3-2

금3,4

과목명	시스템프로그래밍
담당교수	김태석 교수님
학과	컴퓨터정보공학부
학년	3학년
학번	2019202009
이름	서여지
제출일	21.05.30(일)

KWANGWOON UNIVERSITY

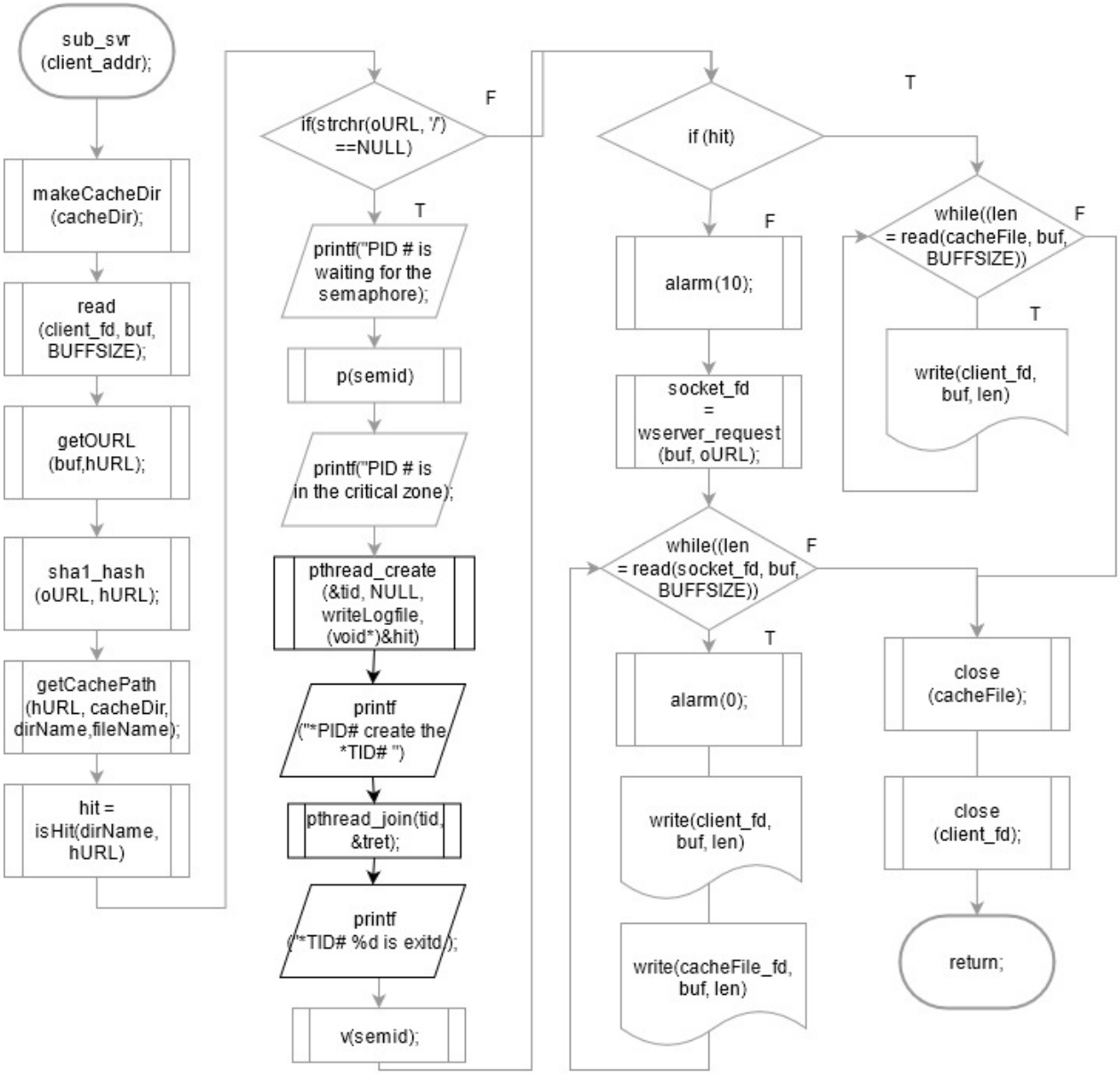
1. Introduction //과제 소개 – 5줄 이하 background 제외

이번 과제는 3-1 과제에서 critical section으로 지정한 logfile write 과정을 별도의 thread를 생성하여 처리하는 것이다. POSIX thread를 이용하여 logfile을 작성하는 함수를 thread에서 수행하도록 한다. terminal 출력에 critical section 내부에서 새로 생성된 thread에 대한 정보를 추가로 출력해야 한다.

2. Flow Chart //코드 작성 순서도

(1) sub_svr

thread를 이용하는 pthread_create와 pthread_join이 추가되었다. 기존의 writeLogfile함수를 thread에서 이용하기 위해 전역변수로 선언한 포인터를 함수 내부에서 사용하도록 수정하였다.

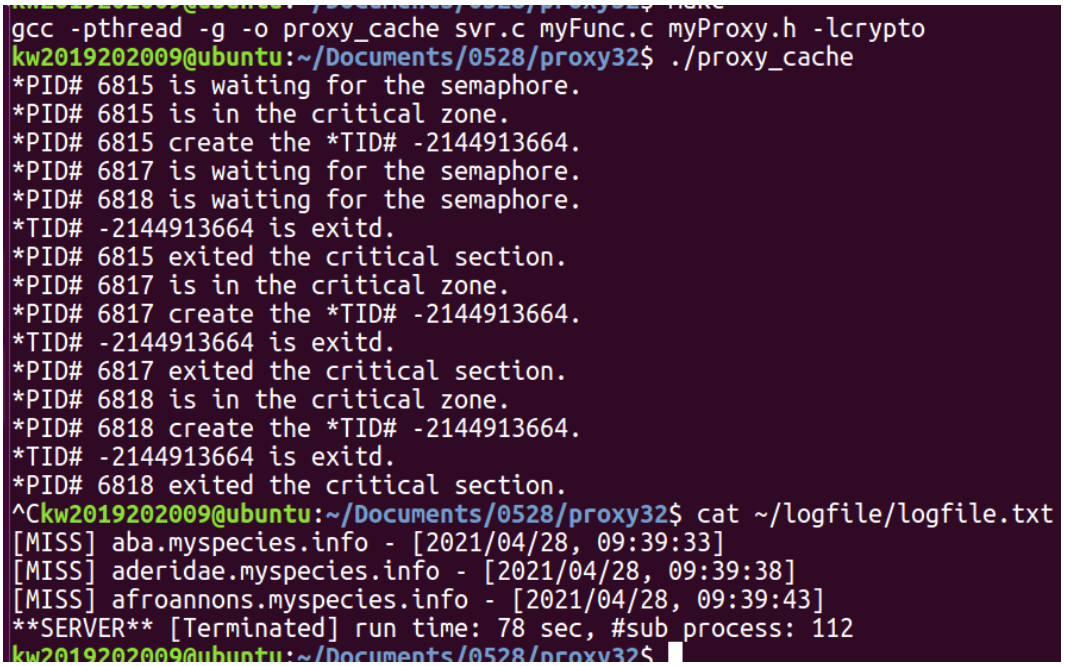


3. Pseudo code //알고리즘

```
void sub_svr(struct sockaddr_in   client_addr) {
    makeCacheDir(cacheDir);

    read(client_fd, buf, BUFFSIZE);
    strcpy(tmp, buf);
    getOURL(tmp, oURL);
    sha1_hash(oURL, hURL);
    getCachePath(hURL, cacheDir, dirName, fileName);

    hit = isHit(dirName, hURL);
    if(oURL is webserver address) {
        printf("*PID# is waiting for the semaphore₩n");
        p(semid);
        printf("*PID# is in the critical zone₩n");
        pthread_create(&tid, NULL, writeLogfile, (void*)hit);
        if(pthread error) return;      // thread error
        printf("*PID# create the *TID₩n");
        pthread_join(tid, &tret);
        printf("*TID# is exited.₩n");
        v(semid);
        printf("*PID# exited the critical section₩n");
    }

    if(hit) {
        cacheFile = open(fileName, "r");
        while((read(cacheFile, buf, BUFFSIZE))>0)
            write(client_fd, buf, len);
        close(cacheFile);
    }
    else {
        alarm(10);
        socket_fd = wserver_requset(buf, oURL);
        cacheFile = open(fileName, "w");
        while((len = read(socket_fd, buf, sizeof(buf)){
            alarm(0);
            write(client_fd, buf, len);
            write(cacheFile, buf, len);
            buf = 0;
        }
        close(socket_fd);
        close(cacheFile);
    }
    close(client_fd);
    return;
}
```

4. 결과화면 //수행한 내용을 캡쳐 및 설명

3-1 과제와 동일한 방법으로 결과를 확인하였다. firefox에서 여러 개의 탭을 열어 첫 번째 자식 프로세스가 critical section의 sleep에서 벗어나기 전에 여러 개의 자식 프로세스를 생성한다. 터미널 출력 결과를 확인한다.



3개의 사이트에 한 번에 접속한 결과 위와 같은 결과가 나타났다. pid가 6815인 프로세스에서 생성한 thread가 종료되기 전까지 pid가 6817, 6818인 프로세스는 semaphore를 대기하며, critical section에 진입한 프로세스는 thread를 생성하여 logfile을 기록한 것을 확인할 수 있다.

5. 결론 및 고찰

이번 과제는 critical section의 logfile기록을 thread를 이용하여 진행하는 과제였다. hit여부와 url정보를 인자로 받아 logfile을 기록하던 writeLogfile함수를 수정하여 전역변수 포인터를 통해 url정보를 받고, pthread_create의 void*를 이용하여 hit여부를 전달받도록 하였다. thread를 생성하고, 자원을 회수하는 함수를 이용하여 과제를 구현하는 것에 큰 어려움은 겪지않았다.