Operating Systems & Adv. Mobile Project

HW1: Simple Shell(SiSH)

단국대학교 국제학부
모바일시스템학과
이종원 32143605
2018/09/30
Freedays Left: 5

1. Introduction

In this project we were tasked to create a simple Shell program. Shell is the program which receive inputs and executes the corresponding executable file in certain destination. In this project we need to utilize many functions, one of which is fork(). Fork() is the function where the program creates a child process. This is crucial to be able to return to the shell program.

2. Goal

In more detail our goal is to receive executable files from the user and execute the corresponding program in the $PATH environment. For example if the user types in the command “ls” the shell should find the “ls” in “/bin/” and execute it. Also if the user inputs “ls” or “/bin/ls” the shell should process and execute the same executable. Also additional instruction such as “ls –a” should be executed. Lastly, before the input, the shell should print the “USER”, “PWD”, and time.

3. Structure of the Code

The key functions of this program is printInfo(), getInput(), tokenInput(), checkInput(), and execute(). In the beginning I allocate memory to the string array and the other variable that I need. Then the main function enters while(1) loop.

First function in the while(1) loop is printInfo(). In this function we print out the user, current directory, and current date/time. We get user information string by getenv(“USER”) and current directory by getenv(“PWD”). The time is received by the ctime() function in time.h.

The second function is getInput(). In this function the program recieves input by getline() function. However getline() also includes

‘\n’ new line character at the end of the input string. As such we replace the ‘\n’ new line character into ‘\0’ null character. Then the program checks if the input is “quit” by using string compare function strcmp() and if the input is quit, the program exit(0) terminate the shell. Lastly I check the first character of the input for ‘/’. If the first character is ‘/’ it means that the user input the directory as well. If there is no ‘/’ it means that the user did not input the directory. I created slashFlag to save if input has directory or not.

The third function is tokenInput(). In this function, I tokenize the additional instruction of the input executable. For example if the input was “/bin/ls -a”, I tokenized by empty space character to separate all the additional instruction.

The fourth function is checkInput(). In this function we utilize the stat() function which checks if the executable exists. If the slashFlag is checked which means that the user included the directory, I stat() the input directory. If there is no directory, I getenv(“PATH”), tokenize them with “:”, and strcat() the directories with the input and stat() checks them all.

The last function in the while(1) loop is execute(). If the previous checkInput() function had a successful stat() check, it executes the corresponding executable. To do this I fork() to create a child process. Then if the pid == 0 it means that it is a child process and I run the execve(). If pid is more than 0 I wait(0) to pause the parent process.

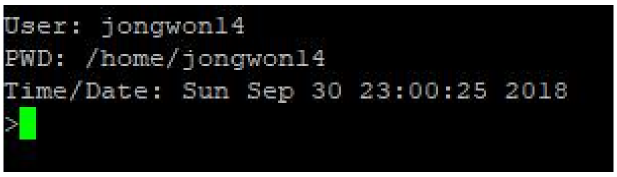
After running the input executable, we continuously recieves input until the input is quit.

4. Difficulty & Problems / Solution

One of the difficulties I encountered is using execve() function. It was difficult to understand the parameters it required. Also other difficulties I had encountered is the “Segmentation Fault: Core Dumped” error with all the strings and other variables.

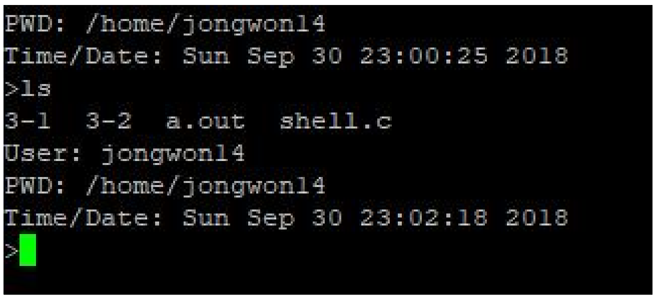
5. Outcome

Printing USER, PWD, and Date/Time



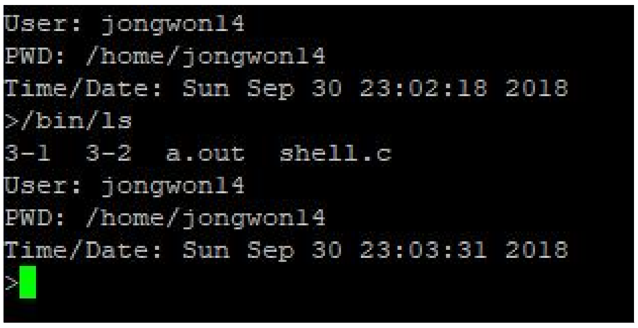
```
User: jongwon14
PWD: /home/jongwon14
Time/Date: Sun Sep 30 23:00:25 2018
>
```

Input without directory



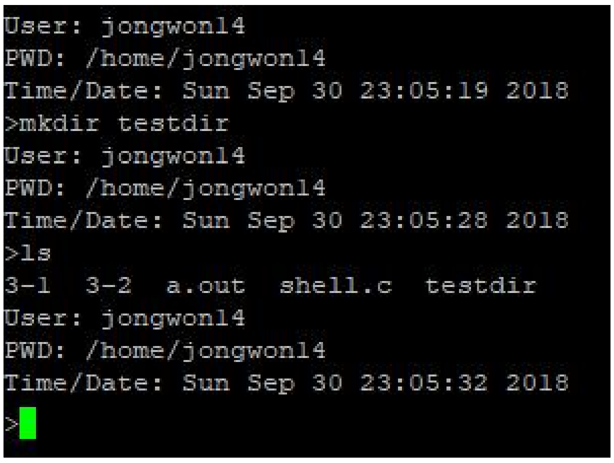
```
PWD: /home/jongwon14
Time/Date: Sun Sep 30 23:00:25 2018
>ls
3-1  3-2  a.out  shell.c
User: jongwon14
PWD: /home/jongwon14
Time/Date: Sun Sep 30 23:02:18 2018
>
```

Input with directory



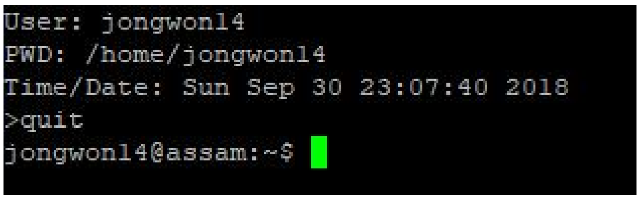
```
User: jongwon14
PWD: /home/jongwon14
Time/Date: Sun Sep 30 23:02:18 2018
>/bin/ls
3-1  3-2  a.out  shell.c
User: jongwon14
PWD: /home/jongwon14
Time/Date: Sun Sep 30 23:03:31 2018
>
```

Input with additional instructions



```
User: jongwonl4
PWD: /home/jongwonl4
Time/Date: Sun Sep 30 23:05:19 2018
>mkdir testdir
User: jongwonl4
PWD: /home/jongwonl4
Time/Date: Sun Sep 30 23:05:28 2018
>ls
3-1  3-2  a.out  shell.c  testdir
User: jongwonl4
PWD: /home/jongwonl4
Time/Date: Sun Sep 30 23:05:32 2018
>
```

Input Quit



```
User: jongwonl4
PWD: /home/jongwonl4
Time/Date: Sun Sep 30 23:07:40 2018
>quit
jongwonl4@assam:~$
```

6. Conclusion

In conclusion, this project was very enjoyable as I was extensively utilize the shell I have create. However fixing all the memory allocation segmentation fault error was very aggravating.