# 시스템 프로그래밍 실습 Report

## 실습 제목: Proxy #1-3

## Multiple-Processing

실습일자: 2022년 4월 04일 (월)

제출일자: 2022년 4월 13일 (수)


학 과: 컴퓨터정보공학부

담당교수: 최상호 교수님

실습분반: 목요일 7,8교시

학 번: 2018202065

성 명: 박 철 준

- Introduction

1. 제목

Proxy 1-3 Multiple-Processing

2. 목적

가. 사용자 요청 처리를 위한 새로운 프로세스(“Sub process”)를 생성하고, 관리해야 한다.

나. 메인 프로세스는 터미널에 [(pid)]input CMD>를 출력하고, 사용자의 명령어 입력을 대기하여야한다.

다. connect 명령어 시 새로운 서브 프로세스를 생성하고, 해당 프로세스의 종료까지 메인프로세스는 대기하여야한다.

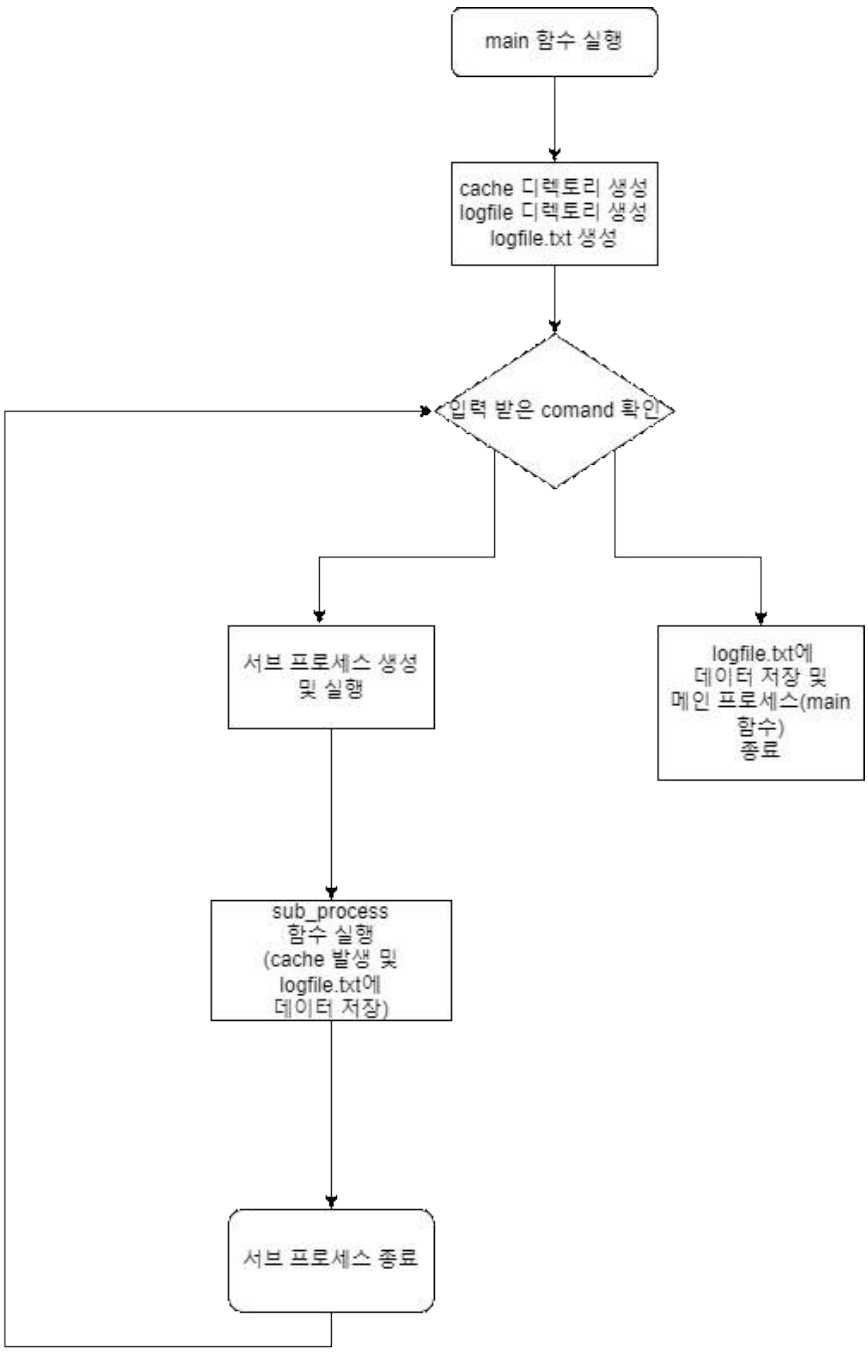
라. quit 명령어 입력 시 메인 프로세스는 종료되고 동작 시간, 생성한 child 프로세스 수 정보에 대한 log를 저장해야한다.

마. sub process는 사용자의 URL을 입력 받고, Proxy 1-2에서의 연산을 수행하여야 한다.

바. bye 입력 시 해당 서브 프로세스를 종료하여야한다.

● 실습 결과

1 proxy #1-2
-Flow Chart
main 함수
1-2에서 작성한 main함수는 이번 과제의 서브프로세스가 되어 sub_process함수로 이름을 변경하였고 새로운 메인함수를 설계하였다. 이것에 대한 플로 차트는 아래와 같다.



-Pseudo code

main함수

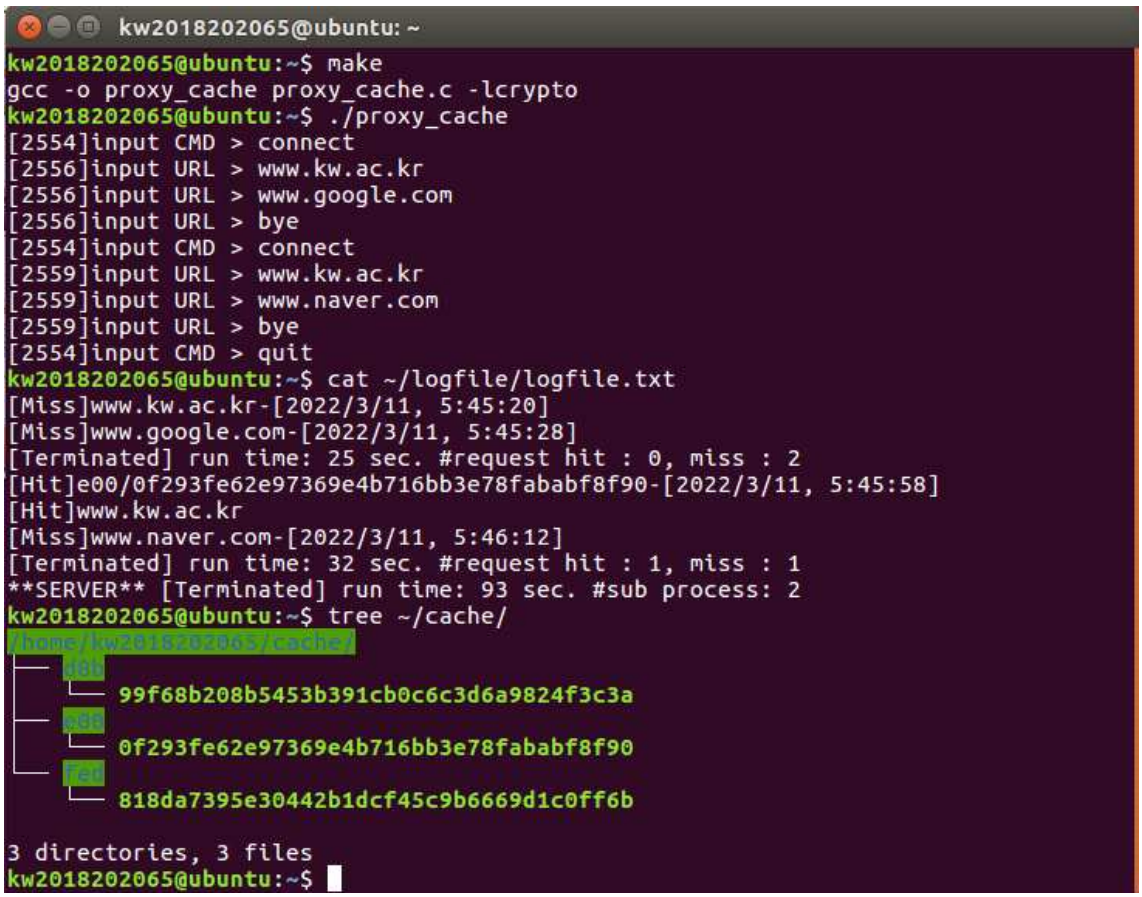
```
int main(void)
{
        fork함수의 리턴 값을 저장할 변수 pid_t형 childPid 생성;
        status를 저장할 int형 변수 status 생성;
        입력받을 커맨드를 저장할 변수 char형 배열 commond 생성;
        프로세스 시작과 끝을 저장할 변수 time_t형 tstart, tend 생성;
        time(&tstart)메인 프로세스의 시작을 저장;
        int subcount=0서브프로세스 생성개수를 저장할 변수;
        사용하고자 하는 디렉터리 위치를 저장할 변수 char형 배열 dire생성;
        홈디렉토리를 dire에 저장;
        strcat(dire, "/cache/")프로그램 시작할 때 루트 디렉터리에 cache폴더 생성하기 위해 dire변수를 사용하여 디렉터리 위치 저장;
        umask(0000)초기 설정된 umask값을 0000으로 변경;
        mkdir(dire, S_IRWXU | S_IRWXG | S_IRWXO)dire 위치에cache폴더 생성;
        홈 디렉터리를 dire에 저장;
        strcat(dire, "/logfile/")프로그램 시작할 때 루트 디렉터리에 logfile폴더 생성하기 위해 dire변수를 사용하여 디렉터리 위치 저장;
        mkdir(dire, S_IRWXU | S_IRWXG | S_IRWXO)dire 위치에 logfile폴더 생성;
        strcat(dire, "logfile.txt")프로그램 시작할 때 logfile.txt 생성하기 위해 dire변수를 사용하여 디렉터리 위치 저장;
        int fd1;
        fd1 = open(dire, O_WRONLY | O_CREAT | O_EXCL, 0777)logfile.txt 생성;
        close(fd1);
        while (1)
        {
                printf("[%ld]input CMD > ", (long)getpid());
                scanf("%s", commond);
                if (command에 quit 코멘트가 들어가면 실행 종료.)
                {
                        // 아래는 메인 프로세스(메인함수) 종료 문을 형식에 맞게 logfile에 입력하는 과정
                        time(&tend);
                        int result = (int)(tend - tstart);
                        char dire4[100];
                        getHomeDir(dire4);
                        strcat(dire4, "/logfile/");
                        strcat(dire4, "logfile.txt");
                        int fd6;
                        fd6 = open(dire4, O_WRONLY | O_APPEND, 0777);
```

```
                write(fd6, "**SERVER** ", strlen("**SERVER** "));
                write(fd6, "[Terminated] ", strlen("[Terminated] "));
                write(fd6, "run time: ", strlen("run time: "));
                char st4[100];
                sprintf(st4, "%d", result);
                write(fd6, st4, strlen(st4));
                write(fd6, " sec. ", strlen(" sec. "));
                write(fd6, "#sub process: ", strlen("#sub process: "));
                sprintf(st4, "%d", subcount);
                write(fd6, st4, strlen(st4));
                write(fd6, "₩n", strlen("₩n"));
                close(fd6);
                return 0;

        }
        if (command에 connect 코멘트가 들어가면 서브프로세스 생성 및 실행.)
        {
                subcount++서브프로세스 생성 개수를 기록하는 변수를 1증가;
                childPid = fork()child프로세스 생성;
                if (childPid가 0보다 크면 부모 프로세스)
                 {
                        // 부모 프로세스
                        pid_t waitPid;
                        waitPid = wait(&status)child프로세스가 끝날 때까지 기
                        다린다.;

                }
                else if (childPid가 0이면 자식 프로세스) {
                        // 자식 프로세스일 경우 자식코드
                        sub_process()자식 프로세스에서 sub_process함수실행;
                        exit(0)자식 프로세스 종료;
                }
        }
    }
}
```

-결과 화면




connect가 두 번으로 두 개의 서브 프로세스를 생성했고
miss가 발생한 URL은 www.kw.ac.kr과 www.google.com 마지막으로 www.naver.com으로 3개이며 www.kw.ac.kr을 중복 입력하여 Hit발생 1개가 잘 발생함을 알 수 있다. 이에 따라 cache디렉터리에 3개의 디렉터리와 파일이 생성됨을 알 수 있다. 또한 서브 프로세스를 종료할 때의 종료문과 메인 프로세스를 종료할 때의 종료문도 잘 출력됨을 알 수 있다.

- 고찰

이번 과제를 진행하면서 proxy 서버의 다중 사용자에 대한 캐시 처리에 대해 심도 깊게 이해 해볼 수 있었다. 1-2까지 구현한 main함수는 proxy 서버가 다중 사용자에 대한 입력을 받을 시 내부적으로 캐시를 만드는 과정을 구현하는 것의 초석이었고 이를 하위 모듈화 시키고 새로운 메인프로세스를 위해 main함수를 따로 설계함에 있어 프로그램이 점점 더 고도화된 작업을 함을 알 수 있었다.