A Simple Linux Shell

2016003672

소프트웨어학부

엄세진

이 프로젝트는 간단한 리눅스 명령어 해석기를 스스로 만들어보는 과제이다.

프로젝트 요구사항

- foreground and background execution (&)
- multiple commands separated by semicolons
- history command
- shell redirection (>, >>, >|, <)
- shell pipe (ls –la | more)
- Multiple pipe (ls | grep "^d" | more)
- cd command

쉘의 동작 과정을 크게 설명하자면, 일단 쉘에서는 종료 시그널을 받기 전까지는 입력받은 명령어를 받아 처리한 후 다시 명령어를 입력받아야 한다. 그래서 while loop를 들어가서 수행한다. 그리고 사용자가 명령어를 입력하였을 때, 쉘은 그저 텍스트로만 받아들이므로, 이 텍스트를 쉘 인터프리터에서 foreground인지 background인지, 세미콜론이 있는지, history command인지, redirection인지 등등 다양한 경우로 나누어 처리를 해 주어야 한다.

첫 번째 요구사항, foreground & background는 입력받은 명령어의 끝 부분에 &가 있는지 먼저 찾는다. 모든 작업은 fork()를 통해 자식 프로세스를 생성하여 진행하게 된다. fork()를 하지 않고 직접 execve를 하게 되면, 모든 코드가 해당 프로그램으로 덮어써져 쉘 프로그램이 해당 프로그램으로 바뀐다. 그 말인 즉슨 쉘은 계속해서 실행되고 있는 상태여야 하기 때문에 fork()를 사용한다. 그러면 자식 프로세스가 생성이 되는데, 부모 프로세스는 wait(&status)를 통해 자식 프로세스

의 종료를 기다리거나 기다리지 않을 수 있다. 부모 프로세스가 자식 프로세스가 끝나기를 기다린다는 것은 자식 프로세스의 실행이 끝날 때 까지는 쉘이 다른 작업을 하지 못하고 대기한다는 것을 의미한다. 반면, 기다리지 않는 경우는 자식 프로세스가 끝나지 않더라도 부모 프로세스는 기다리지 않고 바로 다음 작업을 진행하게 된다. 다시 말해 쉘이 해당 작업이 끝날 때 까지 기다리지 않고 바로 다음 명령어 입력을 받는 상태로 넘어간다는 것이다.

정리하자면 &가 명령줄에 존재한다면 & 앞에 있는 부분은 fork를 통해 자식 프로세스를 생성하고, execve를 통해 수행한 후 부모 프로세스에서 wait()를 하지 않는 것으로, &가 없다면 wait(&status)를 통해 기다린다.

두 번째 요구사항 multiple commands seperated by semicolons에서는 명령줄에 세미콜론으로 구분 된 여러 명령어를 순차적으로 실행하라는 이야기이다. 명령줄 안에 ;가 있다면 ;로 구분된 명령어가 몇 개가 있는지 파악한 후, 배열에 각각 구분된 명령어를 넣어준다. ;가 없다면 해당 명령줄에는 하나의 command만 있는 것으로 간주하여 1개의 명령어로 간주한다. 그리고 처음에 설계해둔 while loop 안에서 명령어 입력을 받기 전 현재 남아있는 command의 개수가 0개라면 명령어 입력을 받고, 1개 이상 있다면 입력을 받지 않고 해당 커맨드를 interpret하여 실행해 준다.

세 번째 요구사항 history command는 그 동안 입력받은 명령어들을 보여주며 !num을 입력하면 해당 num번째 history command를 실행시켜준다. 초기에 history들을 저장할 배열을 미리 정의해두고, 쉘의 while문에서 명령어를 입력받으면 하나씩 저장해 쌓아둔다. 그리고 history 명령어가 들어오면, 차례차례 histroy 배열에 들어있는 명령어를 순서대로 출력해준다.

네 번째 요구사항 shell redirection은 프로그램의 실행 결과를 파일로 저장하거나, 파일을 프로그램의 input으로 넣는 기능이다. 먼저, 명령어를 입력 받고 >>도 >가 있는지 확인하면 존재한다고 처리되므로 >>와 일치하는 것이 있는지 먼저 파악한 후, 있다면 일단 파일을 open해주는데 이 때 있는 파일에는 붙여써야 하므로 O_WRONLY | O_CREAT | O_APPEND 모드로 열어주어야 한다. 그리고 오픈한 fd를 dup2(fd,1)을 통해 표준 출력으로 나갈 출력을 fd로 복사해준다. 그리고 execvp를 이용해 해당 프로그램을 실행시켜주면, 원래 있는 파일이라면 그 파일의 끝에 붙여서 출력내용이 들어가고, 없다면 새로 생성이 될 것이다. >의 경우에는 없는 파일이면 생성하고, 있는 파일이면 그대로 덮어쓰면 되므로 O_WRONLY | O_CREAT | O_TRUNC의 옵션으로 file을 open 해준다. < 은 파일을 input으로 받아야 되므로, 이전과 동일하게 open 해주고, dup 할때만 dup2(fd,0)으로 해 준다.

다섯 번째 요구사항 shell pipe는 다른 작업들과는 다르게 fork 후 그 자식 프로세스에서 한번 더 fork를 해서 자식과 자식의 자식프로세스간에 통신을 하게 해야 한다. 우선 | 기호가 있는지 부터 파악을 한 후 있다면 fd[2]를 선언을 미리 해 두고, pipe(fd)를 통해 파이프를 생성한다. 그리고 fork를 하고, 자식에서는 dup2(fd[1],1);close(fd[0]);close(fd[1]);을 해준 후 execvp를 통해 | 앞부분의 명령어를 실행해준다. 그리고 부모에서는 dup2(fd[0],0);close(fd[0]);close(fd[1]);을 해준 후 execvp를 통해 | 뒷 부분의 명령어를 실행해준다. 그리고 쉘은 wait을 통해 wait을 해 준다.

마지막 요구사항인 cd는 directory를 변경을 해 주면 되는데, chdir로 cd 뒤의 경로를 넣어주고 다시 while loop의 처음으로 돌아가주면 된다.

출력 예시1. >

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ echo hi > hi
/mnt/c/Users/sejin/Desktop/smsh/sh$ cat hi
hi
```

출력 예시2. >|

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ ls .. -l >| a
/mnt/c/Users/sejin/Desktop/smsh/sh$ cat a
total 68
-rwxrwxrwx 1 sejin sejin 17321 Jun 19 18:39 A Simple Linux Shell.docx
-rwxrwxrwx 1 sejin sejin   105 Jun 19 10:55 Makefile
drwxrwxrwx 1 sejin sejin  4096 Jun 19 19:25 sh
-rwxrwxrwx 1 sejin sejin 21664 Jun 19 19:23 smsh
-rwxrwxrwx 1 sejin sejin  6363 Jun 19 19:23 smsh.c
-rwxrwxrwx 1 sejin sejin  8744 Apr 15 00:07 test
-rwxrwxrwx 1 sejin sejin  1654 Jun 19 18:28 test.c
-rwxrwxrwx 1 sejin sejin   162 Jun 19 16:15 ~$Simple Linux Shell.docx
```

출력 예시3. <

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ wc < hi
1 1 3
```

출력 예시4. >>

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ echo hello >> hi
>> 실행
/mnt/c/Users/sejin/Desktop/smsh/sh$ cat hi
hi
hello
```

출력 예시 5. History

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ history
    1 ls
    2 mkdir test
    3 mkdir sh
    4 cd sh
    5 cd ~
    6 echo $HOME
    7 ls
    8 ls > a
    9 ls > a
   10 rm a
   11 ls > a
   12 cat a
   13 ls ..
   14 ls .. -l >| a
   15 cat a
   16 echo hi > hi
   17 cat hi
   18 wc < hi
   19 echo hello >> hi
   20 cat hi
   21 history
```

출력 예시 6. History 사용 !

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ history
    1 ls
    2 mkdir test
    3 mkdir sh
    4 cd sh
    5 cd ~
    6 echo $HOME
    7 ls
    8 ls > a
    9 ls > a
   10 rm a
   11 ls > a
   12 cat a
   13 ls ..
   14 ls .. -l >| a
   15 cat a
   16 echo hi > hi
   17 cat hi
   18 wc < hi
   19 echo hello >> hi
   20 cat hi
   21 history
/mnt/c/Users/sejin/Desktop/smsh/sh$ !1
a  hi
```

출력 예시 7. 단일 파이프

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ ls -al | wc
      5      38     190
```

출력 예시8. Change directory

```
/mnt/c/Users/sejin/Desktop/smsh/sh$ cd ..
/mnt/c/Users/sejin/Desktop/smsh$ cd sh
/mnt/c/Users/sejin/Desktop/smsh/sh$
```

출력 예시9. Background

```
/mnt/c/Users/sejin/desktop/smsh$ ls &
/mnt/c/Users/sejin/desktop/smsh$ 'A Simple Linux Shell.docx'   Makefile  a   sh   smsh   smsh.c   test   test.c  '~$Sim
ple Linux Shell.docx'
```