**Small Shell**

리눅스 프로그래밍

학 번 12191659

이 름 이화연

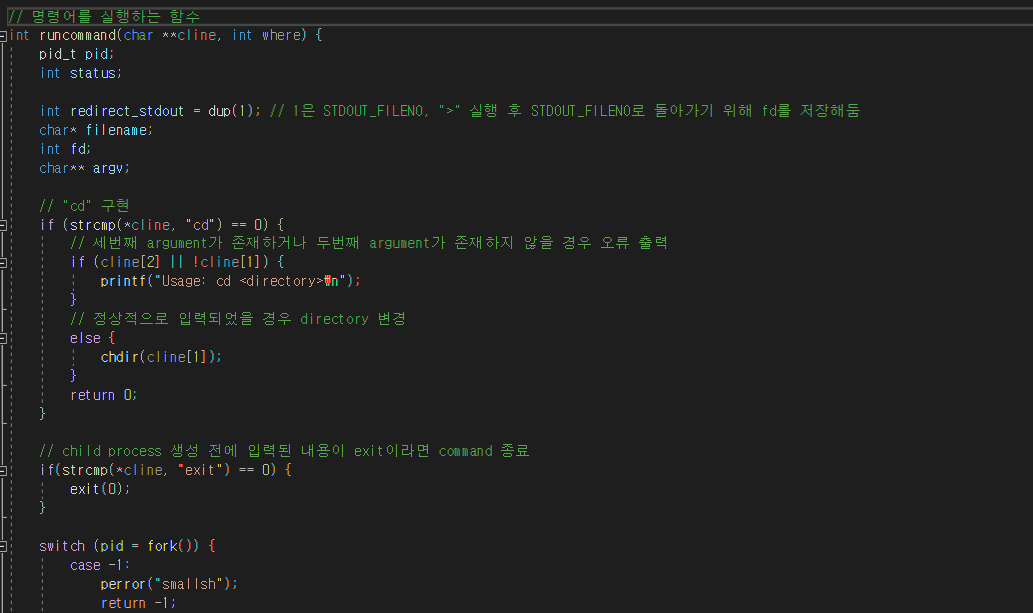
소속학과 컴퓨터공학과

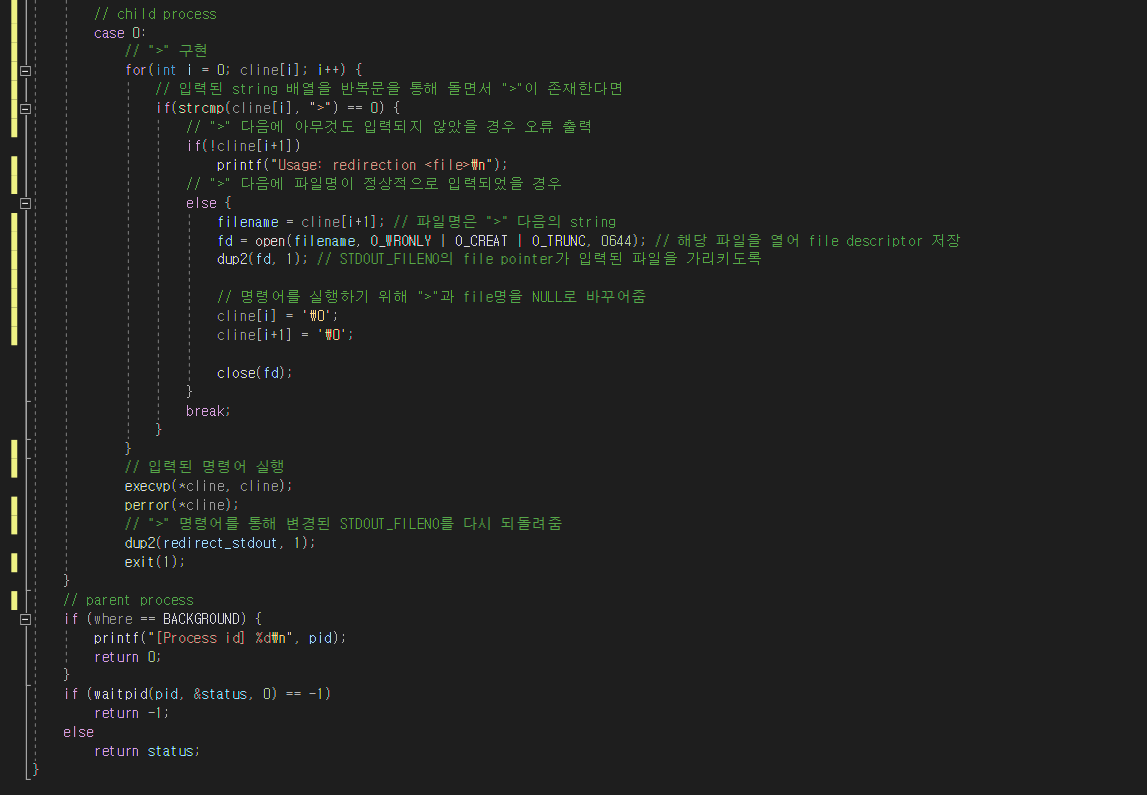
교 수 님 이어진 교수님

제출일자 2021.11.19

**1. 구현 방법**

다음 코드는 smallsh.c 내의 runcommand() 코드이다. 기능 구현을 위해 추가한 부분은 procline()과 runcommand() 함수 내에 있다. 하지만 ‘>’ , ‘;’, ‘&’가 동시에 사용될 경우 정상 동작하기 위해서 구현한 procline() 내의 코드는 문제점이 존재한다. 때문에 procline() 함수는 Discussion에서 다루도록 하겠다. 추가된 기능의 구현 방법은 코드 내의 주석으로 설명하겠다.





**2. ‘&’ 문제점**

기존 코드에서 background 실행을 위한 명령어인 ‘&’ 실행 시 문제점이 존재한다.

부모 프로세스(parent process)가 먼저 return 되어 죽어버리면, 자식 프로세스(child process)가 고아 프로세스(orphan process)가 된다. 이 때 부모 프로세스(parent process)는 pid가 1인 init으로 바뀌고 자식 프로세스(child process)가 종료되면 이는 좀비 프로세스(zombie process)가 되는 문제가 발생한다.

**3. Discussion**

다음장의 코드는 smallsh.c 내의 runcommand() 코드이다. ‘>’ , ‘;’, ‘&’가 동시에 사용될 경우 정상 동작을 위한 코드를 구현하였지만 구현에 실패하였다. 주석으로 나타난 코드가 구현하지 못한 코드이다. ‘\n’, ‘;’, ‘&’ 문자가 올 경우 명령어를 runcommand()하고 다음 token으로 넘어가므로 이들을 종료문자라고 하겠다.

‘>’는 runcommand() 내에 따로 구현하였으므로 ‘;’. &’과 동시에 사용이 가능하다. 그러므로 ‘;’과 ‘&’가 동시에 사용될 경우의 예외만 잡아주면 되는데 ls -l; ls& 와 같은 형태의 명령어는 종료문자인 ‘;’과 ‘&’가 서로 영향을 미치지 않는다. 따라서 ls -l의 runcommand() 실행 후 ls가 background에서 runcommand() 된다.

따라서 잡을 예외는 ls &; ls -l과 같이 종료문자인 ‘&’과 ‘;’가 연달아 오는 경우만 존재한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

첫 번째 주석으로 나타난 코드의 의도는 다음과 같다. 만약 ‘&’의 다음 token이 ‘;’이라면 ‘&’ 문자를 NULL로 변경하고 runcommand()를 실행하지 않고 다음 token으로 넘어간다.

두 번째 주석으로 나타난 코드의 의도는 다음과 같다. 만약 ‘;’의 전 token이 ‘&’라면 type을 BACKGROUND로 바꾸어 주고 ‘;’ 문자를 NULL로 변경하고 runcommand()를 실행한다.

만약 ls &; ls -l가 입력된다면 ‘&’, ‘;’ 모두 NULL로 변경되어 ls만 BACKGROUND에서 runcommand()되고 정상적으로 다음 ls -l 명령어가 runcommand()되기를 의도하였다. 하지만, 의도한대로 실행되지 않았고 정확한 이유도 자세히 모르겠다. 위의 코드가 아닌 다른 코드를 짜면서 ‘&’과 ‘;’가 동시에 사용될 수 있도록 노력하였지만 모두 실패하였고, 결국 이는 구현해내지 못하였다.