**Oeprating System**

Project #1

-Simple Shell

단국대학교

모바일시스템공학과

32144055 장준모

2018. 9. .30

FreedaysLeft : 5

**Project Introduction :**

Simple Shell을 만든다.

* fgets()로 cmd를 받아준다.
* getenv()를 이용하여 환경변수를 설정한다.
* stat()을 이용하여 그 파일의 존재 여부를 확인한다.
* fork()를 이용하여 Child Process를 만든다.
* execve()를 이용하여 복사된 Child Process를 원하는 것으로 바꿔준다.

1. **Motivation :**

Simple Shell을 직접 구현해봄으로써 LINUX 상에서 bash나 내부 명령함수가 어떤 방식으로 구현되는지 알 수 있다.

1. **Program Structure :**

프로그램은 main 함수 안에 하나의 while loop으로 실행된다.

변수 선언

+-----------------while------------------+

fgets(cmd)

cmd뒤에 개행문자 삭제

+---------for-----------+

cmd tokenize

+-----------------------+

if(cmd = quit)

else if(cmd = quit)

else if(cmd = pwd)

else if(cmd includes “/”)

stat()

else if(cmd = time)

else if(cmd = cd)

else

+---------for-----------+

path tokenize

strcpy(path, cmd)

stat()

+-----------------------+

fork()

execve()

+----------------------------------------+

While loop을 들어가기 전에 모든 변수들을 선언해준다. While loop에서 cmd를 받아주고 여러 인수를 받더라도 실행 가능하게 바로 토큰화를 한다. If-else if 문으로 cmd가 quit, user, pwd, /를 포함, time, cd 혹은 나머지 상황인지를 판별하여 각각 실행한다. 나머지인 경우에는 for loop으로 PATH의 환경 변수를 “:”으로 토큰화를 해주고, 토큰화가 된 것들에 /와 cmd를 붙여 full이라는 스트링에 저장을 해준다. 저장된 full은 stat함수에 들어가 존재 여부를 확인하고 만약 존재한다면 exec\_cmd에 저장된다. (존재 안 할 경우 그냥 넘어간다) 그렇게 나머지 상황에 대한 처리가 끝나면 fork()를 실행하여 child process를 만들고 execve()를 한다. execve()까지 끝나면 다시 while loop으로 올라와 다시 실행 할 수 있게 해준다.

1. **Problems and Solutions :**

이번 과제의 핵심은 스트링과 캐릭터의 구분이다. Execve()를 실행하는데 그 전 단계에서 출력을 하면 들어가야 할 값이 잘 나오는데 실행은 되지 않았다. 그 이유는 스트링의 형태로 들어가야 할 인수 자리에 캐릭터 타입으로 넣었기 때문이다. 그리고 fgets 함수를 이용하여 라인을 받으면 마지막에 개행 문자가 들어가는 것을 몰라 한참을 고생하였었다. 이 정도 문제들을 제외하고는 그렇게 어려운 개념이나 문제는 없었다. 실행하는 과정이 가시적이지 않고 추상적인 느낌이 강하여 에러를 고치는데 있어 조금 난해하긴 하였다.

1. **Build Environment :**

Compilation : Linux Assam, with GCC

To compile, please type : gcc myshell.c

To run, please type : ./a.out

1. **Screen Capture :**



각 변수들의 설정이다.

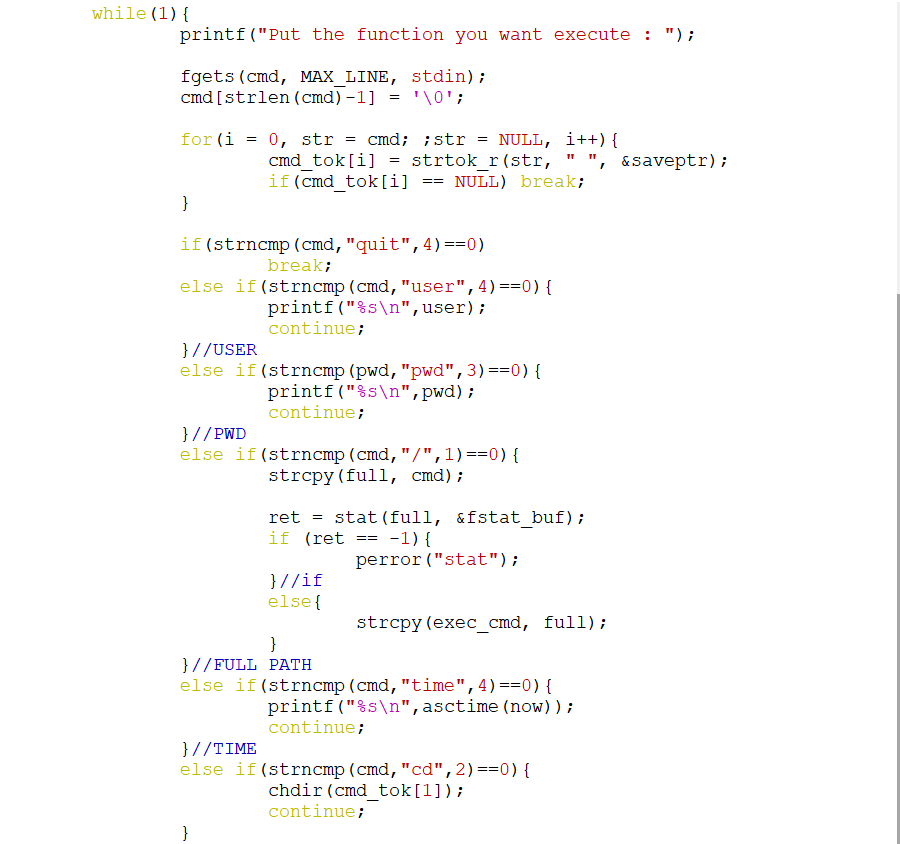
Exec\_cmd는 fork()를 한 후, execve의 첫번째 인수로 들어간다.

Cmd\_tok는 받아오는 cmd를 띄어쓰기 단위로 토큰화 하기 위함이다.

Cmd는 받아오는 라인이다.

Full은 stat()에 넣어 존재 여부를 확인하기 위함이다.

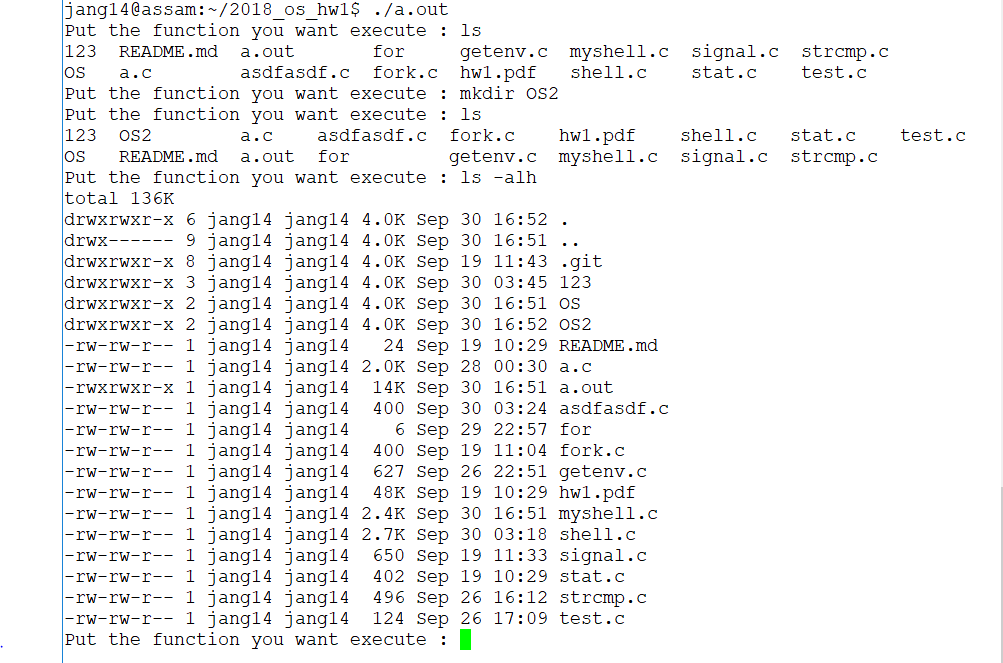
saveOOO은 getenv()를 하고 임시 저장장소이다.



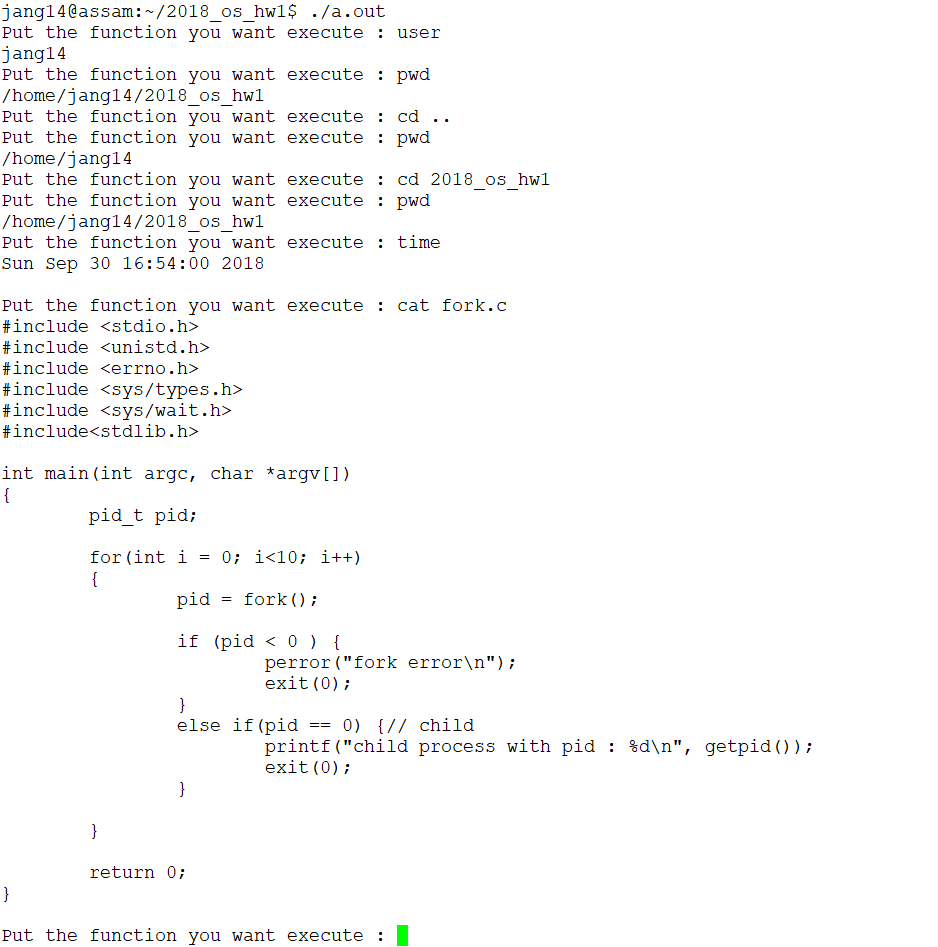
fgets로 cmd를 받아준다. For loop으로 cmd를 cmd\_tok에 토큰화를 하여 넣는다. Cmd가 quit이면 프로그램을 끝낸다. Cmd가 user면 위에서 getenv(“USER”)를 실행한 것을 출력한다. Cmd가 pwd면 위에서 getenv(“PWD”)를 실행한 것을 출력한다. Cmd에 “/”가 포함 되어있으면 그 자체가 full이 되어 stat()을 실행하고 존재한다면 exec\_cmd로 저장한다. Cmd가 time이라면 위에서 설정한 현재 시각을 출력한다. Cmd가 cd면 chdir()에 토큰화한 cmd의 두번째 값을 넣는다.



나머지의 경우는 getenc(“PATH”)를 for loop으로 토큰화 해준다. 토큰화 한 것 뒤에 “/”를 붙이고, cmd\_tok의 첫번째를 붙여 full을 완성한다. Full을 stat()에 넣어 존재여부를 확인하고 만약 존재한다면 exec\_cmd에 저장한다. Path에 savepath를 넣는 이유는 for loop을 돌고 나면 path가 토큰화 된 첫번째 스트링만 기억을 하기 때문이다. 그리고 fork를 실행하고 execve()을 실행한 후 다시 while loop을 돈다.



Ls와 mkdir을 한후 ls에 OS2가 생성되었다. Ls -alh도 실행한 모습이다.



User를 확인하고, pwd로 현제 위치를 찍어보았다. Cd .. 으로 상위 디렉토리로 간 것을 확인하고 다시 돌아왔다. time으로 현제 시각을 출력하였고 cat fork.c를 출력해보았다.

1. **Personal Feeling :**

이번 학기가 개강하고 운영체제 수업이 많이 없었다. 낯선 과목에 대한 수업인데, 수업까지 몇 번 없어 과제를 시작하기가 조금 어려웠다. 생소한 함수 fork, execve, wait을 꽤 오래 구글링을 하여 살펴보았다. 중요하고 큰 개념은 이해가 된 상태에서 과제를 시작하였는데, 생각보다는 쉽지 않았다. 하지만 이번 계기를 통해 알게 된 점은 내가 프로그래밍의 기초 상식에 많은 빈틈이 있다는 것이다. 스트링과 캐릭터 타입에 혼동이 왔고, 그것을 해결하는데 많은 시간을 할애 하였다. 직접 종이에 어레이를 그려보고 한 칸에 스트링이 들어가는지 캐릭터가 들어가는지를 파악하고 에러를 고쳤다. 그 덕분에 지금은 완벽히 구현하게 되었다. 과제를 처음 시작할 때에는 “당연히 이번 과제의 핵심은 fork이다” 라고 생각했는데, 시작해보니 fork, wait, exec은 너무 쉬운 개념이었고, 생각지도 못한 부분에서 애를 먹었다.

아쉬운 점은 time관련해서 getenv를 하면 아무것도 지정되어 있지 않은 NULL상태 인 것은 알았는데 setenv를 이해 하지 못하여 원하는 방식데로 구현하지 못한 점이다. 그리고 처음 구현 할 때 모든 것들을 다 main함수 안에 넣어 기본 구현을 마친 후 여러 함수들을 따로 구현하고 main함수는 간결하고 짧게 표현하고 싶었지만, 게으른 성격 때문에 그러지 못한 것이 아쉽다.

지난 학기 모바일프로세서 첫번째 과제인 계산기와 비슷한 수준으로 생각하고 과제에 임하였는데, 너무 가벼운 마음으로 임하여서 그런지 시간에 쫓기듯이 과제를 마무리 하였다. 다음 과제부터는 조금 일찍 시작하여 여유롭게 할 생각이다.