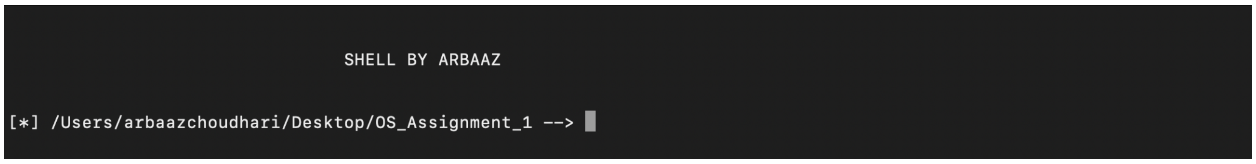
# Documentation for Shell created by Arbaaz Choudhari

Roll no: 2021034

The shell should start using the following commands:
1. make
2. ./shell



```
                              SHELL BY ARBAAZ

[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> 
```

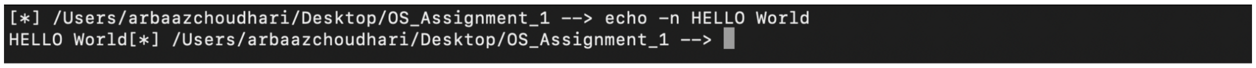
Displays the current directory path before input.

---

## 1. Command: echo

Options that the command takes: “-n” and “--help”

1. echo -n {sample text}  → Does not print the trailing new character.

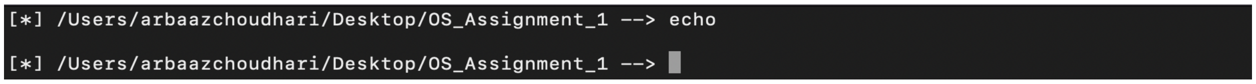


```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> echo -n HELLO World
HELLO World[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> 
```

2. echo –help → Displays help text about echo command.

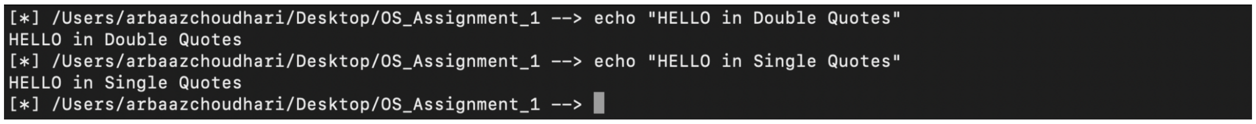
Corner Cases handled:

1. No arguments with echo:



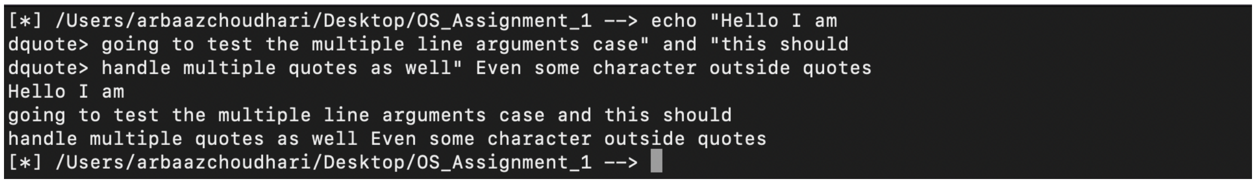
```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> echo

[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> 
```

2. Use of single and double quotes:
   Omits printing the quotations as characters.



```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> echo "HELLO in Double Quotes"
HELLO in Double Quotes
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> echo "HELLO in Single Quotes"
HELLO in Single Quotes
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> 
```

3. Multiple Line arguments while using Quotes:
   When using quotes if the user starts a quotes and does not end it before pressing enter the shell takes the newline character as a character to print in the string and asks user to dquote their arguments for printing.



```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> echo "Hello I am
dquote> going to test the multiple line arguments case" and "this should
dquote> handle multiple quotes as well" Even some character outside quotes
Hello I am
going to test the multiple line arguments case and this should
handle multiple quotes as well Even some character outside quotes
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 -->
```

<u>Assumptions made</u>:
	I have not made any specific assumptions for echo. It should handle any general case with the given options.

<u>Test Cases</u>:

1. echo Hello
2. echo “HELLO”
3. echo “hel
lo”
4. echo Hello World
(Also see the Corner cases)

---

## 2. Command: pwd

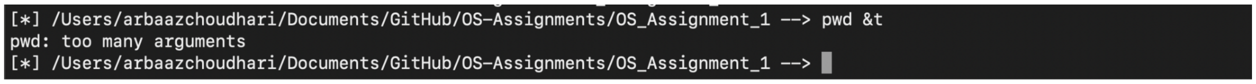
<u>Options that the command takes</u>: “-L” and “-P”

1. pwd -L → Display the logical current working directory.

2. pwd -P → Display the physical current working directory (all symbolic links resolved).

Corner Cases Handled:

1. Displays error when an invalid argument is entered. (pwd should not get any argument).



```
[*] /Users/arbaazchoudhari/Documents/GitHub/OS-Assignments/OS_Assignment_1 --> pwd &t
pwd: too many arguments
[*] /Users/arbaazchoudhari/Documents/GitHub/OS-Assignments/OS_Assignment_1 -->
```

2. Displays error for invalid cases such as following:
   The process did not have read or search permission on some component of the working directory's path name.
   *size* is less than or equal to zero.
   A component of the working directory's path name does not exist.

   To check use following testcase:
   mkdir RandomDirectory
   (Delete this directory manually using File Explorer)
   pwd

Assumptions made:

No specific assumptions.

Test Cases:
pwd
(Also look at corner cases)

---

## 3. Command: cd

Options that the command takes: “-L” and “-P”

1. cd {directory path} -L → Force symbolic links to be followed. In other words, if you tell cd to move into a "directory", which is actually a symbolic link to a directory, it moves into the directory the symbolic link points to.

2. cd {directory path} -P → Use the physical directory structure without following symbolic links. In other words, only change into the specified directory if it actually exists as named; symbolic links are not followed.

Corner Cases Handled:

1. Displays error when directory does not exist.

2. cd . → change directory to the current directory.

3. cd .. → changes directory to the parent directory of the current directory.

4. cd / → changes directory to the root directory.

Assumptions made:
No specific Assumptions.

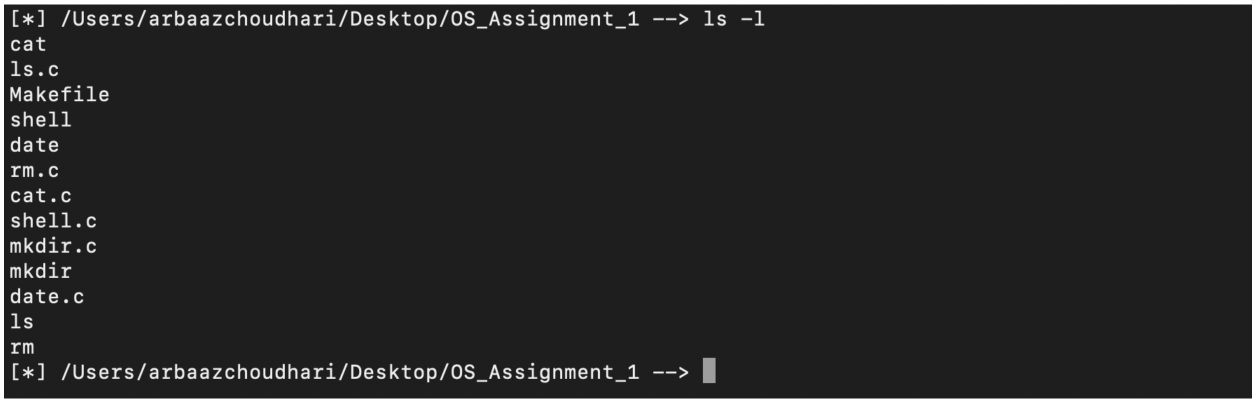
Test Cases:
cd ..
cd OS_Assignment_1
(Also look at Options and Corner Cases)

---

## 4. Command: ls

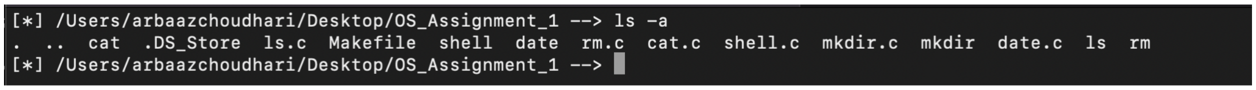
Options that the command takes: “-l” and “-a”

1. ls -l → Lists the files one after another in new lines.



```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> ls -l
cat
ls.c
Makefile
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 -->
```

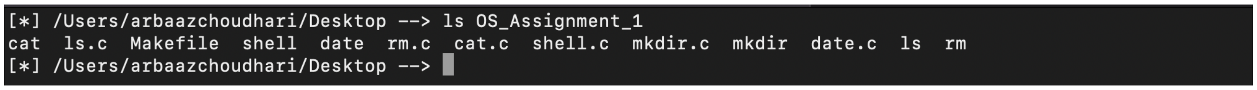
2. ls -a → Include directory entries whose names begin with a dot (‘.’).(Hidden directories)



```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> ls -a
.  ..  cat  .DS_Store  ls.c  Makefile  shell  date  rm.c  cat.c  shell.c  mkdir.c  mkdir  date.c  ls  rm
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 -->
```

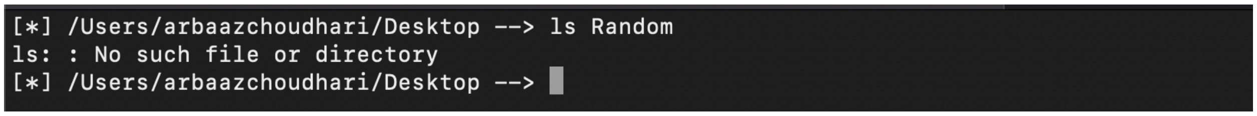
Corner Cases Handled:

1. ls {directory path}
   This helps to use ‘ls’ on directories other than current directory.



```
[*] /Users/arbaazchoudhari/Desktop --> ls OS_Assignment_1
cat  ls.c  Makefile  shell  date  rm.c  cat.c  shell.c  mkdir.c  mkdir  date.c  ls  rm
[*] /Users/arbaazchoudhari/Desktop -->
```

2. Displays error when directory does not exist.



```
[*] /Users/arbaazchoudhari/Desktop --> ls Random
ls: : No such file or directory
[*] /Users/arbaazchoudhari/Desktop -->
```

3. You can use flags while using ls {directory name}

Assumptions made:

Cannot use both the flags simultaneously.

Test Cases:

ls

ls -l

ls -a
cd ..
ls OS_Assignment_1
ls -l OS_Assignment_1

---

## 5. Command: cat

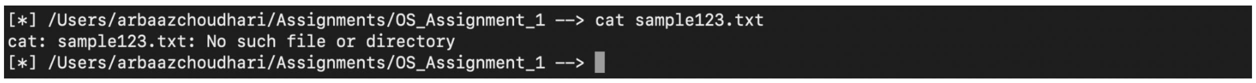
Options that the command takes: “-n” and “-b”

1. cat -n {filename} → Number the output lines, starting at 1.

2. cat -b {filename} → Number only the non-blank output lines, starting at 1.

Corner Cases Handled:

1. Displays error when file does not exist.



```
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> cat sample123.txt
cat: sample123.txt: No such file or directory
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 -->
```

2. Multiple file arguments handled.
   Eg. Sample test case - cat sample.txt sample1.txt sample2.txt (concatenates the text in all three files and displays).

3. While handling multiple files, the shell can handle the case where some files exist while the others don’t.

4. You can use flags while having multiple file arguments.

Assumptions made:

No specific assumptions for this command.

Test Cases:
cat sample.txt
cat sample.txt sample1.txt
cat non_existing_file.txt
cat -n sample.txt
cat -b sample.txt sample2.txt

---

## 6. Command: date

Options that the command takes: “-r” , “-u” , “+%{format-option}”

1. date -r {filename} → Print the date and time of the last modification of file.

2. date -u → Display the time in UTC (Coordinated Universal) time zone.

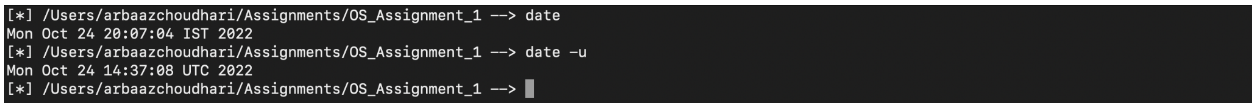
3. date “+%format”
   %D: Display date as mm/dd/yy.
   %d: Display the day of the month (01 to 31).
   %a: Displays the abbreviated name for weekdays (Sun to Sat).
   %A: Displays full weekdays (Sunday to Saturday).
   %h: Displays abbreviated month name (Jan to Dec).
   %b: Displays abbreviated month name (Jan to Dec).
   %B: Displays full month name(January to December).
   %m: Displays the month of year (01 to 12).
   %y: Displays last two digits of the year(00 to 99).
   %Y: Display four-digit year.
   %T: Display the time in 24 hour format as HH:MM:SS.
   %H: Display the hour.
   %M: Display the minute.
   %S: Display the seconds.

Test Cases for this option:
date "+%Y-%m-%d"
date “+%S”

Corner Cases Handled:

1. While using the ‘-r’ flag, if a file doesn’t exist the shell displays an error.

2. The outputs of ‘date’ and ‘date -u’ are distinguished by specifying the time zone in the output.



```
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> date
Mon Oct 24 20:07:04 IST 2022
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> date -u
Mon Oct 24 14:37:08 UTC 2022
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 -->
```

Assumptions made:
No specific assumptions for this command.

Test Cases:
date
date -u
date -r sample.txt
(Also see the options)

---

## 7. Command: rm

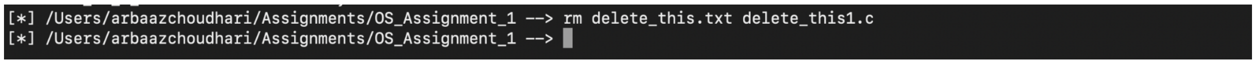
Options that the command takes: “-v” and “-i”

1. rm -v {filename} → Be verbose when deleting files, showing them as they are removed.

2. rm -i {filename} → Request confirmation before attempting to remove each file.

Corner Cases Handled:

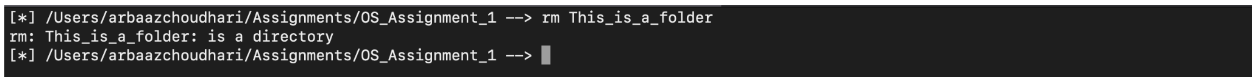
1. Multiple file arguments handled.



```
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> rm delete_this.txt delete_this1.c
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 -->
```

2. Flags can be used while having multiple file arguments.

3. Does not execute command and displays error when directory name is given instead file.



```
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> rm This_is_a_folder
rm: This_is_a_folder: is a directory
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 -->
```

Assumptions made:

No specific assumptions for this command.

Test Cases:
rm delete_this.txt
rm -v delete_this1.txt
rm -i delete_this.txt
(Also see the options and corner cases handled.)

---

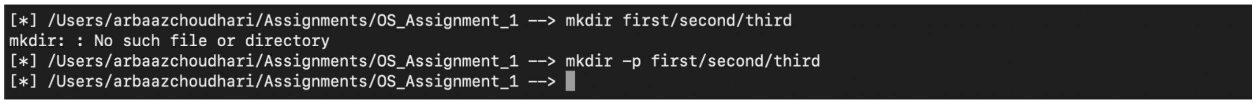
## 8. Command: mkdir

Options that the command takes: “-p” and “-v”

1. mkdir -p {directory-name} → Creates parent directories as necessary. This command creates intermediate directories as required. If this option is not specified, the full path prefix of each operand must already exist. With this option specified, no error will be reported if a

directory given as an operand already exists. If the directory does not exist then it creates all the intermediate directories required.

Example for this option:



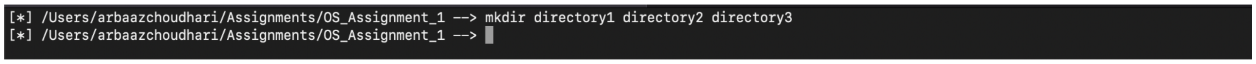
```
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> mkdir first/second/third
mkdir: : No such file or directory
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> mkdir -p first/second/third
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> 
```

If the first and second directories do not exist, due to the -p option, mkdir will create these directories for us. If we do not specify the -p option, and request the creation of directories, where parent directory doesn't exist, we will get an error message.

2. mkdir -v {directory-name} → Be verbose when creating directories, listing them as they are created.

<u>Corner Cases Handled</u>:

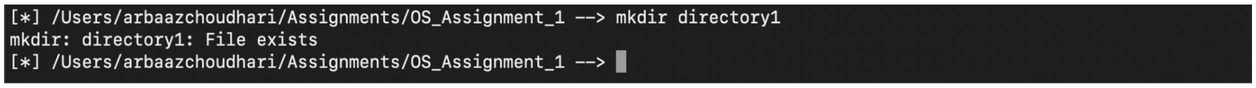
1. Multiple directory arguments handled.



```
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> mkdir directory1 directory2 directory3
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> 
```

2. The shell allows the use of flags while giving multiple directory arguments.

3. An error is displayed if a directory with the given name already exists.



```
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> mkdir directory1
mkdir: directory1: File exists
[*] /Users/arbaazchoudhari/Assignments/OS_Assignment_1 --> 
```

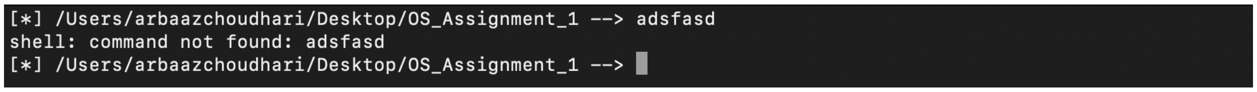
<u>Assumptions made</u>:

The flags cannot be used simultaneously.

<u>Test Cases</u>:
mkdir Home
mkdir -v Folder
Mkdir -p MilkyWay/SolarSystem/Earth
(Also see corner cases)

## General Corner Cases:

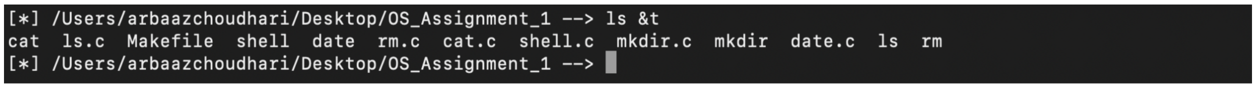
1. For not allowed commands:



```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> adsfasd
shell: command not found: adsfasd
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> 
```

## Thread Based Execution:

For thread based execution, you have to add “&t” at the end of the command for ‘ls’, ‘cat’, ‘date’, ‘rm’ and ‘mkdir’.



```
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> ls &t
cat  ls.c  Makefile  shell  date  rm.c  cat.c  shell.c  mkdir.c  mkdir  date.c  ls  rm
[*] /Users/arbaazchoudhari/Desktop/OS_Assignment_1 --> 
```

## Sample Test Cases for the entire program:

echo hello world
echo “This is a multi-line
input for echo”
pwd
cd ..
ls -l &t
cd OS_Assignment_1
cat -b textfile1.txt textfile2.txt
date
date -r textfile1.txt &t
rm -i textfile1.txt textfile2.txt
mkdir -p eFolder1/Folder2/Folder3
rm Folder1 &t

References:

https://www.computerhope.com/unix/ucd.htm

https://www.geeksforgeeks.org/mkdir-command-in-linux-with-examples/