시스템 프로그래밍 (월요일)

최상호 교수님

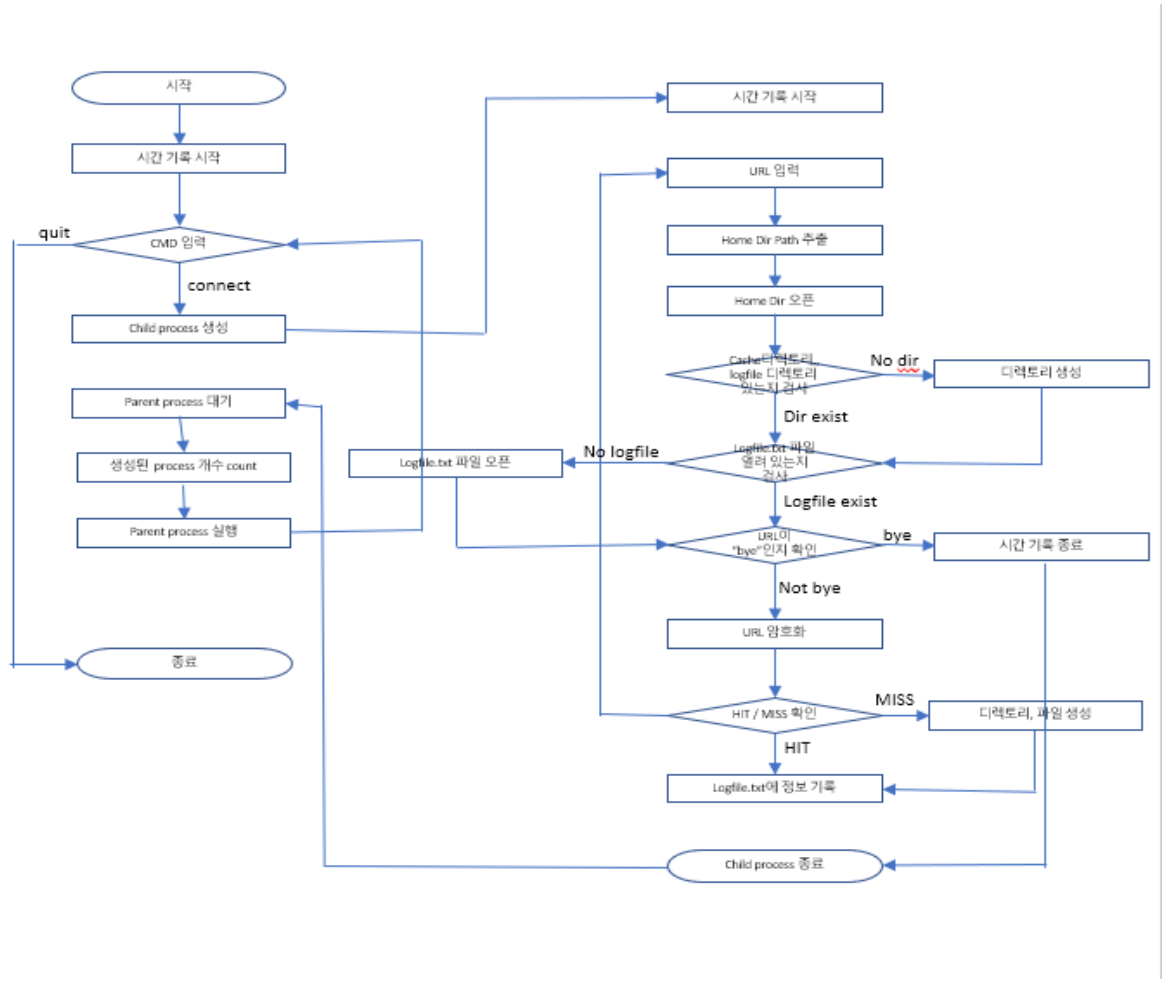
Proxy 1-3

컴퓨터정보공학부 2018202076 이연걸

Introduction:

이번 과제는 Main Process와 Sub Process를 분리하여 Main Process에서는 프로그램의 실행과 종료를, Sub Process에서는 메인 로직을 담당한다. 사용자의 입력을 통해 분리하며 connect 입력 시 Sub Process 생성이며 quit 입력은 프로그램의 종료이다. Sub Process의 동작은 Proxy1-2에서 설명한 것과 동일하게 동작한다. Main Process(Parent process)에서 fork()를 통해 Sub Process(child process)를 생성하며 Parent process는 child process의 동작이 끝날 때까지 wait()을 통해 실행대기 상태이다. 이런 흐름 때문에 분리가 어렵지는 않지만 parent process와 child process의 관계를 확실히 알지 못하면 여러 에러가 발생한다. 이 둘은 open files, root directory 등을 공유하지만 fork() 함수의 결과값, child process에서의 리소스 활용은 공유하지 않는다. 이런 내용들을 기억하고 과제를 진행한다.

Flow Chart:



Pseudo code:

시작

필요 변수들 선언 및 초기화

Home Directory Path 추출

시간 기록 시작

while 참

CMD 입력

if CMD == quit then

시간 기록 종료

로그파일 작성

프로그램 종료

else if CMD == connect then

child process 생성

if child process 생성 실패 then

프로그램 비정상 종료

else if child process 생성 성공 then

child process 동시 실행

URL 입력

Home Directory Path 추출

while HomeDirectory에 하위 폴더, 파일 전부 확인

캐시 파일 유무 확인

Endwhile

If cache 디렉토리 존재 X then

cache 디렉토리 생성

Endif

If logfile 디렉토리 존재 X then

Logfile 디렉토리, 텍스트 파일 생성

Endif

If 프로그램의 첫번째 루프 then

Logfile.txt파일 오픈

Endif

If URL 이 "bye"인지 확인 then

child process 종료

Endif

URL 암호화

Home Directory 오픈

암호화된 URL로 디렉토리 이름과 파일이름 설정

If 이름이 같은 디렉토리가 존재하지 않는다 then

로그 파일에 MISS를 기록한다.

디렉토리와 파일 생성

else

for 이름이 같은 디렉토리 내의 파일을 전부 확인 do

endfor

if 이름이 같은 파일 존재 then

로그 파일에 HIT를 기록한다.

else

로그파일에 MISS를 기록한다.

디렉토리와 파일 생성

Endif

Home Directory 닫음

endWhile

else

실행된 sub process 갯수 추가

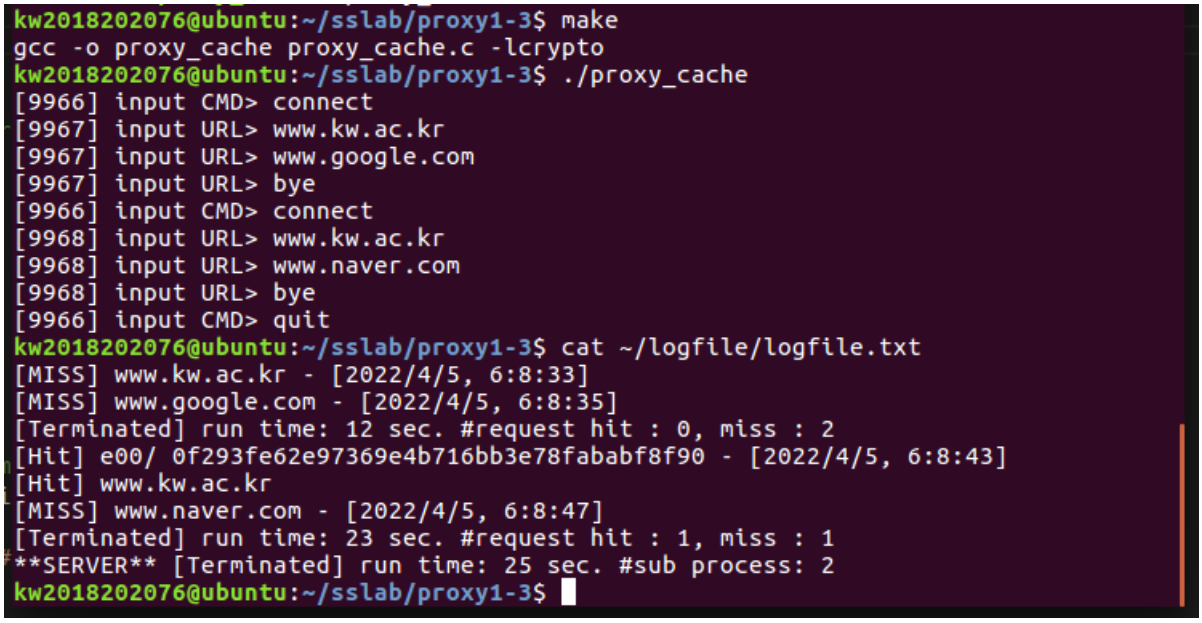
child process 종료까지 parent process 실행 대기

Endif

endWhile

종료

결과화면:



고찰:

이번 과제는 Proxy1-2를 업그레이드하는 방식이었다. Proxy1-2에서는 없었지만 이번 과제에서는 필요한 기능을 새롭게 추가했다. 우선 logfile.txt에 프로그램 종료시까지의 모든 log가 기록되어야 하므로 fopen의 모드를 w가 아닌 a로 바꾸고 덧붙이기(append)형식으로 열었다. w(쓰기 형식)로 열면 해당 파일의 내용이 모두 지워지므로 다수의 Sub process가 존재할 수 있는 이번 과제에는 적합하지 않았다.

그리고 Main process와 Sub process 모두 logfile에 기록이 가능해야 하고, 시간 또한 따로 기록되어야 하므로 Home Directory Path를 가져오는 함수를 Main process로 이동시켰고, 동시에 네개의 time_t 변수를 만들어 시간 기록을 다르게 하였다.

과제를 진행하면 fork()함수를 사용했을 때 변수의 흐름이 중요하단 것을 알게 된다. fork()함수는 parent process에서 child process를 만든다. 이때 2개의 return 값이 발생하는데 child process에는 return 값이 0이며 parent process에는 child process의 id가 들어간다. 이러한 특징과 if - else if - else 구문을 사용해 쉽게 child와 parent process를 분리할 수 있었다. 여기서 에러가 한가지 있었다. parent process에서 선언한 변수를 child process에서 수정하면 parent process에 적용이 되지

않는 것이었다. parent process는 child process와 uid, gid, pwd, env 등을 공유하지만 child process에서의 리소스 활용은 공유하지 않는다. 때문에 sub_prcess의 총 갯수를 구하기 위해선 child process가 종료되고 wait()를 사용해 대기중인 parent process가 시작할 때 ++연산자를 사용해야 한다.

처음에는 parent process와 child process의 running time을 확실히 구분하기 위해 time_t와 clock_t를 같이 사용했지만 오히려 가독성이 떨어졌고 전체 코드의 통일성을 해쳤기 때문에 time_t로 running time을 측정하였다.

Reference:

fopen 관련: https://modoocode.com/58