Hengnan Ma
Fakharyar Khan

Professor Hakner
ECE357

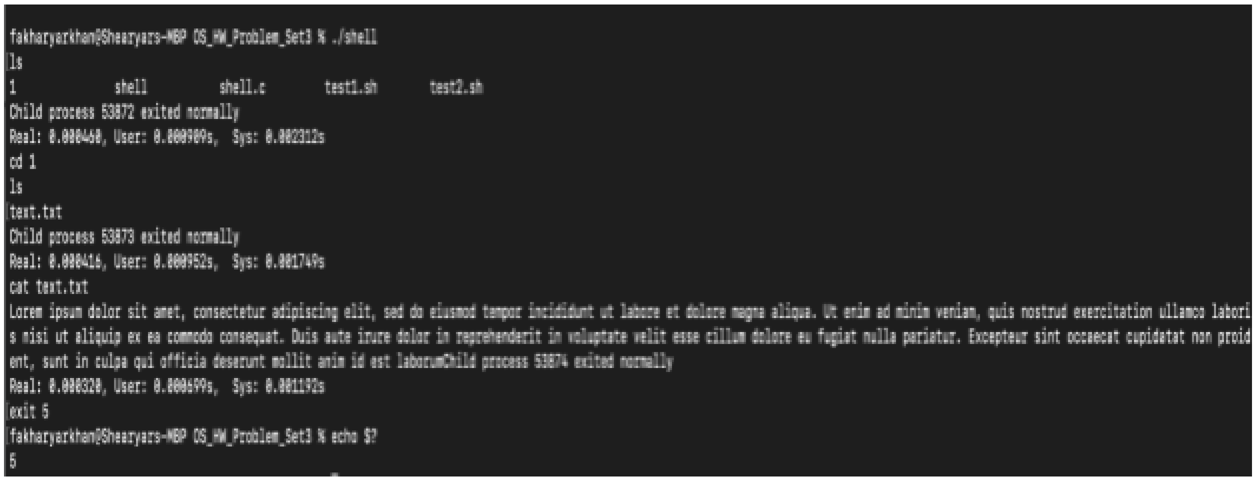
Problem Set 3 Q3

Program #3 Writeup: Simple Shell Program

In this project, we created a basic shell that's able to execute commands given by the user and can also execute bash scripts. To run the program without a bash script, you just need to compile and execute our program without any arguments. Then you can enter in any commands you would like just as you would in a regular terminal. To exit from the program, you can either hit control D or enter in the exit command. The example below demonstrates how our program works in this mode.

Test Case #1: Executing Commands in Terminal

We first read through the contents of the folder by using the ls command. After the command is executed, the program also lists the real, user, and system time taken to execute this command. We then cd into one of the folders and then display the contents of the folder as well as text.txt, a file inside the folder. Finally, we exit the shell with the exit command.



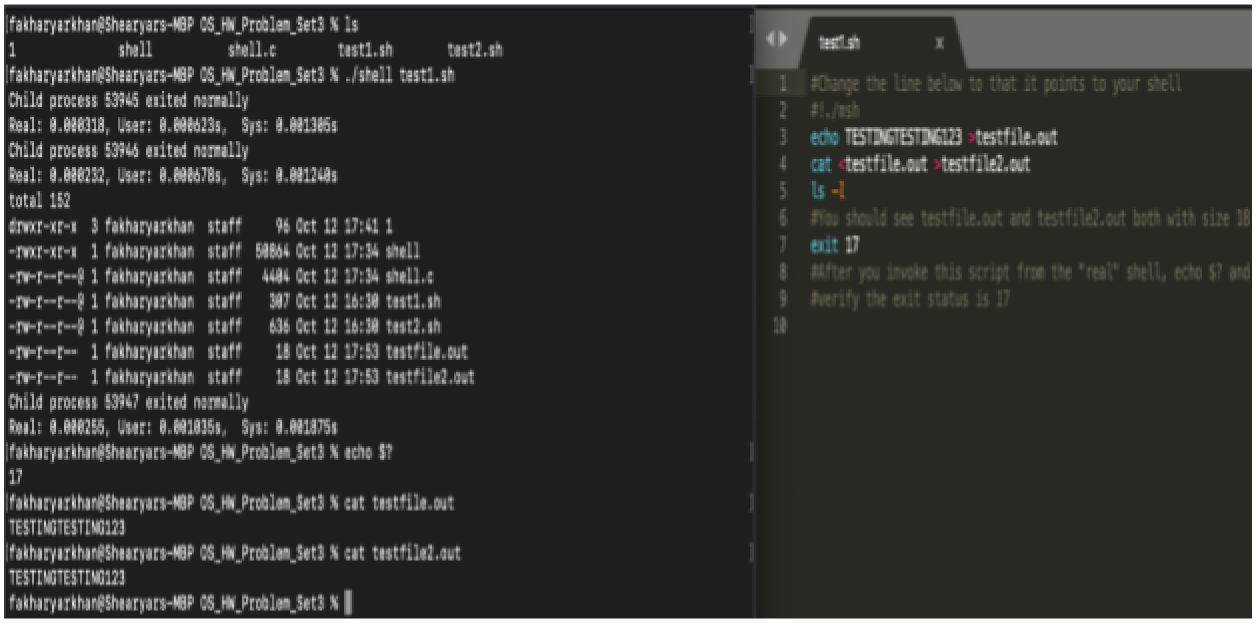
```
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % ./shell
ls
1            shell         shell.c       test1.sh      test2.sh
Child process 53872 exited normally
Real: 0.000460, User: 0.000909s,  Sys: 0.002312s
cd 1
ls
text.txt
Child process 53873 exited normally
Real: 0.000416, User: 0.000952s,  Sys: 0.001749s
cat text.txt
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco labori
s nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proid
ent, sunt in culpa qui officia deserunt mollit anim id est laborumChild process 53874 exited normally
Real: 0.000320, User: 0.000699s,  Sys: 0.001192s
exit 5
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % echo $?
5
```

Figure 1: Sample Run of Program Without Bash Script

Test Case #2: Executing Commands from Bash Scripts and File Redirection

Similiarly, to run shell scripts in our shell, simply type the executable file name followed by the name of the shell script. For example, if the executable name is "shell" and the shell script is "test1.sh", one can run the shell script file by typing "./shell test1.sh" into the terminal as shown in Figure 2.

We see here with the cat command on line 4 of the bash script that the shell is also able to support file redirection. This command redirects standard input to testfile.out and standard output to testfile2.out so that the contents of testfile.out get written to testfile2.out.



```
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % ls
1             shell         shell.c        test1.sh       test2.sh
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % ./shell test1.sh
Child process 53945 exited normally
Real: 0.000318, User: 0.000623s,  Sys: 0.001305s
Child process 53946 exited normally
Real: 0.000232, User: 0.000678s,  Sys: 0.001240s
total 152
drwxr-xr-x  3 fakharyarkhan  staff    96 Oct 12 17:41 1
-rwxr-xr-x  1 fakharyarkhan  staff  50864 Oct 12 17:34 shell
-rw-r--r--@ 1 fakharyarkhan  staff   4404 Oct 12 17:34 shell.c
-rw-r--r--@ 1 fakharyarkhan  staff    307 Oct 12 16:30 test1.sh
-rw-r--r--@ 1 fakharyarkhan  staff    636 Oct 12 16:30 test2.sh
-rw-r--r--  1 fakharyarkhan  staff     18 Oct 12 17:53 testfile.out
-rw-r--r--  1 fakharyarkhan  staff     18 Oct 12 17:53 testfile2.out
Child process 53947 exited normally
Real: 0.000255, User: 0.001035s,  Sys: 0.001875s
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % echo $?
17
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % cat testfile.out
TESTINGTESTING123
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % cat testfile2.out
TESTINGTESTING123
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 %
```

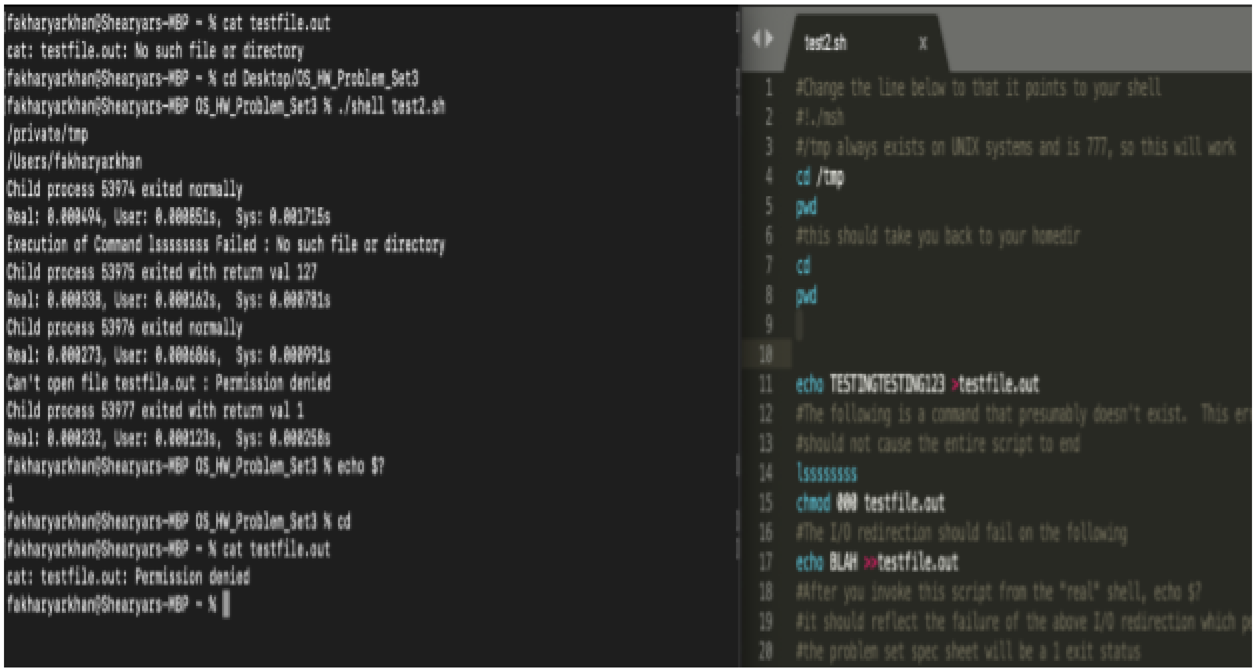
test1.sh

```
1  #Change the line below to that it points to your shell
2  #!./msh
3  echo TESTINGTESTING123 >testfile.out
4  cat <testfile.out >testfile2.out
5  ls -l
6  #You should see testfile.out and testfile2.out both with size 18
7  exit 17
8  #After you invoke this script from the "real" shell, echo $? and
9  #verify the exit status is 17
10
```

Figure 2: Program Output After Executing test1.sh

Test Case #3: Error Handling

In this test case, we show the error handling capabilities of our shell program. When the shell tries to execute the lssssss command, it says that it was unable to do so since that command doesn't exist. By convention, when the shell is told to execute a command that doesn't exist, the child process running the command returns an exit status of 127. After that, we change the permissions of a file so that no one can read or read, write, or execute it. So when we try to write to it, our shell tells us that it was unable to open the file and has the child process give an exit status of 1.



```
fakharyarkhan@Shearyars-MBP ~ % cat testfile.out
cat: testfile.out: No such file or directory
fakharyarkhan@Shearyars-MBP ~ % cd Desktop/OS_HW_Problem_Set3
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % ./shell test2.sh
/private/tmp
/Users/fakharyarkhan
Child process 53974 exited normally
Real: 0.000494, User: 0.000851s,  Sys: 0.001715s
Execution of Command lssssssss Failed : No such file or directory
Child process 53975 exited with return val 127
Real: 0.000338, User: 0.000162s,  Sys: 0.000781s
Child process 53976 exited normally
Real: 0.000273, User: 0.000686s,  Sys: 0.000991s
Can't open file testfile.out : Permission denied
Child process 53977 exited with return val 1
Real: 0.000232, User: 0.000123s,  Sys: 0.000258s
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % echo $?
1
fakharyarkhan@Shearyars-MBP OS_HW_Problem_Set3 % cd
fakharyarkhan@Shearyars-MBP ~ % cat testfile.out
cat: testfile.out: Permission denied
fakharyarkhan@Shearyars-MBP ~ %
```

test2.sh

```
#Change the line below to that it points to your shell
#!./msh
#/tmp always exists on UNIX systems and is 777, so this will work
cd /tmp
pwd
#this should take you back to your homedir
cd
pwd


echo TESTINGTESTING123 >testfile.out
#The following is a command that presumably doesn't exist.  This er
#should not cause the entire script to end
lssssssss
chmod 000 testfile.out
#The I/O redirection should fail on the following
echo BLAH >>testfile.out
#After you invoke this script from the "real" shell, echo $?
#it should reflect the failure of the above I/O redirection which p
#the problem set spec sheet will be a 1 exit status
```

Figure 3: Program Output After Executing test2.sh