# Operating System

## Homework #1

## -simple shell

**Sungmi Ryu**

**Dankook University**

**2018 Fall**

# Index

**Operating System : Simple shell**

**Sungmi Ryu 32161431**

# 1 Homework Introduction

이번 첫번째 숙제의 가장 핵심적인 개념은 "shell" 이라고 불리는 작은 프로그램을 만드는 것이다. 구체적인 구현에 관한 내용은 다음과 같다.

**1. 쉘의 시작 : 실행가능한 파일명 + Enter 을 입력함으로써 쉘이 시작하도록 한다.**

: 파일 a.out 은 소스코드의 컴파일 과정(>>gcc shell.c)이 성공적으로 수행되면 생기는 실행 가능한 파일이다. >>./a.out + Enter 로 쉘(SiSH)이 시작한다.

**2. 쉘의 끝 : 문자열 "quit" 을 받으면 쉘이 끝나도록 한다.**

: 쉘이 시작한 후에 입력값을 받는데에는 fgets() 함수와 stdin(표준 입력 스트림)을 이용한다. 그리고 만약 입력값이 스트링 "quit"이라면, 쉘이 끝난다. (즉, 소스코드가 끝난다는 의미)

**3. 쉘의 작동**

**1) 입력값 :** 문자열 형태의 프로그램 이름

: 입력값이 quit 이 아닐 경우, 받을 수 있는 입력값의 경우를 나누어 보았다.

⑴ 환경변수(ex. $PATH, $HOME, $PWD..)

⑵ 환경변수가 아닐 경우

① 명령어의 fullname 일 경우 (ex. /bin/ls, /bin/cp, /bin/rm ..)

② 명령어의 fullname이 아닐 경우

- 명령어의 간단한 이름일 경우 (ex. ls, cp, rm, echo $PATH ..)
- 명령어가 아닐 경우 (ex. aa, " ", ..)

**2) 실행 :** 만약 적절한 권한이 있다면, 모든 실행 가능한 프로그램을 실행해야 한다.

: 실행 가능한 프로그램이라면 모두 실행시켜야 함

**3) 실행경로 :** 유저가 프로그램을 실행하고자 할 때, '/'로 시작하는 전체 경로를 다 입력하기 보다는 보통 단순화한 실행파일명만 입력하는 경우가 많다. 이 경우, 쉘은 그 실행파일의 전체경로가 무엇인지를 환경변수 PATH에서 찾아보아야 한다.

: 환경변수인 $PATH는 명령어 cd.. 등을 제외하고 거의 모든 파일들의 경로를

가지고 있다. 예를 들어 PATH = a:b:c:d 라면, a,b,c,d 는 파일의 경로가 될 수 있는 디렉토리라고 할 수 있다. 일단, getenv() 함수를 이용해 PATH 라는 환경변수의 값을 읽어온다.

**4)** 환경변수 PATH는 :로 구분되어지는 많은 경로들을 가지고 있다.

: 읽어온 값을 strtok_r() 함수를 이용해 ":" 기준으로 토큰화 시킨 후, 그 경로에 파일이 있는지를 차례대로 탐색한다.

**5)** 유저가 입력하는 프로그램을 실행 중일 때, 쉘은 비활성화 상태여야 한다.

**6) 반복 :** 입력 받은 프로그램이 실행을 마치면, 계속해서 입력값을 받도록 한다.

**4. getenv() 함수를 이용해 다른 쉘 명령을 구체화할 수 있다.**

: 환경변수의 값을 읽어온다.

**5. 프로그램 실행 시 exec() 함수를 이용하는데 이때 여러 개의 인자를 받을 수 있다.**

# 2 Motivation

쉘은 운영체제와 유저 사이의 인터페이스 역할을 하는 일종의 명령어 해석기 프로그램이다. 따라서, 유저가 입력시킨 명령어 라인을 읽고서 필요한 시스템 기능을 실행시킬 수 있다. 이번 과제를 통해 간단한 쉘을 구현해 봄으로써, 사용자와 운영체제 간의 인터페이스 역할을 하는 쉘의 추상적인 개념을 직접 확인할 수 있게 된다. 좀더 구체적으로 설명하자면, 내가 구현한 쉘은 명령어 해석기와 같은 역할을 한다. 따라서, 나의 쉘에서 리눅스 명령어를 입력 받았을 때 이를 똑같이 실행하기 위한 방법을 찾아보게 되면서 리눅스 명령어의 단계적인 실행과정을 알게 된다. 우리는 편의를 위해 명령어의 전체 경로(/bin/ls) 대신 가령 ls 와 같은 실행 프로그램명만 입력하는데, 실제로는 명령어도 하나의 실행 프로그램이고 이를 성공적으로 실행시키기 위해서는 이 프로그램의 전체 경로를 찾아야 한다는 사실을 알게 된다. 이때 그 전체 경로는 환경변수 PATH에 저장되어 있으며 그 환경변수의 값을 불러오는 과정에서 getenv() 함수의 사용법을 익힐 수 있다. 환경변수의 값을 불러온 후, 많은 경로 중에서 실행 프로그램(e.g. ls)을 실행하기 위한 알맞은 경로가 있는지를 확인하는데 stat() 함수가 쓰임을 알고, 이를 활용하여 여러 입력값들에 대한 프로그램을 실행시킬 수 있게 된다.

# 3 Concepts used in CPU simulation

**1. fork() 함수**

- 호출한 프로세스의 복사본 프로세스(자식 프로세스)를 생성하는 함수

- 프로세스는 고유한 양수값 PID(Process ID)를 가짐

- fork 함수 실행시 부모 프로세스에서 자식의 PID 반환, 자식 프로세스에서 0 반환, 실패시 -1 반환

**2. execve() 함수**

- 현재 실행중인 프로세스를 내가 실행하고자 하는 다른 프로세스로 바꾸는 함수

- 내가 실행하고자 하는 프로세스를 child process에서 run할 수 있음

- execve("fullname",argv,NULL) (argv = 입력값)

**3. wait() 함수**

- 부모 프로세스의 실행을 중지하고 자식 프로세스의 실행이 끝날 때까지 기다리는 함수

- wait(0) 과 같은 형태로 사용

**4. getenv() 함수**

- 입력된 환경변수의 값을 읽어오는 함수

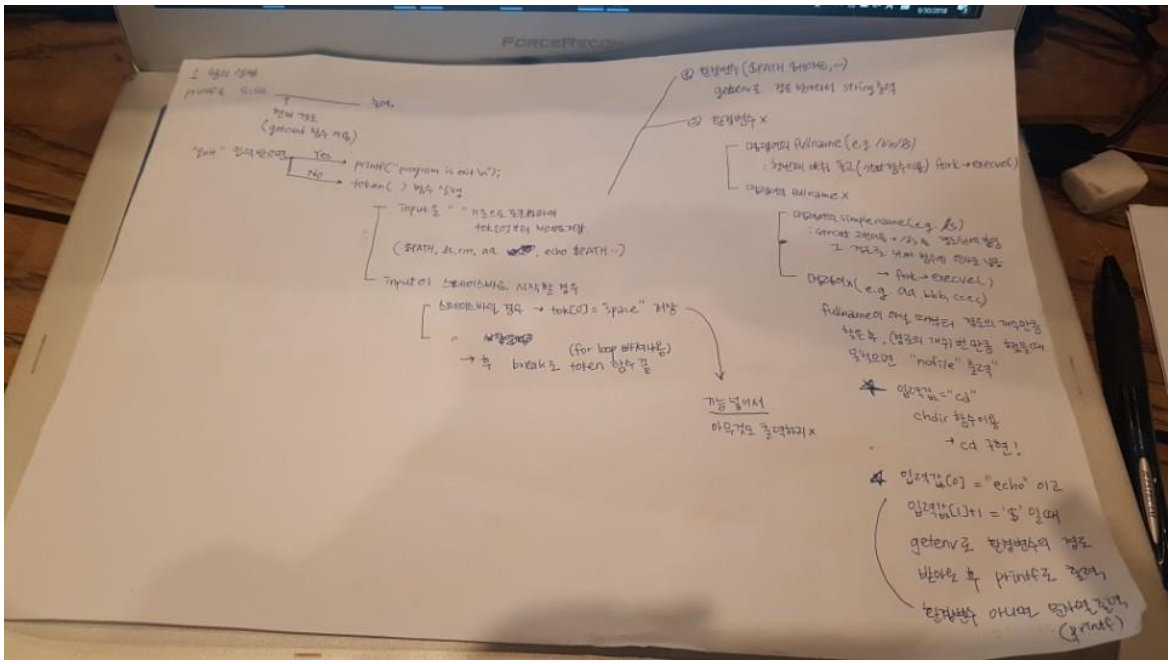
- getenv("PATH") 와 같은 형태로 사용 (PATH 는 환경변수), 리턴값은 스트링

**5. stat() 함수**

- 파일의 상태, 정보를 읽어오는 함수

- 있으면 0, 없으면 -1 반환

# 4 Program Structure



# 5 Problems and Solutions

처음 코드를 짤 때, 과제를 완벽하게 이해하지는 못한 상태로 제대로 된 알고리즘 없이 코딩 시작을 했다. 어떤 큰 틀이 없이 코딩을 하려다 보니 많은 곳에서 에러가 생기고 코드가 무작정 길어지게 되었다. 틀 없이 코드를 짜기 시작하면서 생긴 문제들이 몇 가지 있었는데 이에 대해 설명해 보려고 한다.

**1. main()이 너무 김**

일단, 처음에 코드가 길어지는 걸 생각하지 않고 모든 걸 main() 함수에 써서 main 이 길어지게 하는 실수를 했다. 내가 짠 코드임에도 불구하고 코드를 볼 때마다 다시 한번 머릿속에서 시뮬레이션을 해봐야 이해가 가는 코드였다. 게다가 변수를 많이 사용하다 보니 변수들의 이름이 비슷비슷해져서 교수님이 항상 말씀하시듯이 main() 함수는 최대한 짧은 것이 좋다. 그래야 코드가 간결해지고 정리가 잘 되어 있어서 가독성이

좋기 때문이다.

봐도 이해하기 힘든 이 코드의 문제를 해결하기 위해 main()을 쪼개서 여러 함수들의 모임을 만들었다. 중첩 if문을 많이 사용해서 이를 쪼개는데 값을 return 하고 그 함수를 빠져나와서 return value == 상수 일때 break; 로 loop을 빠져나오는 방식을 썼다.

**2. 명령어 cd 의 실행파일이 $PATH의 경로에 존재하지 않음**

대부분의 명령어들의 실행파일은 $PATH를 토큰화한 경로들에 존재하는데, cd는 이에 존재하지 않았다. chdir 함수를 이용해 성공적으로 현 디렉토리 옮길 시 0, 실패 시 -1을 반환하도록 하였다.

**3. space바 입력 시 segmention fault 뜨면서 프로그램이 종료됨**

space바를 받으면 " "기준으로 토큰화하는 token 함수에서 tok[0]에 NULL 값이 들어가기 때문에 오류가 떴다. 이를 해결하기 위해 if(tok[0] == NULL) 일 때 tok[0]에 새로운 스트링을 저장하고, 이 케이스를 따로 빼서 break; 로 loop을 빠져나오도록 하였다.

**4. 파일의 fullname(e.g. /bin/ls)을 받을 때 실행되지 않음**

fullname 입력시, 일단 경로의 개수만큼 tok[i]를 전체적으로 한바퀴 찾는 방식을 택했다. 만약 simple name 을 입력했다면, 전체 한바퀴 이후 PATH 의 토큰화된 경로에 "/ls" 를 붙인 값을 한번 더 찾은 후 성공시 fork 함수 실행, 실패시 "nofile" 을 출력하도록 하였다.

**5. echo $PATH 입력 시, 스트링 "$PATH" 이 출력됨**

입력받은 스트링[0]이 "echo" 일 때의 케이스에서, 입력받은 스트링[1]+1이 '$'라면 getenv 함수를 이용해 PATH의 경로를 받아온 후 이를 printf를 이용해 출력하도록 하였다. '$'가 아닐 경우 그대로 문자열을 출력하도록 하였다.

**6. 메모리 문제로 segmentation fault 뜸**

메모리를 할당하는 방식이 어려웠다. 계속 segmentation fault(core dumped) 가 뜨는 상황에서 동적할당으로 메모리를 바꿔주거나 메모리 사이즈를 크게 할당해 주었을 때 오류가 사라졌다.

**7. makefile 만들고 컴파일 시 오류 뜸**

이전에 실행했을 때의 주소값이 남아있었는데 clean 을 해주지 않아서 계속 오류가 떴다. 헤더파일을 다시 선언하고 저장함으로써 오류가 사라졌다.

# 6 Build environment

**컴파일 : linux assam, with GCC**

****기본 틀(입력 값 치기 전 컴파일 과정 – 실행파일)**

>> cd 2018_os_hw1 //2018_os_hw1 으로 현 디렉토리를 옮김

>> ls //현 디렉토리에 있는 파일들을 나열함

>> gcc shell.c //gcc 를 사용해 내 코드 shell.c 를 컴파일함

>> ./a.out //성공적으로 컴파일되어 만들어진 파일 a.out을 실행

****환경변수**

>> $PATH($HOME, $PWD, ..)

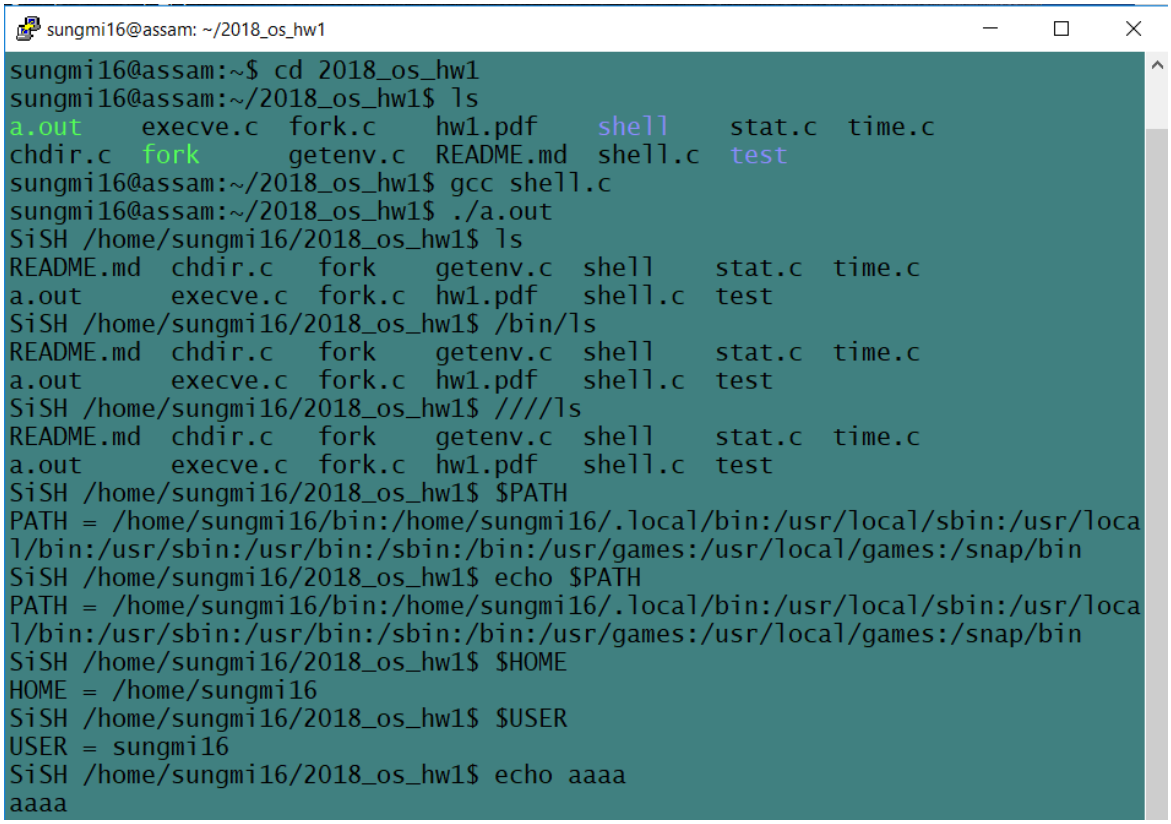
****리눅스 명령어**

>> ls(rm, cp a.c b.c, date, echo sungmi .. )

****출력**

>> echo $PATH

# 7 Screen capture




***vi Makefile**


# 8 Personal feelings

이제 운영체제 과목의 첫번째 과제가 주어졌는데, 과제를 하면서 내 자신에 대한

**Operating System : Simple shell**

**Sungmi Ryu 32161431**

부족함을 많이 느꼈다. 처음 과제를 읽고 코드를 짜기 시작할 때 나의 목표는 최소한 기본 구현은 해서 내는 것이었다.

과제를 하는데 있어서 가장 힘들었던 점은 일단 주어진 과제를 이해하는 것이었다. 수업시간에 배웠던 fork() 함수의 개념을 이해하지 못하고 넘어갔던 나는 처음 과제를 읽었을 fork() 함수 외에 getenv(), execve(), wait() 등의 함수들의 개념을 이해하는 것이

힘들었다. 하지만 목표한 바를 이루기 위해 계속 읽고 모르는 개념을 하나하나 찾아가면서, 그리고 주변 친구들에게 물어가면서 과제를 이해해 나가기 위해 노력했다.

둘째로, 사용해야 하는 여러 함수들의 개념을 이해하는 게 힘들었다. 최대한 많은 리눅스 명령어들을 구현하기 위해서는 C 라이브러리에 있는 함수들을 사용해 코드를 짜야 했다. getcwd, strcpy, strcmp 부터 시작해서 추가 구현을 위해 써야 했던 getenv, execve, fork 등등 여러 생소한 함수들의 개념을 이해하고 사용법, 특히 함수에서 사용하는 인자들의 형(int, char, char* 등)을 익히는 것이 가장 어려웠다. 이 코드를 이해하고 사용하기 위해서, 인터넷에서 찾은 예제 코드를 분석하고 내 코드에 맞게 다시 짜는 방법을 선택했다.

셋째, 추가 구현을 하면 할수록 나오는 예외들, 구현되지 않은 것들이 많았다. 나름대로 열심히 코드를 짠다고 짰지만, TIME 이라는 환경변수를 구현하지 못한 것, 그리고 echo $PATH “string” $HOME 을 입력할 때 세 개 다 한꺼번에 출력되야 하는 것을 구현하지 못한 것이 많이 아쉽다. 하지만 학우들과 함께 예외의 상황을 찾아가면서 각자만의 방식으로 코드를 짜고 오류를 줄여나가는 과정이 매우 재미있었다. 이번 과제는 이해하는 것이 어려웠지만 이해하고 나서는 충분히 할 수 있었던 과정이라고 생각한다. 다음에 배울 내용이 기대되고, 이 기세로 다음 과제들과 프로젝트는 더 열심히 해서 추가구현도 완벽하게 해내고 싶다.