Computer Architecture and Operating Systems

Assignment 0 (Part 2)

Name: Anmol Gupta

Roll Number: 2018329

Internal Commands

The system handles 5 internal commands:

1. `cd`
2. `echo`
3. `history`
4. `pwd`
5. `exit`

In the `main` file, a function has been created to handle each of these commands using C functions. No child process has been created, because we want the change to be reflected in the current process, which won't happen if we `fork` it.

1. cd

The command changes directory.

Test cases

```
cd dir/
cd ..
cd .
cd
```

Error handling and corner cases:

Works with relative paths such as `../dir`.

Extra spaces between the arguments are removed.

If the user tries to enter into a directory that doesn't exist then the shell outputs `No such file or directory` message.

Entering `cd` without any other arguments will take you to the home directory, thus replicating the behaviour of a real shell.

2. echo

The command writes arguments to the standard output.

Test cases

```
echo hello world
echo -n hello world
echo -E hello world
echo “hello world”
```

Error handling and corner cases:

If the arguments are provided inside double quotes, the quotes are removed.

If multiple spaces are provided between the arguments, only a single space is shown in the output.

3. history

The command displays the history of commands entered into the shell.

Test cases

```
history
history -c
history 5
```

Error handling and corner cases:

Even invalid commands are added to the history.

Extra spaces between the arguments are removed.

The command `history x` displays the last `x` commands, and if `x` is more than the total number of commands that have been entered then the complete history is displayed.

The history is stored in a file, so even after exiting a session of the shell, it is not lost.

4. pwd

The command returns working directory name.

Test case

```
pwd
```

5. exit

The command exits the shell.

Test case

```
exit
```

Assumptions

Only one option will be provided at a time, e.g., commands like `echo -nE` or `echo -n -E` won't be entered.

In `echo`, multiple backslashes are not entered.

External Commands

The system handles 5 external commands:

1. `ls`
2. `cat`
3. `date`
4. `rm`
5. `mkdir`

A different `.c` file has been created for each of these commands. After creating a child process in the `main` file, the function in the respective `.c` file is called from the `main` file. The parent process waits for the child process to kill itself after running the `execl` command, and then proceeds with its own execution.

1. ls

The command lists all directory contents.

Test cases

```
ls -l
ls -a
ls -al
```

2. cat

The command concatenates and prints files.

Test cases

```
cat filename.txt
cat -n filename.txt
cat -b filename.txt
```

3. date

The command displays or sets date and time.

Test cases

```
date
date -u
date -r 9000
```

4. rm

The command removes directory entries.

Test cases

```
rm filename
rm -r dir/
```

5. mkdir

The command makes directories.

Test cases

```
mkdir dir
mkdir -p /dir1/dir2/
```

Assumptions

The shell will be provided with either

1. Zero options, e.g., `ls`
2. One option, e.g., `ls -a`
3. Two options, combined together, e.g., `ls -al`, or given separately `ls -a -l`
4. More than two options, but only combined together, e.g. `ls -alR`

In `cat` and `rm` only one file at a time will be provided.

Error handling and corner cases

Extra spaces between the arguments are removed.

The user can press enter many times, to create space after the last output.

Both options can be entered together, either combined or separately, e.g. `ls -al` and `ls -a -l` both will work.

The shell outputs `command not found` when an invalid command is entered.