**유닉스 프로그래밍**

**-쉘 만들기-**

12171651 컴퓨터 공학과 오윤석

**1. 제출일**

- 2019/11/22 (3차)

**2. 요구 사항 정의**

**a.** ‘cd’ 명령이 제대로 먹히지 않는 버그를 수정

- 쉘에서 프로그램을 실행시키면 실제로 실행되는 프로세스는 child 프로세스에서 실행이 된다. cd(change dir)는 쉘의 내장 명령어임으로 우리가 따로 구현을 하여 만들어야지 실행된다. child에서 cd를 실행하여 dir을 변경하여도 parent 프로세스에서는 직접적으로 변경이 되지 않기 때문에 cd 함수를 main에서 실행을 하게 구현하였다.

**b.** ‘exit’ 명령을 구현

- exit 명령어를 사용하면 프로세스가 종료된다.

**c.** 백그라운드 실행을 구현

- 백그라운드 실행이란 현재 하고 있던 작업을 잠시 혼자서 실행시키고 사용자는 다른 작업을 할 수 있게 해준다. 일종의 멀티태스킹 개념이라고 생각하면 된다. 여기서 백그라운드의 반대적인 개념은 포그라운드이다.

**d.** SIGCHILD로 자식 프로세스 wait()시 프로세스가 온전하게 수행되도록 구현

- 시그널 함수를 통해 자식 프로세스가 좀비 프로세스가 되지 않기 위해 sigchild로 시그널을 받게 구현하게 해야된다.

**e.** SIGINT, SIGQUIT 사용시 쉘이 종료되지 않고록, Foreground 프로세스 실행 시 SIGINT를 받으면 프로세스가 끝나는 것을 구현

- 이것 또한 SIGNAL 함수를 통해 SIG\_IGN으로 해당 시그널을 무시하게 구현하면 된다.

**f.** 리다이렉션 구현

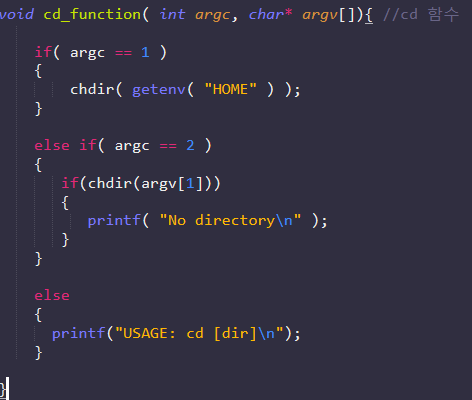
- 리다이렉션이란 컴퓨터에서 표준 스트림을 사용자가 지정한 위치로 우회할 수 있는 유닉스 쉘을 포함한 대부분의 명령어 interpreter에 일반적인 명령이다. 리다이렉션의 종류는 2가지가 있는데 1개는 방향 지정 [1], 또다른 건 방향 변경[2]이라고 한다. 리다이렉션은 open함수와 dup를 활용하여 구현할 수 있다.

**g.** 파이프 구현

-파이프는 유닉스에서 제공되는 병행성 메커니즘으로써 2개의 프로세스가 모델에 따라서 통신할 수 있게 해주는 원형 버퍼이다. 파이프는 “|” 이걸로 구현이되고, 파이프는 한 방향으로 통신을 할 수 있다. 하지만 한 방향당 1개의 파이프를 두면 두개의 파이프가 양방향 통신을 할 수 있다. 파이프 또한 dup2와 fork를 통하여 구현할 수 있다.

**3. 구현 방법 서술**

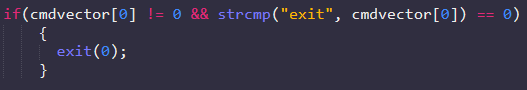
**a.**



cd 명령어를 위해 따로 cd\_function이란 함수로 정의하였다. 인자 값은 argc와 argv[]를 가진다. 우선 조건문을 보면 argc에 따라 분리되는데 우선 argc가 1이면 그냥 cd만 입력했다는 의미로 getenv함수를 통해 home으로 돌아가게 된다. 여기서 HOME은 환경 변수 목록에 자동으로 저장되어있다.

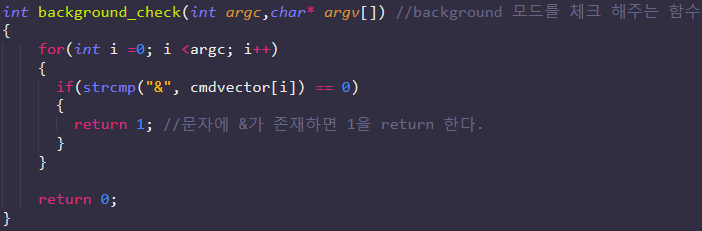
또한 argc가 2이면 error를 출력하고, 나머지 상황일때는 getcwd함수를 통해 cd후 현재 pwd를 가져온다.

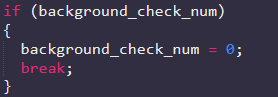
**b.**



Exit은 따로 함수를 구현하지 않고 그냥 exit을 입력하였을 때 exit(0)를 통해 프로세스를 종료하게 만들었다. 또한 아무것도 입력을 하지 않으면 cmdvector[0]에는 null 값이 들어가게 되는데 segment 오류를 방지하고자 cmdvector[0]가 NULL일때를 고려하였다.

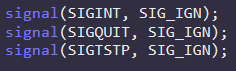
**c.**



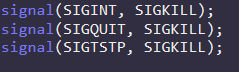


우선 입력 문자열에서 &가 존재하는지 체크를 한다. 만약에 있다면 1을 return을 하고 background\_check\_num 변수에 저장을 해둔다. 이 변수가 1이면 break를 통해 waiting을 하지 않게 해준다.

**d & e**

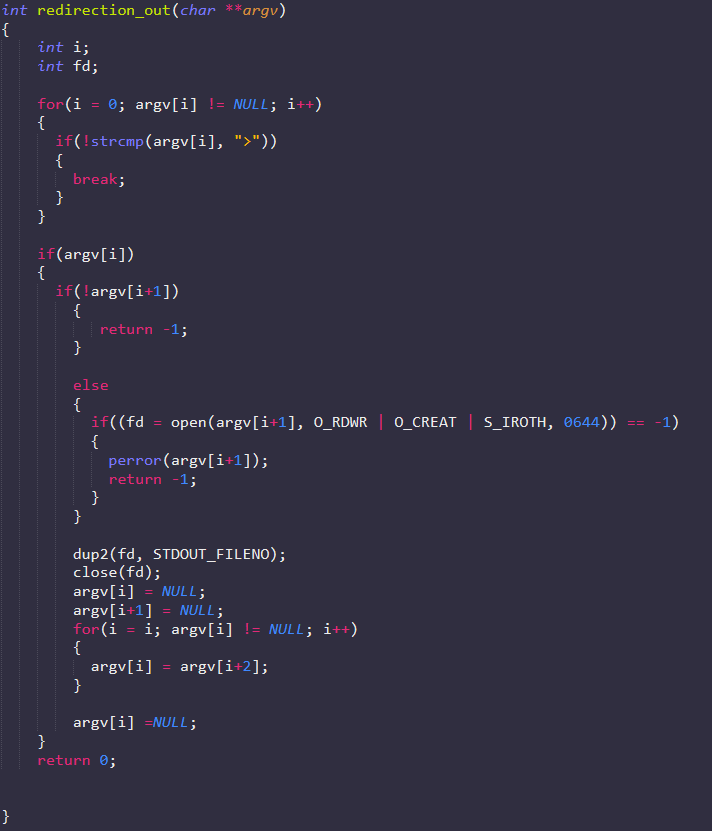


d와 e는 해당 시그널 함수를 통해 sigint, sigquit, sigtstp를 무시하게 구현을 하였다.

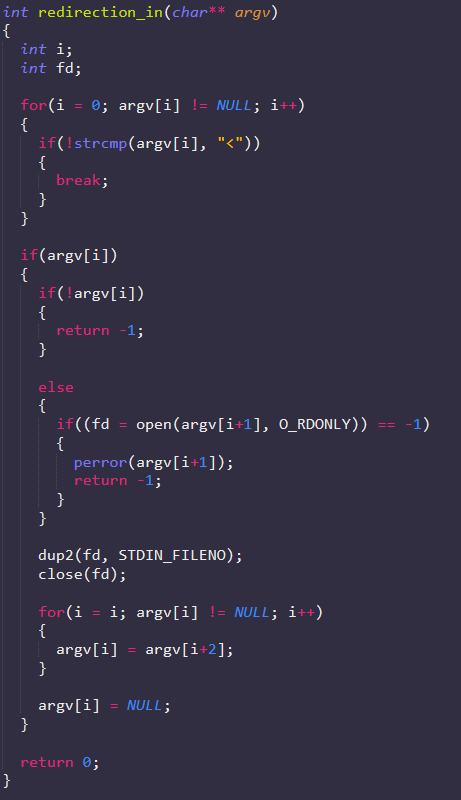


또한 프로세스가 실행 중 해당 시그널을 받게 되면 즉시 해당 프로세스를 종료되도록 구현을 하였다.

**f.**

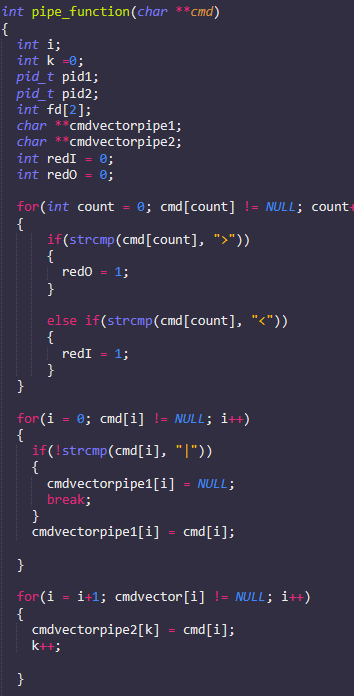


우선 argv 문자열이 들어오게 되면 이 문자열에 “>”가 존재하는지 확인을 한다. 그후 open을 한 뒤에 dup2에 있는 STDIN\_FILENO로 읽기 전용 파일에 접근하여 디스크립터를 열고 명령어를 수행한다.

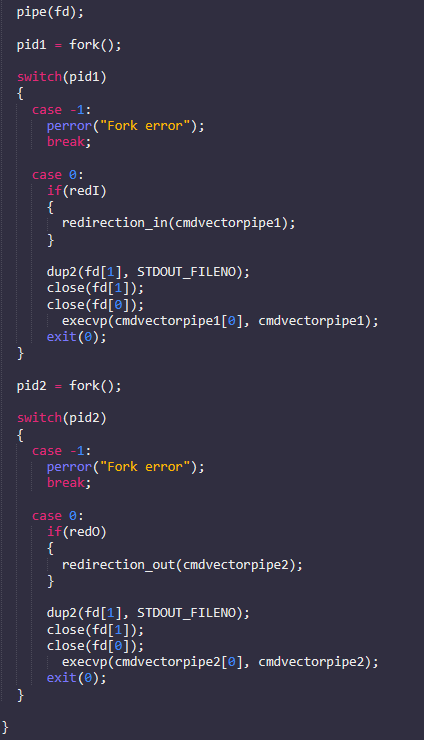


리다이렉트 in도 out과 비슷한 방식으로 구현을 하였다. 다만 다른점은 open 함수 인자 속에 RDONLY를 사용한다.

**g.**



파이프 구현의 경우에는 우선은 리다이렉션을 유무를 for문을 통해 탐색항 체크하도록 하였다. 또한 파이프 (“|”)도 검사를 하게 만든 후 파이프 이전 명령어를 cmdvectorPipe1에, 이후 명령어 들을 cmdvectorPipe2에 저장하도록 만들었다.



이후 pipe 함수를 실행하고, fork를 통해 자식 프로세스를 생성하고 dup2를 통해 파이프 명령어가 실행되도록 구현하였다.

**4. 구현 시 문제점**

그냥 cd를 입력하게 되면 무조건 home으로 이동하는 흠이 있다. 사람마다 home이 다르므로 이를 더 자세히 조사해서 수정해야 될 것 같다.

또한 exit에서는 다른 명령어 실행 후 exit을 하게 되면 바로 종료되지않고 해당 프로세스가 종료가 되야지 exit 명령어가 실행이 된다.

파이프 같은 경우 fork하는 과정에서 segmentation 오류가 난다. 구현 이론 상으로는 잘돌아가는데 컴파일 과정에서 오류가 존재한다. 시간이 더 주어졌으면 파이프도 정상적으로 구현할 수 있을 것 같다. (파이프 경우 함수는 코드에 있으나 main함수에 넣을 경우 오류가 발생하여 main에서는 주석 처리 했습니다.)