Group: Gabriela Acevedo, Zubair Matani, Baudoin Ramazani, Deandra Rodricks

Dr. Kim

ECE 408/ CSCI 403

Project 1 Report

Purpose:

The purpose of this project is to familiarize ourselves with a Linux system and learn how to use it to execute multiple commands and files. We will be implementing a command line interpreter to make use of both interactive and batch modes, and further understand the specific difference between both modes. Our method of choice for this project is Raspbian on Raspberry Pi hardware.

Work Division:

- Baudouin M Ramazani implemented the batch mode, concurrent command running in the program.
- Deandra Rodricks researched background and contributed with code implementation ideas for the program and report writing.
- Gabriela Acevedo researched about the inbuilt function which made this program easier to design and implement and report writing.
- Zubair Matani coded the interactive mode of the shell along with building functions such as execute and makeToken to allow the program to run as per requirements.

Implementation Method:

The code had three main parts to work through to make it work efficiently and fulfill all requirements:

- Create a basic C code that would implement the working of a simple linux terminal and execute commands. This would help us achieve the interactive mode of the program.
    - The first function we created was a print prompt function which basically prints a text string, just like a terminal system prints. In our case, the prompt prints *strawberry* which is our group's name.

    - For interactive mode to work, the function in the code is void () execute, this function uses the fork function, which creates new context based on the context of the calling system.
    - If the fork command is successful, it returns a number of pid which is greater than 1, indicating that a new process has been created. In case the fork system fails, then it returns a number less than 0.
    - This is completely what this function does, and responds back with the proper output each time.
- Once that was properly implemented, then we had to edit the code and make sure the program can take in more than one argument from the console, mainly file and shell scripts so that it would work in batch mode.
    - In order to implement the batch mode, we worked on our main function and allowed it to take in more than one argument when running on the console. We used the in built file functions which are used to open and read a file. Once opened, the program scans the commands and then starts to run them one by one.
    - This part of the code also helps to run shell scripts in the program which run as expected.
- Once both environments were implemented and responded properly, then the rest of the requirements were implemented as asked in the project guidelines.
    - In order for the program to run multiple commands each time which are supposed to be separated by a semicolon, we used the inbuilt function of strtok() in the function makeTokens. This allows the program to make small snippets of strings from a given command each separated by the semi colon.
    - Once the commands are separated by the program using this function, we call the execute function to run the commands and output the results.
    - In order to make the program to ignore the commands which were given after a # sign, we simply set a few if statements in the main function which allowed the program to simply ignore and move to the next command.
    - Inorder for the program to detect that the user wants to exit the program, we simply set a if statement that compares strings for “exit” and closes the program

once detected. If the user tries to forcefully exit the program using Control D, that is also detected and the program exits.

    - For each wrong command typed, the program gracefully gives a warning message letting the user know that the command is wrong, and continues to take in commands.

Conclusion:

- The assignment asked us to implement a command line interpreter that would work interactively as well as process commands in a batch.
- This program taught us numerous new concepts and helped to gain a more in depth understanding of using C as a programming language.
- Some inbuilt C functions were very helpful to execute this program, which includes fork(), strtok() which we have not used before.
- The appendix on the next page shows the program, the compilation result and the output when we tested the required steps as guided in the assignment.

Appendix:

C Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>   // Definition for fork() and execve()
#include <errno.h>    // Definition for "error handling"
#include <sys/wait.h> // Definition for wait()

   /* Declarations for getline() */
   char *input = NULL;
   size_t capline = 0; // Capacity

   /* Declaration for strtok() */
   int i;
   char *token;
   char *array[512];


   /* Print out "prompt" */
   void displayPrompt(){
       printf("Strawberry: ");
   }

   /* Divide input line into tokens */
   void makeTokens(char *input){
       i = 0;
       token = strtok(input, "\n");
           while (token != NULL) {
               array[i++] = token; // Add tokens into the array
               token = strtok(NULL,";");
               token = strtok(NULL, "\n ");
           }
       array[i] = NULL;
   }
   /* Execute a command */
  void execute(){
      int pid = fork(); // Create a new process
               if (pid != 0) { // If not successfully completed
                   int s;
                   waitpid(-1, &s, 0); // Wait for process termination
           } else {
                   if(execvp(array[0], array) == -1){ // If returned -1 =>
something went wrong! If not then command successfully completed */
```

```
                perror("Wrong command"); // Display error message
                exit(errno);
            }
        }
}


 int main(int argc, char *argv[]){
     if(argc == 1){
         displayPrompt();
        int i = 0,j=1;
        char temp [512]={0};
        char line[512] ={0};
        char str [20];
        const char semicolon[2] = ";";
        const char *command;
        while (fgets(temp, sizeof(temp),stdin)){
            displayPrompt();
            while(temp[i] != '\0'){
                if(temp[i] == '#')
                    break;
                else{
                    line[i] = temp[i];
                    i++;
                    }

            }
            i=0;
            command = strtok(line, semicolon);
            while(command != NULL){
                execute();
                system(command);
                command = strtok(NULL, semicolon);
            }
            j++;
            memset(&line[0],0,sizeof(line));
         }




    } else if(argc == 2){
     int i = 0,j=1;
     char temp [512]={0};
     char line[512] ={0};
     char str [20];
     const char semicolon[2] = ";";
```

```
        const char *command;
        char * file = argv[1];
        FILE* user_file = fopen(file,"r");
        while(!user_file){
            printf("File not found! Try again...\nFile name: ");
            scanf("%s",str);
            user_file = fopen(str,"r");

        }
        while (fgets(temp, sizeof(temp),user_file)){
            while(temp[i] != '\0' ){
                if(temp[i] == '#')
                    break;
                else{
                    line[i] = temp[i];
                    i++;
                    }
            }
            i=0;
            command = strtok(line, semicolon);

            while(command != NULL) {
                execute();
                system(command);
                command = strtok(NULL, semicolon);
            }
            j++;
            printf("\n[Line %d]: %s ---- End of execution ----\n\n",j,
temp);
            memset(&line[0],0,sizeof(line));
        }
        printf("End of file");
        printf("\n\n\n== Done! ==");
        fclose(user_file);

    }
}
```

Bash Script to Test:

```
#!/usr/bin/sh

pwd
```

```
clear
date
df
du
less
find okc.txt; #pwd; less
pw#d
who
touch okc.txt; ls;# who; du; less
# Ignore this line
# THis one too!
#pwd
dff

dda; dhd; sj; pwd

exit
```

Results:
Comments are mentioned in front of a few commands to explain.

Batch Mode: (uses script shell file to run a set of commands)

```
zubairmatani@Zubairs-MacBook-Air project1 % clear

zubairmatani@Zubairs-MacBook-Air project1 % ./Shell script_1.sh

[Line 2]: #!/usr/bin/sh
 ---- End of execution ----


[Line 3]:
 ---- End of execution ----

/Users/zubairmatani/Desktop/project1

[Line 4]: pwd
 ---- End of execution ----
```

```
[Line 5]: clear
 ---- End of execution ----

Mon Feb 15 22:18:28 EST 2021

[Line 6]: date
 ---- End of execution ----

Filesystem    512-blocks      Used Available Capacity iused      ifree
%iused  Mounted on
/dev/disk1s5   236306352  21650448  38693096    36%  488248 1181043512
0%   /
devfs                373       373         0   100%     646          0
100%   /dev
/dev/disk1s1   236306352 170322336  38693096    82% 1016115 1180515645
0%   /System/Volumes/Data
/dev/disk1s4   236306352   4196392  38693096    10%       3 1181531757
0%   /private/var/vm
map auto_home          0         0         0   100%       0          0
100%   /System/Volumes/Data/home
/dev/disk1s3   236306352   1033112  38693096     3%      50 1181531710
0%   /Volumes/Recovery

[Line 7]: df
 ---- End of execution ----

248   .

[Line 8]: du
 ---- End of execution ----

Missing filename ("less --help" for help)

[Line 9]: less
 ---- End of execution ----

okc.txt

[Line 10]: find okc.txt; #pwd; less [Concurrent commands running]
 ---- End of execution ----
```

```
sh: pw: command not found

[Line 11]: pw#d
 ---- End of execution ----

zubairmatani console  Feb 14 00:45
_mbsetupuser console  Feb 10 18:45
zubairmatani ttys000  Feb 15 22:17

[Line 12]: who
 ---- End of execution ----

Shell       abc.txt           okc.txt           script_1.sh
a.out       backup.rtf  os1.c

[Line 13]: touch okc.txt; ls;# who; du; less
 ---- End of execution ----


[Line 14]: # Ignore this line
 ---- End of execution ----


[Line 15]: # THis one too!
 ---- End of execution ----


[Line 16]: #pwd              [running commands with # sign]
 ---- End of execution ----

sh: dff: command not found

[Line 17]: dff
 ---- End of execution ----


[Line 18]:
 ---- End of execution ----

sh: dda: command not found
sh: dhd: command not found
sh: sj: command not found
/Users/zubairmatani/Desktop/project1

[Line 19]: dda; dhd; sj; pwd      [wrong and right commands together]
 ---- End of execution ----
```

```
[Line 20]:                [No command provided, test of empty line]
 ---- End of execution ----


[Line 21]: exit                 [exit program]
 ---- End of execution ----

End of file


== Done! ==%    [Shell file read, and program exits]
```

Interactive Mode:

```
zubairmatani@Zubairs-MacBook-Air project1 % gcc os1.c -o Shell [Compiles]
zubairmatani@Zubairs-MacBook-Air project1 % ./Shell
Strawberry: ls
Shell      abc.txt        okc.txt        script_1.sh
a.out      backup.rtf  os1.c
Strawberry: ls; pwd                               [Concurrent commands]
Shell      abc.txt        okc.txt        script_1.sh
a.out      backup.rtf  os1.c
/Users/zubairmatani/Desktop/project1
Strawberry: #pwd                                  [Ignores command]
Strawberry: echo "Zubair"
Zubair
Strawberry: df
Filesystem    512-blocks      Used Available Capacity iused      ifree
%iused  Mounted on
/dev/disk1s5   236306352  21650448  38713080    36%  488248 1181043512
0%   /
devfs                373       373         0   100%     646          0
100%   /dev
/dev/disk1s1   236306352 170302352  38713080    82% 1016213 1180515547
0%   /System/Volumes/Data
/dev/disk1s4   236306352   4196392  38713080    10%       3 1181531757
0%   /private/var/vm
map auto_home          0         0         0   100%       0          0
100%   /System/Volumes/Data/home
/dev/disk1s3   236306352   1033112  38713080     3%      50 1181531710
0%   /Volumes/Recovery
Strawberry: date
Mon Feb 15 22:24:30 EST 2021
Strawberry: date; whoami; ls
```

```
Mon Feb 15 22:24:38 EST 2021
zubairmatani
Shell      abc.txt          okc.txt          script_1.sh
a.out      backup.rtf  os1.c
Strawberry: ldd                   [Wrong command generates warning]
sh: ldd: command not found
Strawberry: Zubair
sh: Zubair: command not found
Strawberry: %               [Exit using control d]
```