**OS ASSIGNMENT 1**

The shell is run when you type in make. It takes one command at a time and after it’s execution, it exits the shell and thanks the user for using the shell.

One common assumption I have made for almost all commands is that for the flags only one letter implementation is allowed, i.e., -n would be valid but –nn wouldn’t be. If this statement is not followed it will be explicitly mentioned.

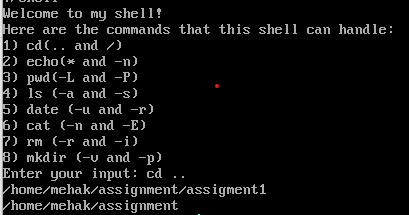
The shell file made can handle the following commands:

1. cd

The implementations I chose for this were ‘cd..’ and ‘cd /’

To implement this in my main shell file, I used chdir().

Since this command just changes the directory in the process itself and not the shell, to show that the command is working, I am printing the working directory before and after the cd command.





In my shell, I am assuming that only ‘cd ..‘ and ‘cd /’ are not valid commands.

Corner cases / Errors handled:

To prevent exhaustive checking, only two cd conditions have been made valid. It also checks for wrong syntax and if the path doesn’t exist, the program terminates.

1. echo

The implementations I chose for this were ‘echo \*’ and ‘echo –n’

For \*, I used the system command to call the ls executable that I had written for my external commands.





This command is fairly easy to execute and one just needs to type in echo and the text.

We have assumed here that only echo –n is a valid command and the number of n’s is limited to one, unlike the linux terminal.

Corner cases/Errors handled:

There are no significant errors to be handled for this as even if the syntax is wrong it just disregards the wrong syntax and considers echo without any flags.



1. Pwd

The implementations I chose for this were ‘pwd –L’ and ‘pwd –P’.

In my shell I have assumed that the flags have only one letter implementation allowed ,i.e, pwd –LL would not be a valid command.

Corner cases/Errors handled:

Syntax errors are handled and terminate the program immediately.



1. ls

The implementations I chose for this were ‘ls –a’ and ‘ls – s’.

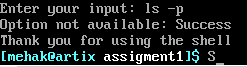
These were implemented using the dirent.h library in c.

Bugs/Corner Cases/Errors:

One major issue that came early on was that for the block size, my implementation using st\_blocks gave me an answer in the 512-byte blocks. The linux terminal uses 1024 byte-blocks for it’s ls-s implementation. Hence, my value from st\_blocks had to be divided by 2 to give the correct value.[[1]](#footnote-1)

At the same time this function also checks for the ENOENT error which and also checks for wrong syntax.





1. rm

The implementations I chose for this were ‘rm –i’ and ‘rm –r’.

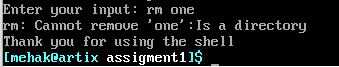
These were implemented by using the unistd.h library in c.

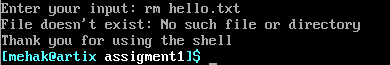
I am assuming that this can only take one file/directory as an input in one go.

Bugs/Corner cases/errors:

This command checks whether the file we are entering exists or not. In the rm implementation with the r flag, it is also checked whether the input is a directory name or not. If it is the program is terminated.

The syntax is also checked and on entering a wrong option , the shell program terminated by calling exit(EXIT\_FAILURE) and printing an error message.





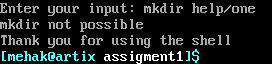
1. mkdir

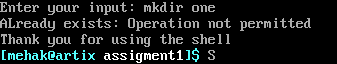
The implementations I chose for this were ‘mkdir –v’ and ‘mkdir –p’.

I am assuming that for the –p flag, only three directories can be entered max. Hence even if four directories are entered, only the first three will be assumed and the directory will be made.

Bugs/Corner Cases/Errors:

It checks for the error EEXIST and also checks for wrong syntax. If the path is also wrong, it throws an error.





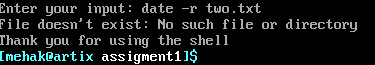
1. date

The implementations I chose for this were ‘date –u’ and ‘date –r’.

I used strftime() to get the time string. Format specifiers were used for each individual aspect of the date.

Errors:

This checks for whether the file/directory exists in –r input and accordingly terminates the program. It also checks for wrong syntax.



1. cat

The implementations I chose for this were ‘cat –n’ and ‘cat –e or cat –E’.

Here –nn and other such options are a valid input.

The cat stdin implementation is also present here. To stop this implementation, ctrl+c has to be used on the keyboard to interrupt the input flow.

This was implemented using fprintf() and other such functions.

I am assuming here that the cat function can only take two files max as an input.

Bugs/Errors/Corner Cases:

This checks if all the files exists or not. It also checks for wrong syntax. A bug I had in this code was in my while loop. The while loop for stdin function would require the cf to be equal to the argc, however for the files, this led to an extra iteration which led to a seg fault. Hence, for the non stdin implementation, when the cf is incremented, the condition whether cf==argc is checked. If found true, the program terminated hence preventing the seg fault.

1. <https://askubuntu.com/questions/1252657/why-is-total-stat-block-size-of-a-directory-twice-of-ls-block-size-of-a-director> [↑](#footnote-ref-1)