시스템프로그래밍
Proxy #1-3

담당교수 : 최상호 교수님(목4)

2021202003 강준우

Proxy #1-3

Introduction

이번 과제에서는 기존의 Proxy#1-2를 바탕으로 프로세스를 복제 및 재사용 하는 등의 구현을 해봅니다. 과제에서는 fork()를 통해서 프로세스의 상태를 확인하고, 이후 조건에 따른 Proxy#1-2기능들을 수행합니다. 해당 프로세스는 connect와 quit을 통해 관리하며 이후 모든 내용을 logfile을 통해서 관리 기록합니다.

결과화면

```
kw2021202003@ubuntu:~/work/Proxy1-3_B_2021202003_강준우$ ./proxy_cache
[13635]input CMD> connect
[13638]input URL> www.kw.ac.kr
[13638]input URL> www.google.com
[13638]input URL> bye
[13635]input CMD> connect
[13639]input URL> www.kw.ac.kr
[13639]input URL> www.naver.com
[13639]input URL> bye
[13635]input CMD> quit
kw2021202003@ubuntu:~/work/Proxy1-3_B_2021202003_강준우$
```

connect를 통해서 2번의 proxy#2 프로세스를 수행하였으며, 각 수행을 connect 와 quit을 통해서 컨트롤하는 모습입니다.

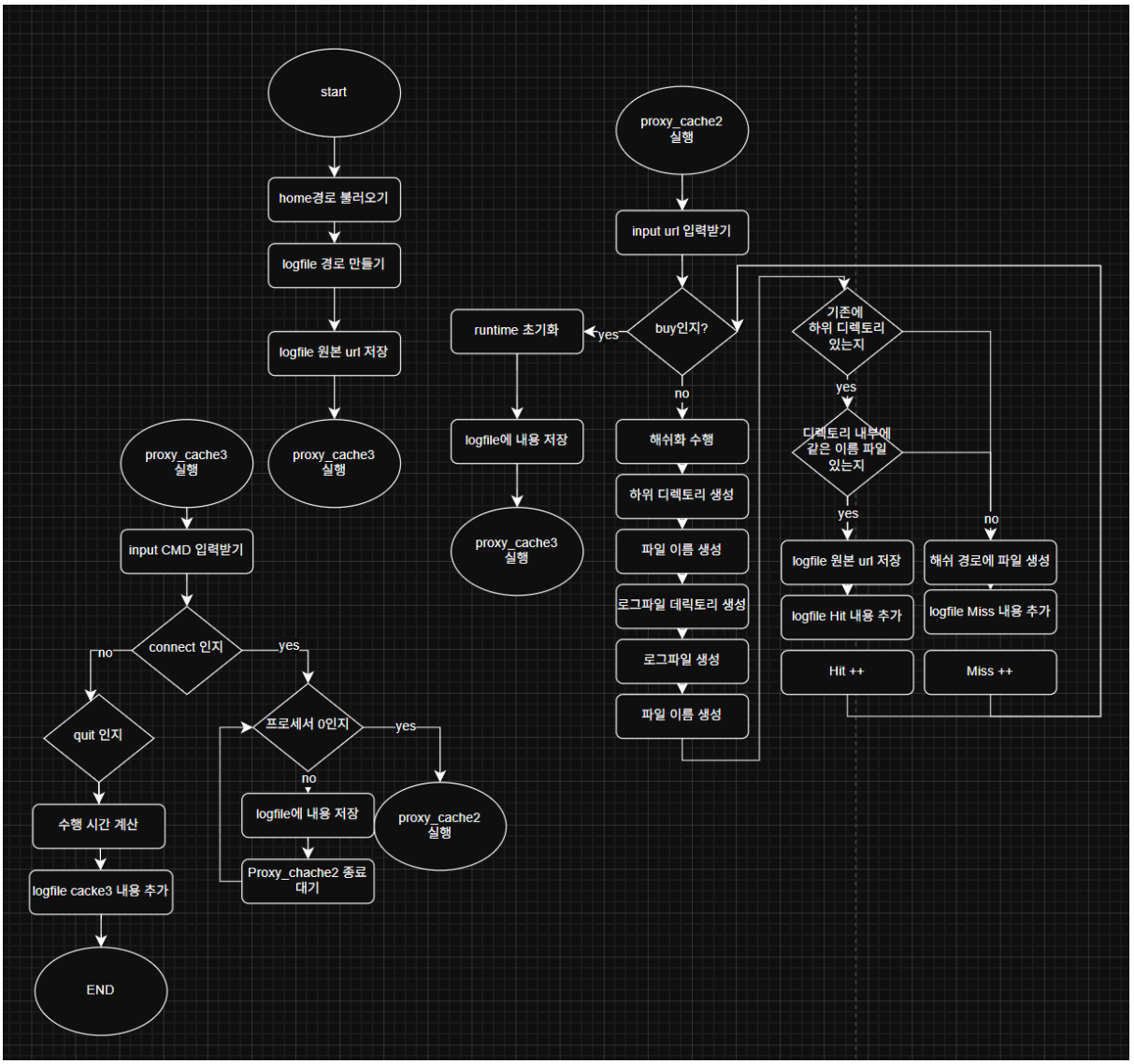
kw , google , naver 통해서 디렉토리와 파일 생성된 모습입니다.

```
kw2021202003@ubuntu:~/cache$ tree
.
├── d8b
│   └── 99f68b208b5453b391cb0c6c3d6a9824f3c3a
├── e00
│   └── 0f293fe62e97369e4b716bb3e78fababf8f90
└── fed
    └── 818da7395e30442b1dcf45c9b6669d1c0ff6b
```

logfile.txt를 통해 확인한 결과 kw.ac.kr HIT를 확인하였고, 이외에는 miss를 확인할 수 있습니다. 또한 프로그램은 각각 내부 실행시간 hit miss의 수를 반환하고 quit 명령어에 따라서 총 수행시간과 sub process의 수행 수를 반환합니다.

```
1 [Miss]www.kw.ac.kr-[2025/04/11, 10:28:48]
2 [Miss]www.google.com-[2025/04/11, 10:28:55]
3 [Terminated] run time: 12 sec. #request hit : 0, miss : 2
4 [Hit]e00/0f293fe62e97369e4b716bb3e78fababf8f90-[2025/04/11, 10:29:06]
5 [Hit]www.kw.ac.kr
6 [Miss]www.naver.com-[2025/04/11, 10:29:10]
7 [Terminated] run time: 11 sec. #request hit : 1, miss : 1
8 **SERVER** [Terminated] run time: 37 sec. #sub process: 2
```

Algorithm – Flow Chart



Pseudo Code

main

- 프로그램 시작 시간 저장
- 사용자의 홈 디렉토리 경로를 얻음
- ~/logfile 디렉토리 생성
- ~/logfile/logfile.txt 파일 경로 설정
- proxy_cache3 호출

proxy_chache2

- 시작 시간 저장
- 홈 디렉토리 경로 획득

- 무한 루프:
  - 사용자로부터 URL 입력받음

- 입력이 "bye"면:
  - 종료 시간과 실행 시간 계산
  - 로그파일에 종료 정보 및 hit/miss 카운트 기록
  - 루프 종료

- URL을 SHA1 해시로 변환
- ~/cache 디렉토리 생성
- 해시값 앞 3자리를 서브 디렉토리 이름으로 사용하여 디렉토리 생성
- 해당 경로 내에 캐시 파일 경로 설정

- 디렉토리 열어서 이미 같은 이름의 캐시 파일이 있는지 확인
- 결과에 따라:

  - 파일이 있으면 (Hit):
    - 현재 시간 저장
    - 로그파일에 Hit 정보 기록
    - hit 카운트 증가

  - 파일이 없으면 (Miss):
    - 현재 시간 저장
    - 로그파일에 Miss 정보 기록
    - miss 카운트 증가
    - 캐시 파일 생성하고 URL 내용 저장

proxy_cacke3
- 무한 루프:
  - 사용자로부터 명령어 입력받음

  - 만약 명령어가 "connect"라면:
    - 자식 프로세스를 fork()
    - 자식이면:
      - 함수proxy_cache2실행 후 종료
    - 부모면:
      - 자식 수 1 증가
      - 자식 종료 대기 (wait)

  - 만약 명령어가 "quit"라면:
    - 서버 실행 시간을 계산
    - 로그파일에 서버 종료 시점 기록
    - 루프 종료

고찰

이번 과제에서는 fork를 통해서 프로세를 복제하는 과정을 수행해보았습니다. 해당 과제를 풀기전까지 그저 함수를 따로 정의해두고 호출하는것과 fork의 차이를 이해하지 못했습니다. 이후 알게도니 사실은 fork는 새로운 프로세스를 생성하는것이고 이에따라 함수와 다르게 새로운 PID가 생긴다는 점이었습니다. 또한 메모리가 왼전히 분리되어있는 fork는 부모와 자식 관계가 있으며, 이에따라 exit과 wait을 구현하는 과정 또한 함수와 차이점을 보임을 알았습니다.

그리고 가장 큰 차이는 포크 함수를 통해서 부모와 자식이 나뉘었지만 이후 코드를 동일하게 수행한다는 점이었습니다. 반대로 함수는 호출된 시점에 호출된 내용만 수행됩니다. (부모 수행 x + 부모라는 개념 또한 없음)

wait이 자식 프로세서를 기다리면서 또한 자식노드를 수거하여 좀비 프로세서를 방지한다는 사실을 알았습니다.

Reference

2025-1_SPLab_proxy_Assginment1-3