Software installation and user guide

This doc contains information regarding how to build and run myshell program.

*************Description***********:**

C implementation of simple shell program.

*************what this program do**************

This software is a terminal (shell/bash) simulator with linux command ,

This shell print the user and the host in this format: user@current dir>

then waiting for command, and try to run the command that the user write.

*************Included Files: *************

My assignment solution consists of the following files:

- myshell.c
- makefile
- batch (a batch file)

************ How to Compile this program***************

in order to compile in linux needed to use gcc command

worthwhile compile by using the file "makefile" to compile all files.

using the command : "make"

*************Running my solution: *************

You can run my assignment as follows:

```
> make
```

This will create an executable file called myshell, and other files.

*************Compiler: *************

gcc version 4.6.3

Software installation and user guide

*************How to run this program*************

command line: "./<the name of the file that the makefile created>"

(in this case "./shell").

************Description of program files***************

-shell.c : hold the main and the functions

-makefile - this file compiling the file type .c and creating an "out" file that contain the program.

*************Instructions for cleaning executable: *************

Run "make clean" to delete the myshell executable file.

This will also delete the files created when it was first complied

*************Additional Tools Used: *************

Al libraries were on the Ubuntu installation at stat of the lab

************Description What input we needed***************

the input need to be real command.

(you can write not real command - the program print: "command not found").

for exit write the word quit or eof and press enter.

*************An explanation of the output we get**************

if the command not exist - the output will: "command not found".

else the output will the out that we given from the orginal sheel

for example - for the command: ls -l :

the orginal sheel display list of the files that there are in the folder

and this program display the same.

Software installation and user guide

*************basic commands**************

The following commands are Implemented using the execvp system call in child process except the 'cd' command which is implemented using the 'chdir' system call. These Programs are invoked by forking child process:

a) cd <Directory_name> :

- Changes the Directory, from the Current Directory to the Path Given as <Directory_name>.

- This command has been implemented using the chdir system call where <Directory_name> is passed to it as parameter.

b) ls :

List all the Files and Folders in the Present Current Directory.

The Function by which this command is implemented is 'executeCommand'

c) rm {-f,-v,-r} <file_names> :

- -f : Removes the files forcefully(-f)

- -r : Remove folders and their contents recursively (-r)

- -v : Remove files/folders with verbose option on, i.e. all deletion actions are explained.

- The Function in which this command is implemented is "executeCommand"

d) history :

- Prints the 10 recent commands issued by the user.

- To implement this command we store all the previously issued command in array of strings and print once the command(history) has been issued.

-We will overwrite oldest commands once all history list becomes full.

-The Function by which this command is implemented is 'history'

f) <program_name> :

- Creates a child process to run <program_name>.
- Supports the redirection operators '>' and '<' to redirect the input and output of the program to indicated files.

Ex - ls > out.txt, ./a.out < in.txt > out.txt, etc

- We have implemented this using **'dup'** and **'dup2**' commands.

g) exit :

-Exits the shell. This command is executed using the exit system call.

Ex - exit(0);

h) pwd :

- It prints current working directory of shell.

*************Advanced commands**************

a) cmd1 && cmd2 :

The "&&" operator is used to execute multiple commands in sequence, where the next command is only executed if the previous command completes successfully (returns an exit status of 0).

b) cmd1 || cmd2 :

The "||" operator is also used to execute multiple commands in sequence, but the next command is only executed if the previous command fails (returns a non-zero exit status).

C) cmd1 ; cmd2 :

The ";" operator is used to execute multiple commands in sequence, regardless of the success or failure of the previous command.

d) pipe :

The "pipe" operator is used to chain commands together and redirect the output of one command as the input of another command. The syntax for using a pipe is "command1 | command2".

For example, the command "ls | grep test" would list the contents of the current directory, and then only show the lines that contain the word "test".

e) output redirection :

The output redirection operator ">" is used to redirect the output of a command to a file. For example, the command "ls -l > file.txt" will redirect the output of the "ls -l" command to a file named "file.txt".

Note : To implement these operators, you will need to use the appropriate system calls and functions, such as fork(), execvp(), and wait() to create child processes, execute the commands, and wait for them to complete.

*************batch mode: *************

./myshell name_of_file_batch : will execute the commands in the batch file one by one, as if they were entered interactively by the user .

*************Conclusions/Remarks: *************

The system displays an error message on the output stderr and exits cleanly if :

- Bash file does not exist
- Inadequate number of arguments

The system displays an error message on the output stderr and continue execution if :

- A command does not exist or can not be executed