OS lab 1

Shell and System Calls

++++++++++++++++

Made By:
Mohamed Ramadan Mohamed
56

Problem Definition:

- Design a unix-like shell program.
- use system calls (fork() , execv()) to execute every command.
- the program should not crash at any circumstances.
- handle variables and some special command (cd , exit , history).

Code Organization:

- Program flow is organized into four stages:
  1- Taking input (user , file).
  2- handling input. Choosing wehther this is a special command , comment or a declaration.
  3- modifying parameters if it contains variables.
  4- executing commands and displaying results to user.

- Main Functions:
  1- *main():* the function that call all the underlying functions when needed.

  2- *dirtyWork():* takes one line of input and then process that line. It checks the input if it has a correct number of (“”) and then clears unnecessary spaces and then sends the command to *execute()*.

  3- *execute():* the core function of the program it takes the line after it's validated and divides it into arguments array and then checks if the command is special command or a declaration command or anything else. Also it sees if the command has '&' at its end and to run it on foreground or background.

  4- *executeDummy():* whether the code is run in background or foreground this function is called in both cases. It takes the command

and checks whether it has its path or it is in one of the paths defined in the $PATH variable and then sends the command to *execv()* by system call to execute it.

5- *executeCD():* handling the command 'CD' mainly by using *chdir()* call.

6- *executeHistory():* called whenever the command 'History' is called and used to display all the commands that the user made.

7- *terminate():* this is responsible for handling 'exit' command and is used to close the program.

8- *executeDeclaration():* this called whenever a declaration is made in format (\s+=\s+). it uses a vector-like data structure to keep the LHS and RHS. If the same LHS is entered no new entery is made and only RHS is changed.

9- *handleProcessTerminated():* this function is initiated by *signal()* and is called whenever a child process is terminated.

10- *initialize():* this function initializes multiple array and pointers that is used during the program execution.

11- *error_msg():* this function is the way every function display any error happened to it. It display the corresponding message to the errors declared in error_codes.h in *stderr* that all the functions use to declare error.

How to run the program:

this program runs only in unix-like systems because of system calls. So in this tutorial it's assumed that it will run on a unix-like system.

1- change directory to the program folder “tinyShell” using 'CD'.
2- type 'make' this will compile the code and make an executable file.
3 –
A- interactive mode:
    I. type ./tinyShell and the program will run in interactive mode.
b- batch mode:
    I. type ./tinyShell lab1_test.txt this will run the pre-made test file that was put on piazza.
    II. type ./tinyShell [complete file path] this will run the program in batch mode on this file.

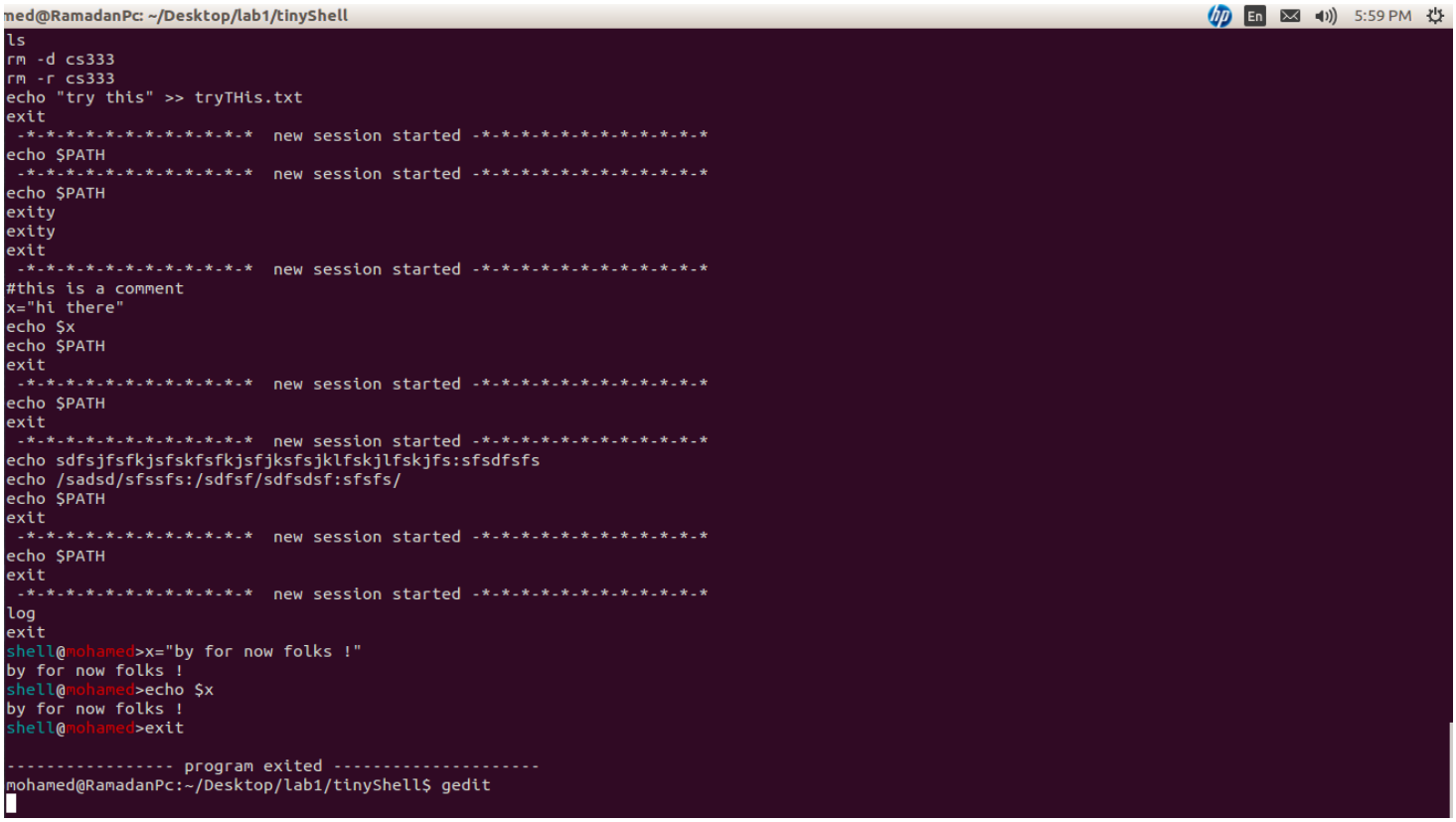
Sample Runs:

```
med@RamadanPc: ~/Desktop/lab1/tinyShell                                  5:54 PM
mohamed@RamadanPc:~$ #this is a sample run for my program
mohamed@RamadanPc:~$ cd Desktop/lab1/tinyShell/
mohamed@RamadanPc:~/Desktop/lab1/tinyShell$ make #this comiles the program
gcc main.o -o tinyShell
mohamed@RamadanPc:~/Desktop/lab1/tinyShell$ ./tinyShell #this runs in interactive mode
----------------- welcome to shell program --------------------
shell@mohamed>cd #go home
/home/mohamed
shell@mohamed>cd Desktop
/home/mohamed/Desktop
shell@mohamed>ls
file name with spaces  lab1  lab1-2015-10-10.zip  marble.c  marble.c~  sala7 el batee5a  Welad.Rezk.720p.HD.mkv
shell@mohamed>rm -r "file name with spaces"
shell@mohamed>ls
lab1  lab1-2015-10-10.zip  marble.c  marble.c~  sala7 el batee5a  Welad.Rezk.720p.HD.mkv
shell@mohamed>mkdir "file name with spaces 2"
shell@mohamed>cd "file name with spaces 2"
/home/mohamed/Desktop/file name with spaces 2
shell@mohamed>touch newFile.txt #makes an empty file
shell@mohamed>cd ../ #go back
/home/mohamed/Desktop
shell@mohamed># no we run the program in batch mode from the program itself
shell@mohamed>cd lab1/tinyShell #program path
/home/mohamed/Desktop/lab1/tinyShell
shell@mohamed>./tinyShell lab1_test.txt  #test file that was put on piazza
----------------- welcome to shell program --------------------
shell@mohamed>ls

bin                                                 history.txt     makefile~                 tinyShell.layout
err_codes.h                                         lab1_test.txt   obj                       TODO.c
_gitattributes                                      lab1_test.txt~  tinyShell                 Untitled 1.odt
_gitattributes~05377cbe764f912f39c20a752bffc79a199194c9  main.c          tinyShell.cbp
_gitignore                                          main.o          tinyShell[Conflict].layout
_gitignore~05377cbe764f912f39c20a752bffc79a199194c9  makefile        tinyShell.depend
shell@mohamed>ls -al

total 184
drwxrwxr-x 4 mohamed mohamed  4096 Oct 16 17:31 .
drwxrwxr-x 4 mohamed mohamed  4096 Oct 16 02:06 ..
drwxr-xr-x 3 mohamed mohamed  4096 Oct 10 22:34 bin
-rw-rw-r-- 1 mohamed mohamed   794 Oct 16 02:15 err_codes.h
```

....

```
med@RamadanPc: ~/Desktop/lab1/tinyShell                                  5:57 PM
shell@mohamed>#this is a comment

shell@mohamed>x=5

5
shell@mohamed>echo $x

5
shell@mohamed>y=5

5
shell@mohamed>z=$((x+y))

$((x+y))
shell@mohamed>echo z

z
shell@mohamed>echo "hello world"

hello world
shell@mohamed>echo                      "hello          world"

       hello          world
shell@mohamed>history

 -*-*-*-*-*-*-*-*-*-*-*-*  new session started -*-*-*-*-*-*-*-*-*-*-*-*
ls
ls -al
ls -a -l
/bin/ls -l -a
       /bin/ls     -l
vim
gedit &
ps
bin/ps
pwd
mkdir cs333
touch cs333/lab1.txt
cd ~/cs333
vi lab1.txt
pwd
```

....



the full sample run is in file : run_case1