

Write - up Assignment-1 Part 2

by-Pranay Bhardwaj 2019263

Information on the Project Folder-

In folder-main.c it is the program for shell where main.out is the executable file. Now ls, date, mkdir, rm, cat are the binary file that can be made using Makefile also these binary are generated from ls.c, date.c, mkdir.c, rm.c, cat.c respectively.

main.c use the syscall execl() for calling the binaries that after fork().

Commands and Options-

1.cd(command)-

1.cd-It is the change directory command and used to change the current working directory.

I have implemented following options for cd as there are two options per command we have to implement-

1.cd-It changed the directory to home directory.

```
cd  
New Directory /home/pranay
```

2.cd / -It is used to change the directory to root directory.

```
cd /  
New Directory- /
```

3.cd "dir\name"- This command is used to move in the directory of name dirname.

```
cd Downloads  
New Directory-/home/pranay/Downloads
```

4.cd ..-This command is used to move in the parent directory of the current directory.

```
cd ..  
New Directory-/home/pranay
```

Options-

1-cd -P dir\name-This command is used to move into directory dir\name and all the symbolic links are being resolved.

```
cd -P Downloads  
New Directory-/home/pranay/Downloads
```

2-cd --help-This option prints the help documentation for cd.

```
cd --help  
cd: cd [-L|[-P [-e]] [-@]] [dir]  
    Change the shell working directory.  
  
    Change the current directory to DIR. The default DIR is the value of the  
    HOME shell variable.  
  
    The variable CDPATH defines the search path for the directory containing  
    DIR. Alternative directory names in CDPATH are separated by a colon (:).  
    A null directory name is the same as the current directory. If DIR begins  
    with a slash (/), then CDPATH is not used.
```

Error and corner Case -Handled-

In my case I handled errors like if a directory does not exist and if we enter that directory we will get an error that this directory does not exist. Also I handled corners in which if I enter folder name p\l and if it exists then we are moved into that directory. If there are multiple spaces between the text then it will be resolved in my case.

2.echo-

1.echo String hello- The echo commands print all the text that is written after it . In my case I implemented it.

```
echo Hello world  
Hello world
```

Options-

1.echo *-This options prints all the file and folder of the present directory except that they are not hidden i.e. not having dot as a character in the name.

```
echo *
Assignment1.2 2019263.zip "hello Report0_1_2019265 (1).pdf
Assignment0_1_2019263.zip labs.txt devcpp-4.9.9.2-src.zip h
LrWO6hx9NF0oJsf5lSmrdaMECVlOghxAnuRAM0muwyp41YNmZQH8J4pIxeYzz
P_4.c Makefile c2nasm-master a.out Document.txt k.c Quiz
```

2. `echo -n string`-This option is used to omit echoing trailing new line.

```
echo -n hello
hello
```

Error and Corner case(Assumptions)- In my case I assume that everything that is written behind `echo` is being printed in the new line. If there is an `\` in the string after `echo` it will automatically print without the `\` (backslash) `echo *` will print all the directory and files assuming that they are not hidden.

3. pwd

1. `pwd`-It prints the working directory starting from the root directory.

```
pwd
/home/pranay/Downloads/Assignment1.2
```

Options-

1. `pwd -P`-It prints the actual path.

```
pwd -P
/home/pranay/Downloads/Assignment1.2
```

2. `pwd --help`-It prints the help message for the `pwd` option.

```
pwd --help
pwd: pwd [-LP]
Print the name of the current working directory.
Options:
-L      print the value of $PWD if it names the current working
        directory
-P      print the physical directory, without any symbolic links
By default, 'pwd' behaves as if '-L' were specified.
Exit Status:
```

4.exit-

1. `exit` - It is used to exit the shell where it is currently running. This option is simple and in this option shell exit without returning a status.

```
exit  
Thanks for using the shell
```

2.exit number-It is used to exit with an exit status that is returned when the shell exits.

```
exit 102  
Thanks for using the shell
```

Options-

Currently there is only one option in exit which is --help.

1.exit --help-It displays the help information for the exit command.

```
exit --help  
exit: exit [n]  
Exit the shell.  
Exits the shell with a status of N. If N is omitted, the exit status  
is that of the last command executed.
```

5.history-

1.history- It prints all the commands that are being entered in the shell before clearing it.

```
history  
1 cd  
2 cd  
3 cd --help  
4  
5 cd -P Downloads  
6 history --help  
7 cd /
```

Options-

1.history -c -It clears all the history and the history file is then empty.

```
history -c  
history  
1 history -c
```

2.history --help-It prints the document regarding the help for history command.

```

history --help
history: history [-c] [-d offset] [n] or history -anrw [filename] or history -ps arg [arg...]
    Display or manipulate the history list.

    Display the history list with line numbers, prefixing each modified
    entry with a '*'. An argument of N lists only the last N entries.

    Options:
    -c      clear the history list by deleting all of the entries
    -d offset delete the history entry at position OFFSET. Negative
            offsets count back from the end of the history list

```

Error handling and assumptions-

As we know there are 2 options implemented in this mini version of this shell so if there is option rather than -c and --help all are giving that you have entered the wrong input please check. I save history in the txt file named history.txt and whenever a command is entered it appends at the end of the history file.

All the preceding commands are external and are executed by calling fork() sys call and executed using execl() syscall.

6.mkdir-

1.mkdir [Directory]- This option creates a directory of the name that you have entered after the term mkdir.

```
mkdir a b c
```

In my version the user can create three directories in a simple command .

Options-

1.mkdir -v [directories]- Using these commands the mkdir will notify that such directory is created.

```

mkdir -v e f
mkdir: created directory 'e'
mkdir: created directory 'f'

```

2. mkdir -p [directories]- This command creates all the directories that are entered if a directory is entered and if it's parent directory is not created then it will automatically be created.

```
mkdir -p one/two/three
```

Error and Assumption-

For instance there is a directory entered such that it's parent does not exist then it will give an error that this particular parent directory does not exist same as the bash mkdir.

7.date

date-It prints the current date, time and day in local time.

```
date
Wed Sep 30 23:14:40 2020
```

Options-

1.date -u-It prints the current date, time and day in GMT.

```
date -u
Wed Sep 30 17:44:44 2020
```

2. date -r file-It prints the last modified date of the file that we have entered.

```
date -r main.c
Wed Sep 30 21:27:30 2020
```

Error and Assumptions-

In this basically if there is a file which does not exist and we entered its name then an error message will arrive that this file does not exist also if we entered invalid input after the history command then it will give a message that this command is invalid.

8-cat-

1.cat file1 file2- This command is used to print the file in the bash and it can handle multiple files.

Options-

1-cat -n file-It prints the file with numbers at the starting of the file.

2-cat -E file -It prints the whole file with \$ at the end of the file.

Error and Assumptions- If there is file that does not exist then it will give an error, / are handled and it reads without it also if there is a

command entered that has the wrong syntax the shell will simply give an error and it handles error according to the situation.

9-rm

rm file -This command removes the file and empty folder and it can delete multiple folders.

Options-

1-rm -i file-It removes the file and before deleting it, it asks the users to delete it before deleting if the user entered y then the file will be deleted otherwise file will not be deleted.

```
rm -i b c
rm: remove regular file b?
y
rm: remove regular file c?
y
```

2. rm --help- It displays the help document for the rm command.

```
rm --help
Usage: rm [OPTION]... [FILE]...
Remove (unlink) the FILE(s).

  -f, --force          ignore nonexistent files and arguments, never prompt
  -i                  prompt before every removal
  -I                  prompt once before removing more than three files, or
                     when removing recursively; less intrusive than -i,
                     while still giving protection against most mistakes
  --interactive[=WHEN] prompt according to WHEN: never, once (-I), or
                     always (-i); without WHEN, prompt always
  --one-file-system    when removing a hierarchy recursively, skip any
                     directory that is on a file system different from
                     that of the corresponding command line argument
  --no-preserve-root   do not treat '/' specially
  --preserve-root[=all] do not remove '/' (default);
                     with 'all', reject any command line argument
                     on a separate device from its parent
  -r, -R, --recursive remove directories and their contents recursively
  -d, --dir            remove empty directories
  -v, --verbose        explain what is being done
  --help              display this help and exit
  --version            output version information and exit
```

Error and Assumptions-

If the format other than the specified format will be entered then it will give an error also if a file does not exist and we give command to

remove it the shell will give error that such file does not exist also if we try to remove a folder which is not empty then also it will give an error.

10-ls

1.ls - It will give all the files and directories present in the current bash directory that are not being hidden.

```
ls
date ls.c ls cat.c cdhelp.txt Makefile main
rm history.txt Shell.c date.c main.out mkdir.c historyhelp.txt
mkdir cat rm.c main.c rmhelp
```

Options-

1ls -a- It will print all the directories including the hidden folders i.e. whose name is starting with .(dot)

```
ls -a
.. date ls.c ls cat.c cdhelp.txt Makefile
main rm history.txt Shell.c . date.c main.out
mkdir.c historyhelp.txt mkdir cat rm.c main.c rmhelp
```


2 ls -1-This option prints the file in order such that every file printed is in the new line.

```
ls -1
date
ls.c
ls
cat.c
cdhelp.txt
Makefile
main
rm
history.txt
Shell.c
date.c
main.out
mkdir.c
historyhelp.txt
mkdir
cat
rm.c
main.c
rmhelp
```


Assumptions and error handling- In the ls command it prints the file and folder in the present working directory so there will be no error unless there is a mistake in the syntax otherwise there will be no error.

At last if there is command use despite these 10 commands then bash will show some error.

Some test cases



```
pranav@pranav-Vostro-3470: ~/Downloads/Assignment1.2
1 history --help
2 history -hsbhd
3
4 cd
5 cd Downloads
6 mkdir a b c
7 mkdir -p a b
8 mkdir -p a b
9 mkdir -p a b
10 mkdir -p one/two/three
11 date
12 date -u
13 date -u
14 pwd
15 cd Assignment1.2
16 date -r main.c
17 date -r /dev/urandom
18
19 cat main.c
20 cat hola
21 cat -n main.c
22 od -t
23 rm -f
24 rm -rf a
25 rm --help
26
27 ls
28 cd Assignment1.2
29 ls
30 ls -l
31 ls -l
32 cd ..
33 echo hello world
34 rm abcd
35 cd abcd
36 date
37 date -u
38 ls
39 date -r 201901.25p
40 cat k.c
```

[illegible]