Homework 2: shell 구현하기

제출마감 : 2018.9.27 11:00 am

제출처 : 손인엽 조교(son901217@naver.com)

제출물 : 소스파일을 이메일로 제출

이번 과제를 통해 shell 의 몇몇 기능들을 구현함으로써 Unix system call interface 와 shell 에 더욱 익숙해질 것이다. Unix API 를 지원하는 OS 에서 해당 과제를 수행할 수 있다. 구현한 shell code file 을 제출하라.

piazza 에서 sh.c 소스코드를 다운 받고 확인하라. sh.c 코드는 shell 명령어를 parsing 하는 parser 코드와 parsing 된 명령어를 실행하는 코드로 나눌 수 있다. sh.c 코드에 parser 부분은 모두 구현 되어 있으며, parsing 된 명령어를 실행하는 코드를 구현하는 것이 이번 과제이다. Parser 는 아래와 같이 간단한 명령어들을 인식할 수 있다.

```
ls > y
cat < y | sort | uniq | wc > y1
cat y1
rm y1
ls |  sort | uniq | wc
rm y
```

위의 명령어들을 t.sh 파일에 복사 하라.

다음 명령어로 다운받은 sh.c 파일을 컴파일 하라.

```
$ gcc sh.c
```

그러면 실행가능한 a.out 파일이 생길 것이다.

```
$ ./a.out < t.sh
```

이후 위의 명령어를 실행하면 에러가 발생할 것이다. sysprog shell 에는 아직 몇 가지 기능을 구현되지 않았기 때문이다. 이 과제를 마칠 때 즈음엔 여러분은 이것을 실행할 수 있을 것이다.

Executing simple commands

다음의 간단한 명령어를 실행해 보라.

```
sysprog$ ls
```

parser 가 해당 명령어를 execmd 형태로 parsing 해준다. 위의 명령어를 수행하기 위해서는 runcmd 함수의 ‘ ’ case 를 코드로 작성해야 한다.

Man page 에서 exec 시스템 콜에 대한 유익한 정보얻을 수 있다. man 3 exec 를 터미널에 입력하고 execv 함수에 대해 읽어보라. 그리고 exec 가 실패했을 때 에러 메세지를 출력하도록 하라.

여러분이 구현한 프로그램을 테스트 하기위해 컴파일 하고 a.out 을 실행하라.

```
sysprog$
```

위와 같은 프롬프트가 출력되고, 입력을 기다릴 것이다. Shell 에 다음의 명령어를 입력해보라.

```
sysprog$ ls
```

shell 은 아마 에러 메시지를 출력할 것이다. (ls 라는 프로그램이 여러분의 작업 디렉터리에 존재하거나 PATH 를 검색하는 버전의 exec를 사용하지 않는다면)

Shell 에 다음 명령어를 입력해보라.

```
sysprog$ /bin/ls
```

이 명령어는 작업 디렉터리에 존재하는 파일들의 이름을 출력하는 /bin/ls 프로그램을 실행한다.

Ctrl-d 를 눌러 sysprog shell 을 종료하고 여러분의 컴퓨터 shell 로 돌아갈 수 있다.

만약 작업 디렉터리에 프로그램이 없을 때 디렉터리를 /bin 으로 변경하고 싶지 않다면 PATH 변수를 지원하도록 구현하여 “/bin” 을 입력하지 않게 할 수 있다.

I/O redirection

I/O redirection 코드를 구현 하면 아래와 같은 명령어를 실행 할 수 있다.

```
sysprog$ echo "6.828 is cool" > x.txt
sysprog$ cat < x.txt
```

parser 가 명령어에서 “>”와 “<”를 인식하고 redircmd 에 parsing 해준다. 이 과제에서는 runcmd 의 각각의 심볼을 위한 코드를 작성해야 한다. man page 에서 제공하는 open / close 의 manual 을 읽어보는 것이 도움이 될 것이다.

redircmd 의 mode 필드가 O_RDONLY와 같은 접근 모드를 포함한다는 것을 유의하라. 이 필드는 open 함수의 flags 인자로 전달되는 값이다. shell 이 사용하는 mode 의 값을 확인하기 위해 parseredirs 를 확인하거나 매뉴얼 페이지에서 open 의 flags 인자를 확인하라.

마찬가지로 system call 이 실패했을 때 에러메시지를 출력하게하라.

위의 명령어를 통해 여러분의 구현이 제대로 실행 되는지 확인할 수 있다. 일반적인 에러는 파일이 생성될 때 권한을 부여하는 것을 잊었을 때 발생한다. (open system call의 3번째 인자)

Implement pipes

파이프를 구현하여 아래의 파이프라인 커맨드를 실행하라.

```
sysprog$ ls | sort | uniq | wc
```

parser 가 커맨드에서 'l'를 인식하고 pipcmd 로 parsing 해주기 때문에 파이프 구현을 위해 runcmd 에 'l'에 대한 코드만 작성하면 된다.

매뉴얼 페이지에서 pipe, fork, close, dup 시스템 콜을 살펴보는 것이 도움이 될 것이다. 위의 명령어를 수행하여 파이프가 정상적으로 실행되는 지 확인하라.

위 명령어에서 사용된 sort 등의 프로그램들은 /usr/bin/에 있을 수 있다. Sysprog shell 에서는 sort 를 실행하기 위해서 /usr/bin/sort 와 같이 절대경로를 입력해야 한다. (sysprog 이 아닌 여러분의 컴퓨터가 사용하는 shell 에서는 which sort 라는 명령어를 통해 sort 프로그램이 어느 디렉토리에 위치하는 지 확인할 수 있다.)

여기까지의 모든 구현을 마치면 아래의 명령어를 수행할 수 있을 것이다.

```
$ a.out < t.sh
```

프로그램에서 올바른 절대경로를 사용해야 한다.

과제 : 위의 기능들이 추가된 sh.c 소스파일
제출 양식 : **hw2_자신의학번.c**