시스템프로그래밍

과 제 명 : Proxy 1-3

교 수 님 : 최상호 교수님

강의시간 : 월요일

학 번 : 2019202100

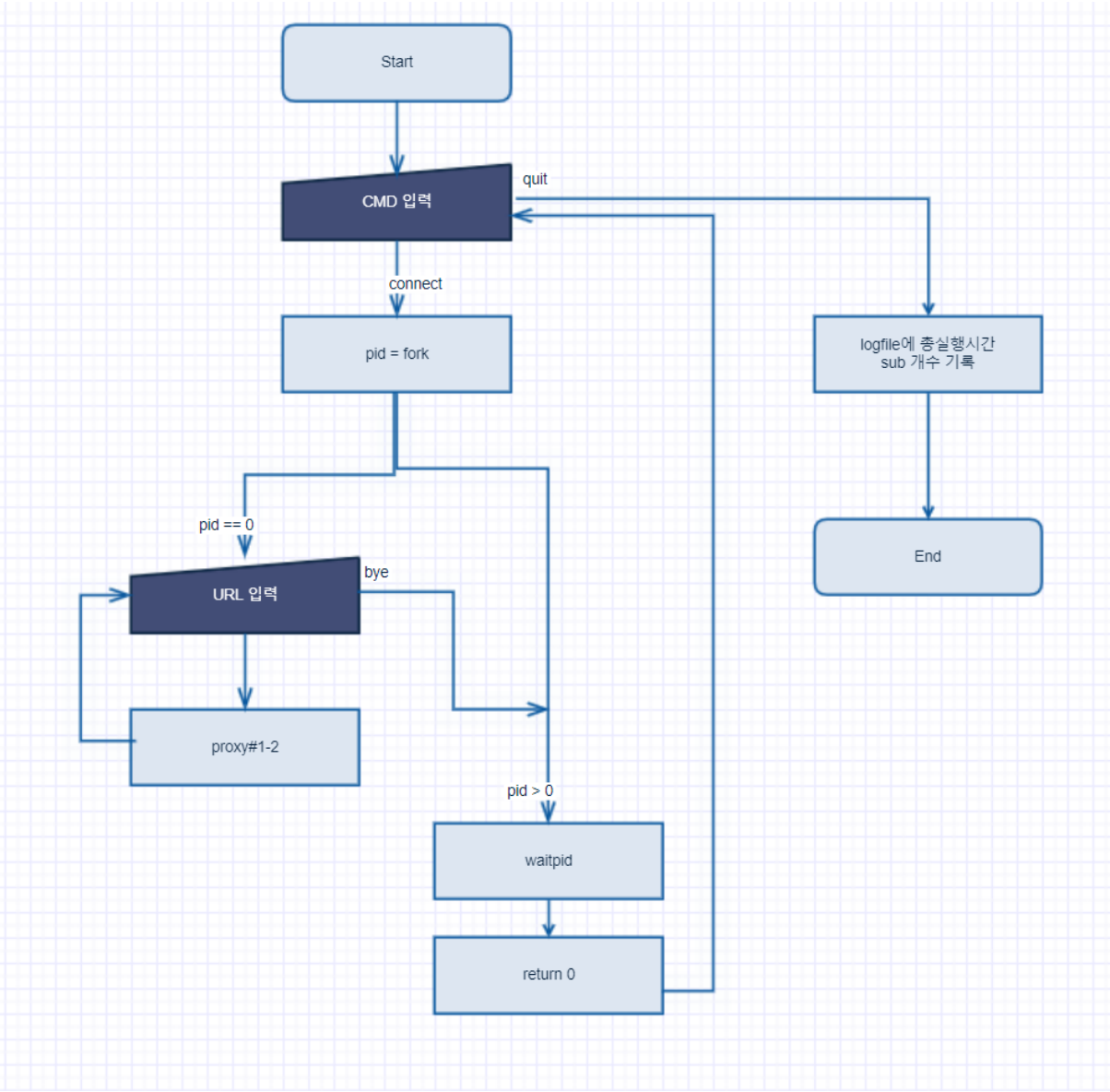
이 름 : 하주영

Introduction

새로운 프로세스를 만드는 fork 함수를 바탕으로 CMD 입력창을 띄우고 connect를 입력했을 때 fork을 실행하여 sub process에서 proxy#1-2을 실행한다. Proxy#1-2에서 URL 입력받을 때 bye를 입력하면 sub process를 return한다. 다시 CMD 입력창을 띄우고 connect or quit을 입력받는데 connect 입력할 경우 sub process를 만들고 quit을 입력하면 전체 프로그램을 종료한다.

```
sslab@ubuntu:~$ ./proxy_cache
[3933]input CMD> connect
[3934]input URL> www.kw.ac.kr
[3934]input URL> www.google.com
[3934]input URL> bye
[3933]input CMD> connect
[3935]input URL> www.kw.ac.kr
[3935]input URL> www.naver.com
[3935]input URL> bye
[3933]input CMD> quit
```

Flow Chart


Start
CMD 입력
quit
connect
pid = fork
logfile에 총실행시간
sub 개수 기록
End
pid == 0
URL 입력
bye
proxy#1-2
pid > 0
waitpid
return 0

Pseudo Code

```
int sub_cnt = 0;

while(CMD 입력 != "quit")

{

        if(CMD == "connect")

                sub_cnt++;

                pid = fork();

                if(pid == 0) /*sub process*/

                        while(URL 입력 != "bye")

                                Excute proxy#1-2

                        if(bye) return 0;


                else if (pid > 0)     /*parent process*/

                        waitpid(pid, &status, 0);


}


write run time , sub_cnt to "logfile.txt"
```

Result Screen

->proxy_cache 실행

connect 명령어 입력하면 자식프로세스를 생성하고 url을 입력할 수 있다. 만약 bye를 입력하면 자식프로세스가 종료되고 다시 connect or quit을 입력할 수 있다. Quit을 입력하면 프로그램이 종료된다.

```
kw2019202100@ubuntu:~/1-3$ ./proxy_cache
(3029)input CMD > connect
(3030)input URL > www.google.com
(3030)input URL > www.naver.com
(3030)input URL > bye
(3029)input CMD > connect
(3031)input URL > www.naver.com
(3031)input URL > bye
(3029)input CMD > quit
```

->Logfile.txt 화면에 출력

Connect 입력하면 자식프로세스를 생성하고 bye를 입력하면 자식프로세스가 종료된다.

이 때 자식프로세스의 실행시간과 hit/miss 횟수를 출력한다.

마지막으로 quit을 입력하면 총 실행시간과 호출한 자식프로세스의 개수를 출력한다.

```
kw2019202100@ubuntu:~/1-3$ cat ~/logfile/logfile.txt
[Miss]www.google.com-[2022/04/09, 00:02:42]
[Miss]www.naver.com-[2022/04/09, 00:02:44]
[Terminated] run time: 6 sec. #request hit : 0, miss : 2
[Hit]fed/818da7395e30442b1dcf45c9b6669d1c0ff6b-[2022/04/09, 00:02:48]
[Hit]www.naver.com
[Terminated] run time: 3 sec. #request hit : 1, miss : 0
**SERVER** [Terminated] run time : 14 sec. #sub process : 2
```

->Cache 디렉토리와 파일명을 트리구조로 출력

```
kw2019202100@ubuntu:~/1-3$ tree ~/cache/
/home/kw2019202100/cache/
├── d8b
│   └── 99f68b208b5453b391cb0c6c3d6a9824f3c3a
└── fed
    └── 818da7395e30442b1dcf45c9b6669d1c0ff6b

2 directories, 2 files
```

Consideration

과제 진행할 때 문제가 2가지가 있었다.

1. strncpy(d_name, hashed_url, 3)

   proxy#1-2 에 해시 url의 앞 3글자를 따서 해시 디렉토리를 만들 때 strncpy(d_name, hashed_url, 3)을 사용하였다.

   하지만 이번에 자식프로세스를 만들고 그 안에서 strncpy 함수를 호출했을 때는 문제가 발생했다.3글자 이외에도 이상한 문자가 포함되어 디렉토리가 만들어졌다.

   그 이유는 바로 strncpy의 안정성 때문이다.

   윈도우와는 다르게 리눅스에서는 'n'을 가져올 때 'n'개의 글자만 복사하고 NULL 처리를 하지 않을 때도 있으므로 마지막 글자에 NULL을 직접 입력해 주어야 한다.

   ```
   strncpy(d_name, hashed_url, 3);
   d_name[3] = '\0';
   ```

   마지막 글자에 '0'을 추가하는 방식으로 이상한 문자가 포함되는 것을 방지할 수 있다.

2. waitpid(pid, &status, 0);

   Waitpid는 부모프로세스가 자식프로세스가 종료될 때까지 기다린 후 자식프로세스가 종료되면 자식프로세스의 상태를 저장하는 함수이다. 자식프로세스가 좀비 프로세스로 계속 남아있지 않기 위해서 무조건 사용해야 하는 함수이고, wait 함수와 달리 특정 pid 값을 가진 자식프로세스의 종료를 기다린다.

   ```
   if( ( pid = fork() ) >0 ) /*parent process */
   waitpid(pid, &status, 0);
   ```

   이때 pid는 기다리는 자식프로세스의 pid이다.

Conclusion

새로운 프로세스를 만드는 함수는 fork이다. Fork의 리턴 값은 0과 자식프로세스의 pid이다. 만약 (pid == 0) 이면 자식 프로세스의 실행 부분이며 (pid > 0) 이면 부모프로세스의 실행 부분이다.

부모 프로세스가 자식 프로세스의 종료 상태를 받아 줄 수 없는 상태일 때

(예를 들어서, 부모가 먼저 종료됐을 때 or 부모가 아직 실행중인 상태) 좀비 프로세스가 생기며

따라서 부모프로세스의 실행부분에 waitpid함수를 호출하여 자식 프로세스의 종료를 대기해야 한다.

Reference

Fgets 함수 특징

https://inhwascope.tistory.com/133

strncpy 사용 시 주의 사항(Linux .. : 네이버블로그 (naver.com)