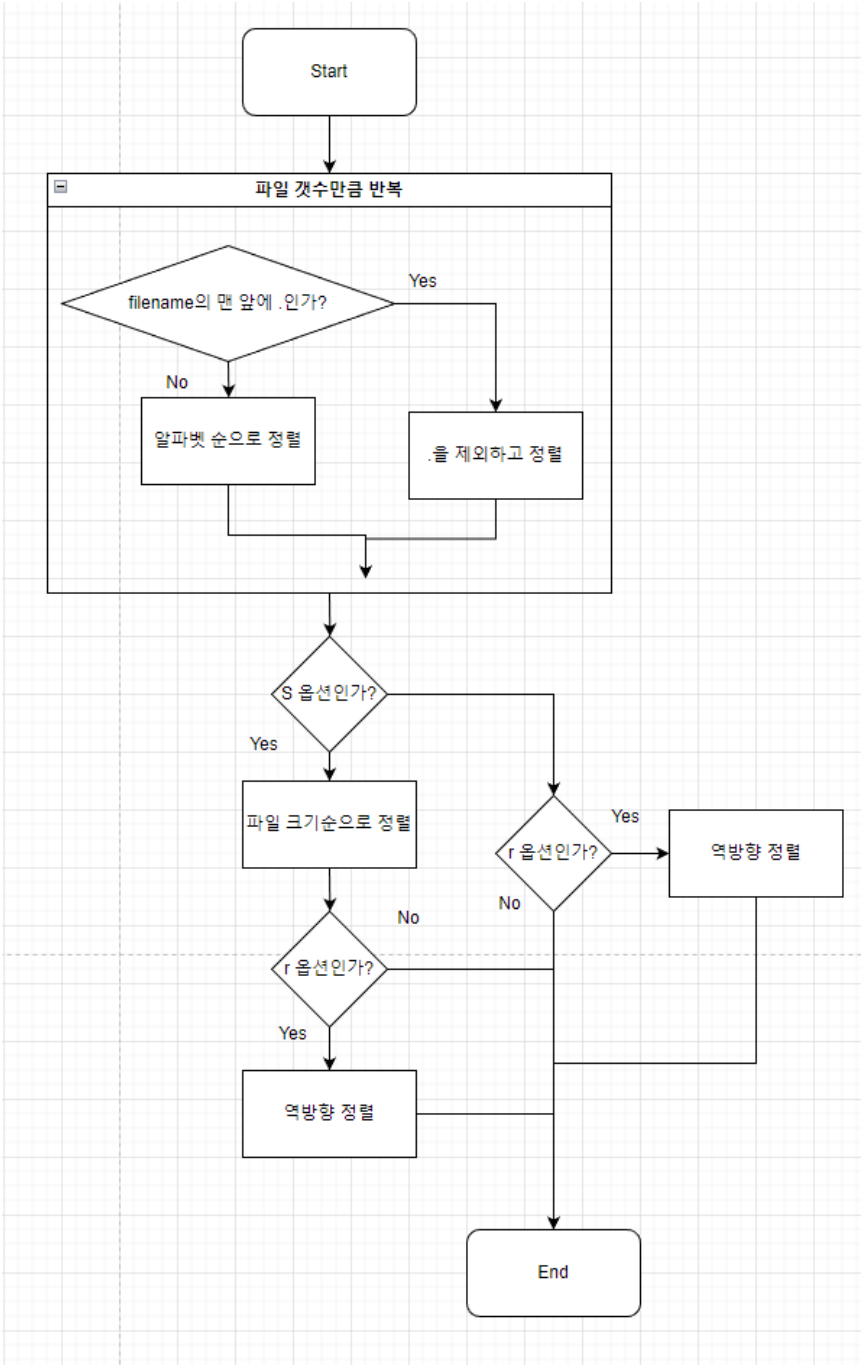
doit7 의 순서도이다.



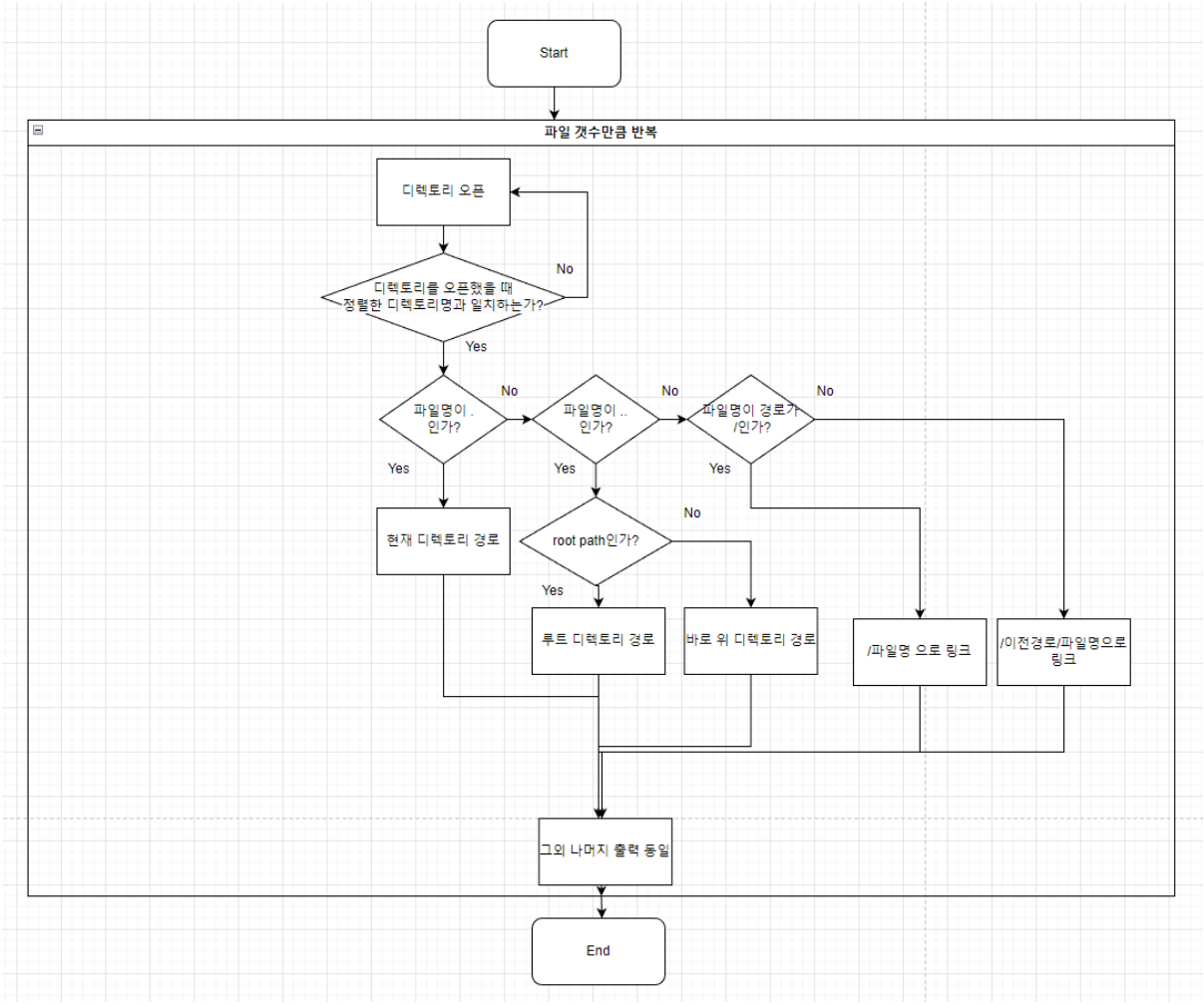
doit8 의 순서도이다.



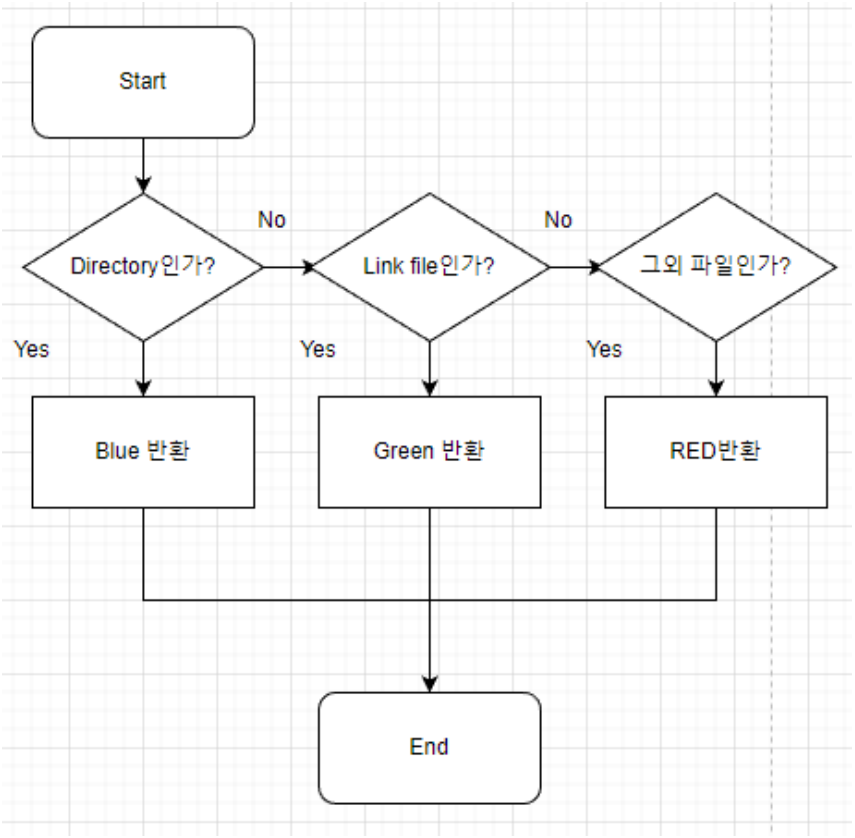
doit_nothing 함수의 순서도이다.



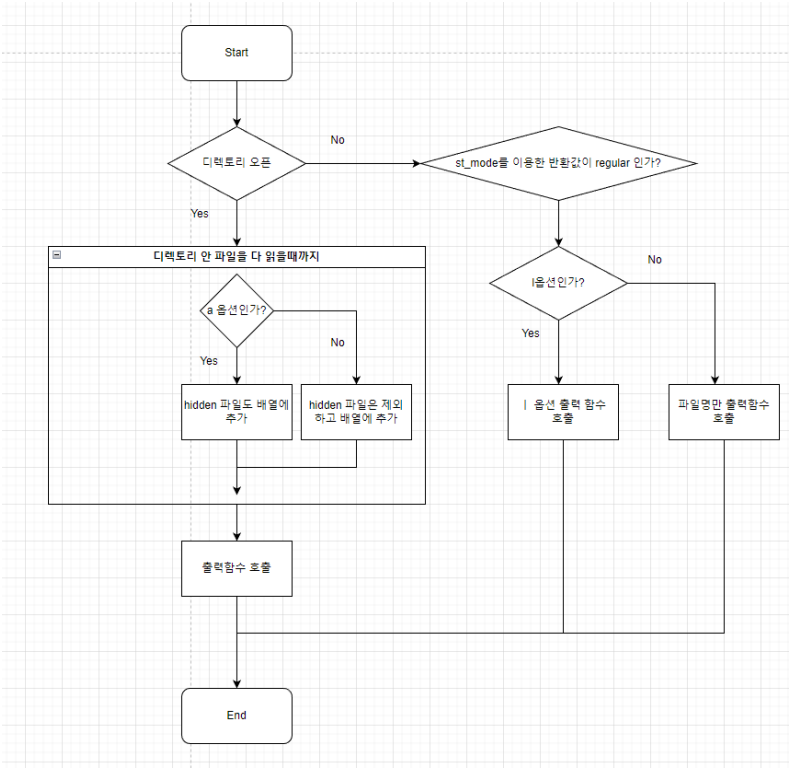
sorting 함수의 순서도이다.



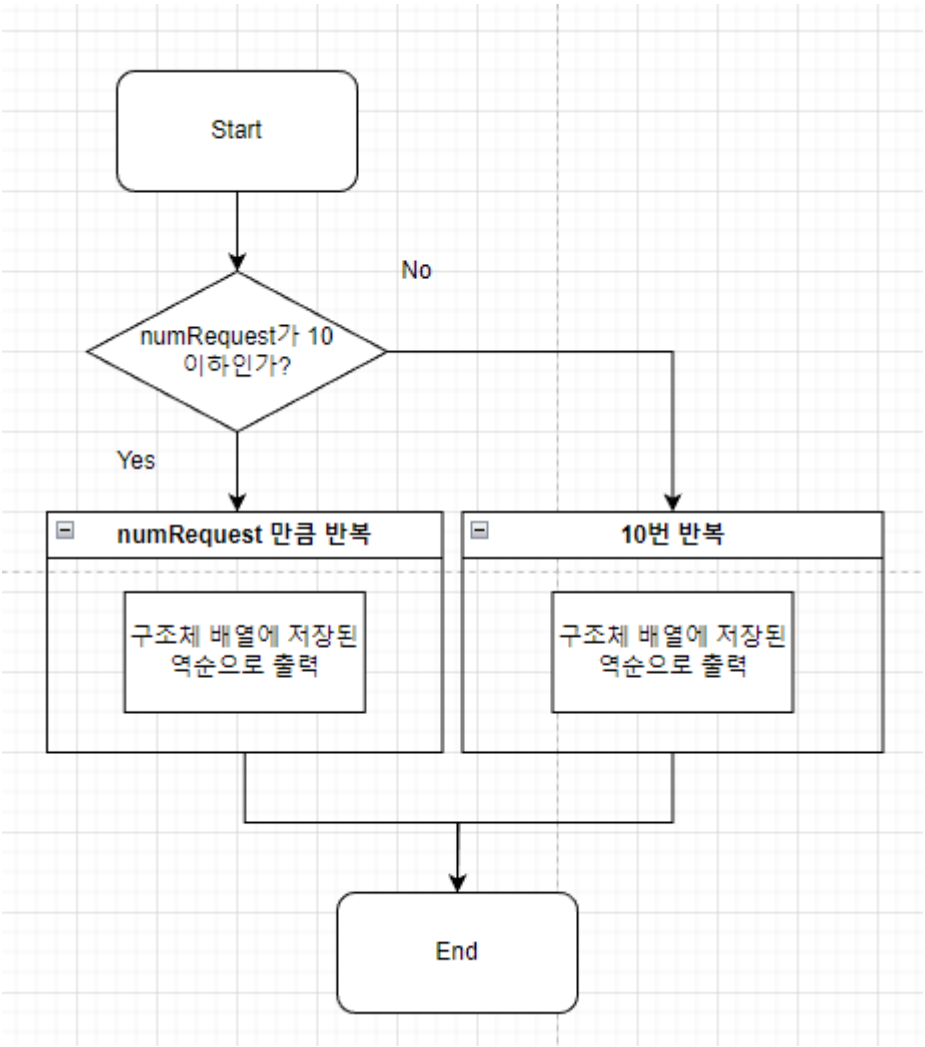
print_file_info 함수의 순서도이다.



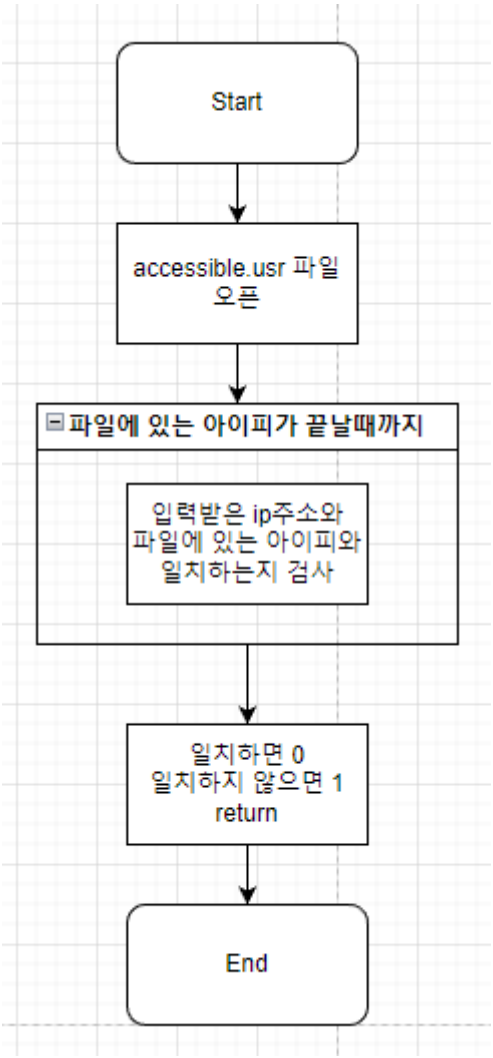
print_filetype 함수의 순서도이다.


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

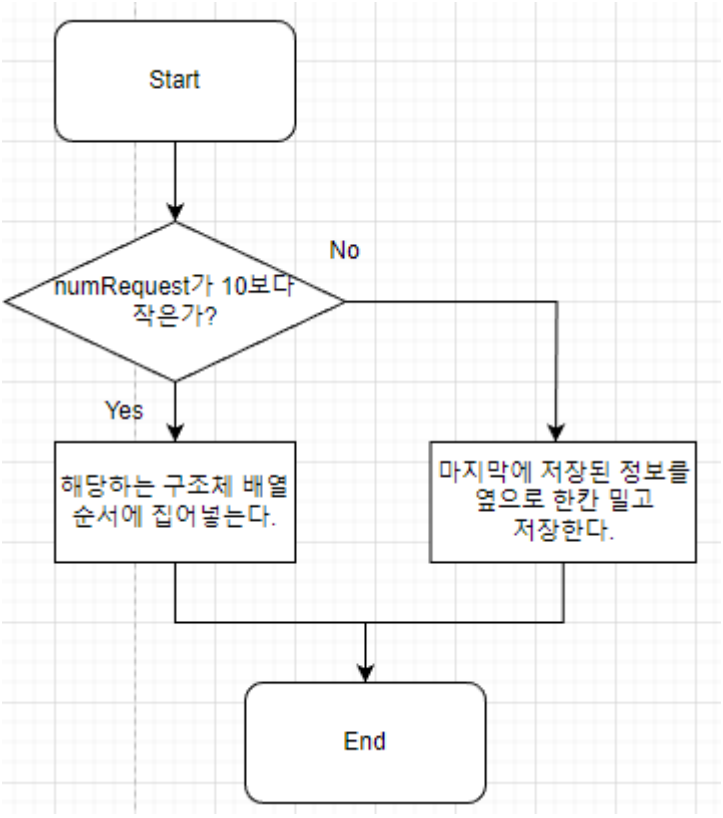
ls 함수의 순서도이다.



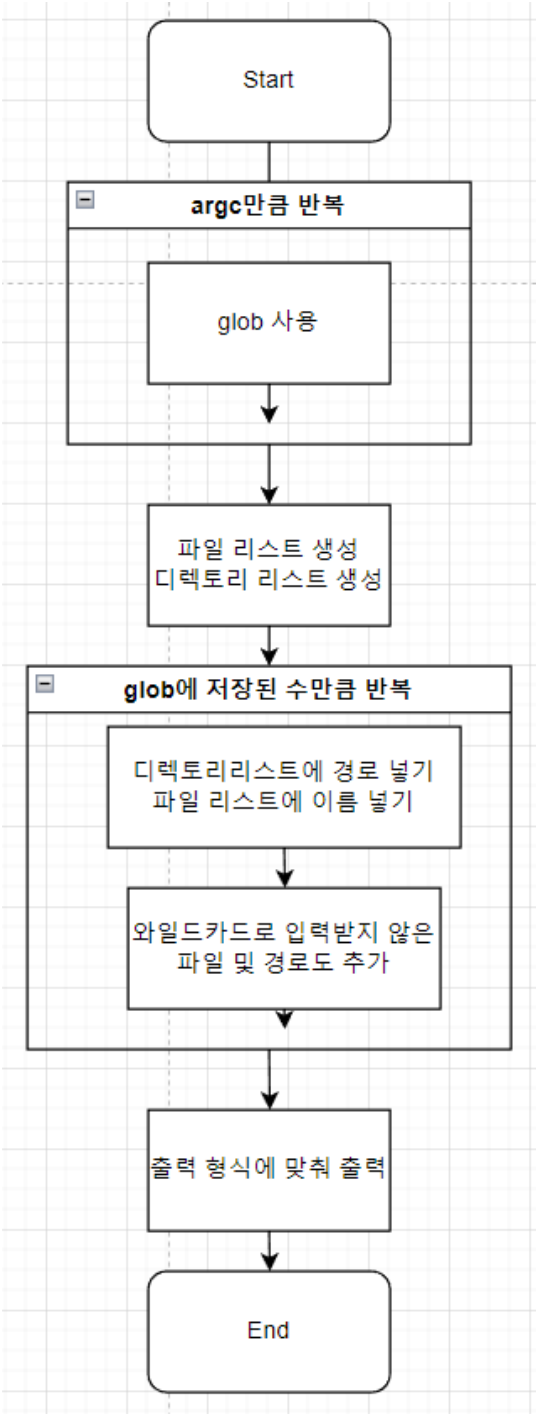
printHistory 함수이다.



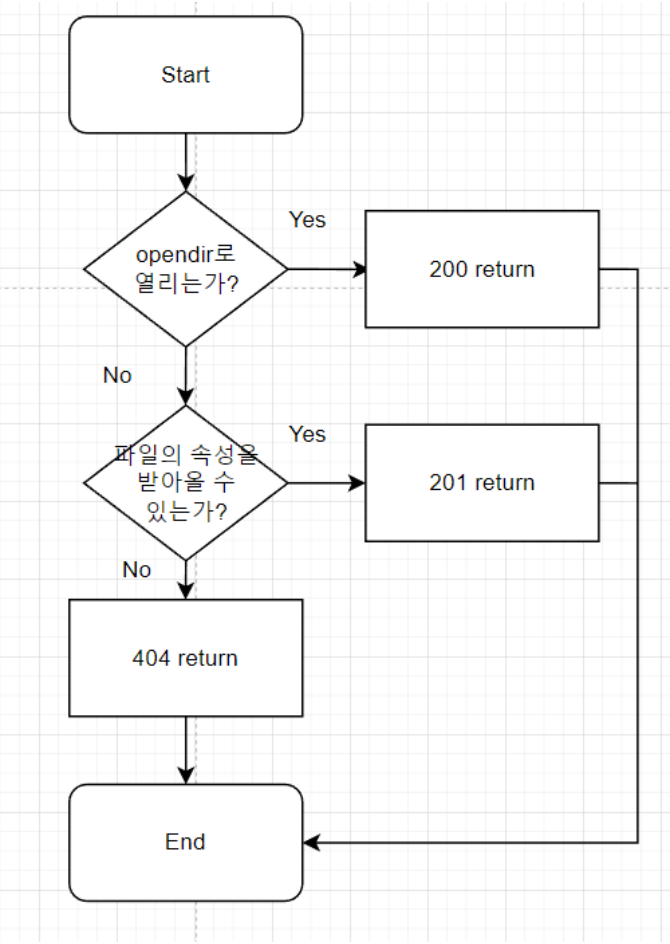
is_ip_allowed 함수이다.



addHistory 함수이다.



wild card 의 순서도이다.



check_404 의 순서도이다.

3. sudo code

```
void alarmHandler(int signum)
{
    공유 메모리 선언 및 사용 가능하게 연결하기
    if (SIGINT 신호가 들어왔을때)
    {
        쓰레드 생성(자식 종료문 호출)
        쓰레드 대기
         부모 프로세스 종료
    }
    else if (SIGCHLD 신호가 들어왔을때)
    {
        종료된 자식들 수거
     }
    else if(SIGUSR1 신호가 들어왔을 때)
```

```
        {
                fork 2번 해준다
        }
    else if(SIGUSR2 신호가 들어온 경우) //if signal is SIGUSR2
    {
        쓰레드 생성해서 Idle process 아무거나 하나 pid 번호 반환하게끔 함
        해당 pid 번호 가진 자식 process 종료
        쓰레드 생성해서 Idle process 아무거나 하나 pid 번호 반환하게끔 함
        해당 pid 번호 가진 자식 process 종료

    }
    else if (SIGALRM) // if parent process alarm
    {
         히스토리 출력

    }
    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)
{
        http.conf 파일 읽기
        http.conf 안에 적혀있는
        MaxChilds 값, MaxIdleNum 값, MinIdleNum 값
        StartProcess 값, MaxHistory 값 셋팅해주기
        시그널들 호출
        소켓 함수 호출
        setsockopt 로 셋팅
        bind 함수로 소켓과 구조체 정보 binding
        listen 함수로 client 로부터 connection 을 위해 대기상태로 변경


        쓰레드 생성
        쓰레드 대기
        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 *print_log(void* vptr)
{
    세마포어 오픈
    세마포어 close
    세마포어 다시 오픈
    세마포어 wait
    로그파일에 입력받은 인자 적기
    세마포어 post
}

void alarmHandler(int signum)
```

```
{
    if (SIGCHILD가 신호로오면)
    {
       waitpid 함수를 이용한 자식 프로세스 pid > 0일때까지 반복
    }
    else if (signal == SIGALRM)
    {
        구조체 배열 선언
    }
    alarm(10); // set 10 second
}

void printHistory()
{
    if (저장된 history가 10개 이하라면) { //process number < 10
        저장된 것 만큼 반복해서 역순으로 출력
    }
    else
    {
        for (10번 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 10개보다 적다면) {
        그다음 위치에 프로세스 정보 저장
```

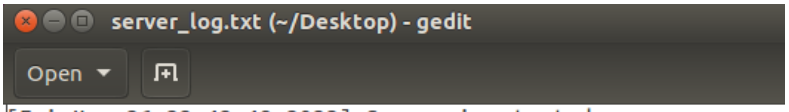
```
    } else {
        기존에 저장된 배열을 한칸 민다.
        밀어서 생긴 공간에 프로세스 정보 저장한다.
    }
    numRequests++;
}

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
    }

}
void *doit_dec(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        배열이 꽉차면 가장 오래된 history 밀기
        뮤택스 락 풀기
        저장할 배열 index 찾은 후 반환

}
void *doit_disconnect(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        생성한 Idle_list 에 pid 값 넣기
        뮤택스 락 풀기
}
void *doit1(void *vptr)
{
        공유 메모리 연결
        뮤택스 잠금
        공유 메모리 값 초기화
        뮤택스 잠금 해제
}

void *doit2(void *vptr)
{
        공유 메모리 연결
```

```
        뮤택스 잠금
        idle list 에서 빈 공간에 현재 pid 번호 넣기
        뮤택스 잠금 해제
        fork 가 되었음을 알리는 출력문
}
void *doit3(void *vptr)
{
        공유 메모리 연결
        뮤택스 잠금
        idle count -1 해주기
        idle list 에서 현재 pid 번호가 같은 곳을 찾기
        찾은 위치에 0 넣기
        뮤택스 잠금 해제

}
void *doit4(void *vptr)
{
        공유 메모리 연결
        뮤택스 잠금
        idle count -1 해주기
        idle list 에서 현재 pid 번호가 같은 곳을 찾기
        찾은 위치에 0 넣기
        뮤택스 잠금 해제
}
void *doit_dec(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        배열이 꽉차면 가장 오래된 history 밀기
        뮤택스 락 풀기
        저장할 배열 index 찾은 후 반환

}
void *doit_disconnect(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        idleprocess 값 +1
        idle list 에서 빈 공간 찾은 후 해당 공간에 pid 번호 넣기
        뮤택스 락 풀기
}

void *doit5(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        pid 번호와 list 에 있는 pid 번호가 같다면
        idle list 종료문 출력
```

```
        같지 않다면 process 종료문 출력
        idleprocess 값 -1
        해당 pid 번호 프로세스 SIGTERM
        뮤택스 락 풀기



void *doit7(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        idle count -1
        뮤택스 락 풀기
}

void *doit8(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        idle count -1
        pids 에 있는 list 중에 0 이 아닌 것 선택
        idle list 에서 pids 에 있는 번호와 같은 것 선택 후 해당 위치에 0 넣기
        뮤택스 락 풀기
}
void *doit_nothing(void *vptr)
{
        공유 프로세스 생성 및 연결
        뮤택스 락 걸기
        뮤택스 락 풀기

}
```

결과화면




```
[Fri May 26 22:42:48 2023] Server is started.
[Fri May 26 22:42:48 2023] 6183 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 1
[Fri May 26 22:42:48 2023] 6180 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 2
[Fri May 26 22:42:48 2023] 6182 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 3
[Fri May 26 22:42:48 2023] 6181 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 4
[Fri May 26 22:42:48 2023] 6184 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 5
=============== New Client ===============
TIME : [Fri May 26 22:42:51 2023]
URL : /aa
IP : 127.0.0.1
Port : 3232
==========================================

[Fri May 26 22:42:51 2023] IdleProcessCount : 4
=============== New Client ===============
TIME : [Fri May 26 22:42:52 2023]
URL : /
IP : 127.0.0.1
Port : 4768
==========================================

[Fri May 26 22:42:52 2023] IdleProcessCount : 3
[Fri May 26 22:42:52 2023] 6204 Process is forked.
[Fri May 26 22:42:52 2023] IdleProcessCount : 4
[Fri May 26 22:42:52 2023] 6203 Process is forked.
[Fri May 26 22:42:52 2023] IdleProcessCount : 5
========== Disconnected Client ===========
TIME : [Fri May 26 22:42:51 2023]
URL : /aa
IP : 127.0.0.1
Port : 3232
CONNECTIONG TIME: 488(us)

[Fri May 26 22:42:56 2023] IdleProcessCount : 6
==========================================
========== Disconnected Client ===========
TIME : [Fri May 26 22:42:52 2023]
URL : /
IP : 127.0.0.1
Port : 4768
CONNECTIONG TIME: 2707(us)

[Fri May 26 22:42:57 2023] IdleProcessCount : 7
==========================================

[Fri May 26 22:42:57 2023] IdleProcessCount : 6
[Fri May 26 22:42:57 2023] 6180 Process is terminated.

[Fri May 26 22:42:57 2023] IdleProcessCount : 5
[Fri May 26 22:42:57 2023] 6181 Process is terminated.

[Fri May 26 22:42:58 2023] 6182 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 4
[Fri May 26 22:42:58 2023] 6183 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 3
[Fri May 26 22:42:58 2023] 6184 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 2
[Fri May 26 22:42:58 2023] 6203 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 1
[Fri May 26 22:42:58 2023] 6204 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 0
[Fri May 26 22:42:58 2023] Server is terminated.
```

저번 과제에서 추가된 URL 과 connectiong time 이 추가된 것을 확인할 수 있다.

```
[Fri May 26 22:42:48 2023] Server is started.
[Fri May 26 22:42:48 2023] 6183 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 1
[Fri May 26 22:42:48 2023] 6180 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 2
[Fri May 26 22:42:48 2023] 6182 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 3
[Fri May 26 22:42:48 2023] 6181 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 4
[Fri May 26 22:42:48 2023] 6184 Process is forked.
[Fri May 26 22:42:48 2023] IdleProcessCount : 5
=============== New Client ===============
TIME : [Fri May 26 22:42:51 2023]
URL : /aa
IP : 127.0.0.1
Port : 3232
=========================================

[Fri May 26 22:42:51 2023] IdleProcessCount : 4
=============== New Client ===============
TIME : [Fri May 26 22:42:52 2023]
URL : /
IP : 127.0.0.1
Port : 4768
=========================================

[Fri May 26 22:42:52 2023] IdleProcessCount : 3
[Fri May 26 22:42:52 2023] 6204 Process is forked.
[Fri May 26 22:42:52 2023] IdleProcessCount : 4
[Fri May 26 22:42:52 2023] 6203 Process is forked.
[Fri May 26 22:42:52 2023] IdleProcessCount : 5
========== Disconnected Client ===========
TIME : [Fri May 26 22:42:51 2023]
URL : /aa
IP : 127.0.0.1
Port : 3232
CONNECTIONG TIME: 488(us)

[Fri May 26 22:42:56 2023] IdleProcessCount : 6
=========================================
========== Disconnected Client ===========
TIME : [Fri May 26 22:42:52 2023]
URL : /
IP : 127.0.0.1
Port : 4768
CONNECTIONG TIME: 2707(us)

[Fri May 26 22:42:57 2023] IdleProcessCount : 7
=========================================

[Fri May 26 22:42:57 2023] IdleProcessCount : 6
[Fri May 26 22:42:57 2023] 6180 Process is terminated.

[Fri May 26 22:42:57 2023] IdleProcessCount : 5
[Fri May 26 22:42:57 2023] 6181 Process is terminated.
^C
[Fri May 26 22:42:58 2023] 6182 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 4
[Fri May 26 22:42:58 2023] 6183 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 3
[Fri May 26 22:42:58 2023] 6184 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 2
[Fri May 26 22:42:58 2023] 6203 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 1
[Fri May 26 22:42:58 2023] 6204 Process is terminated.
[Fri May 26 22:42:58 2023] IdleProcessCount : 0
[Fri May 26 22:42:58 2023] Server is terminated.
kw2019202031@ubuntu:~/Desktop$ ps
```

정상적으로 terminal 에서도 나오는 것을 확인할 수 있다.

고찰

세마포어를 이용해 값이 섞이지 않고 정확하게 나오는 것을 보며 이번 과제에서 가장 편안한 기분이 들었다. 왜 세마포어를 사용해야 하는 것인지 이해했다. 내가 짠 과제가 완벽하다는 생각이 세마포어를 이용해 결과를 테스트하며 느꼈다. 마지막 웹 서버 구현이 끝났다고 생각하니 아쉽지만 기분이 좋다.

reference

강의자료 참고