PART 2 DESCRIPTION

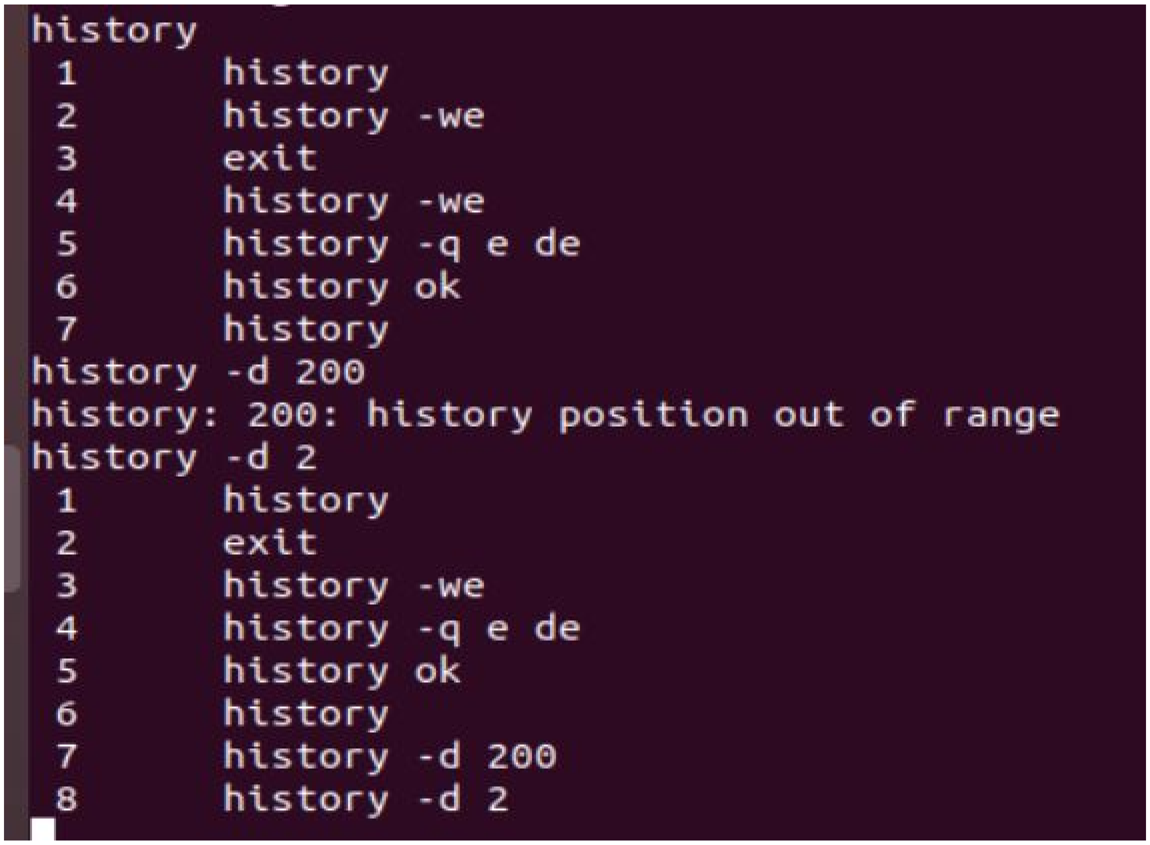
1. **Internal commands**

   a) **history →**
   - **This command is used to print the history of commands written by the user.**
   - **My code prints all the past commands written by the user on the terminal until he clears it using -c flag.**
   - **I am storing the entire history on a txt file and appending it in every instance of command entered by the user.**

   Flags used -

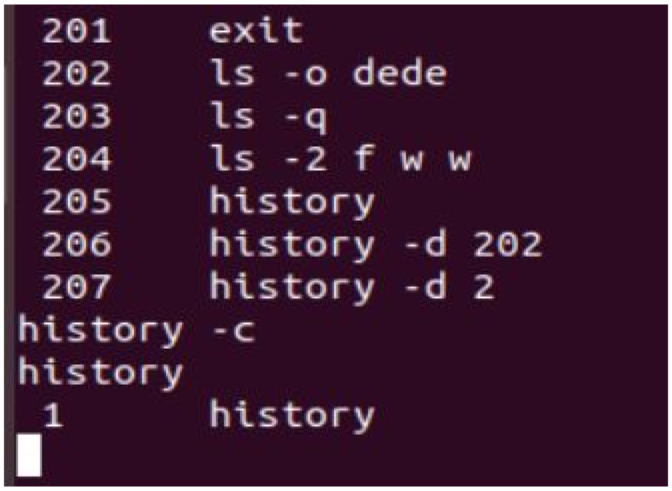
   1) **‘-c’**
      - **This flag clears the history from the text file.**
      - **When this flag is invoked I open the file in “w” mode due to which it creates a new file and overwrites data on it. Hence this flag clears the previous history.**
   2) **‘-d’**
      - **This flag is used to delete a particular line of history.**
      - **We have to specify a number beside it to clear that history line.**
      - **In my code I am creating a next text file and writing the lines of the history text file onto it except the deleted command. Then I am renaming the new file and deleting the old one.**
   3) **history**
      - **This command is used to print the entire history stored in the history text file.**

Test cases



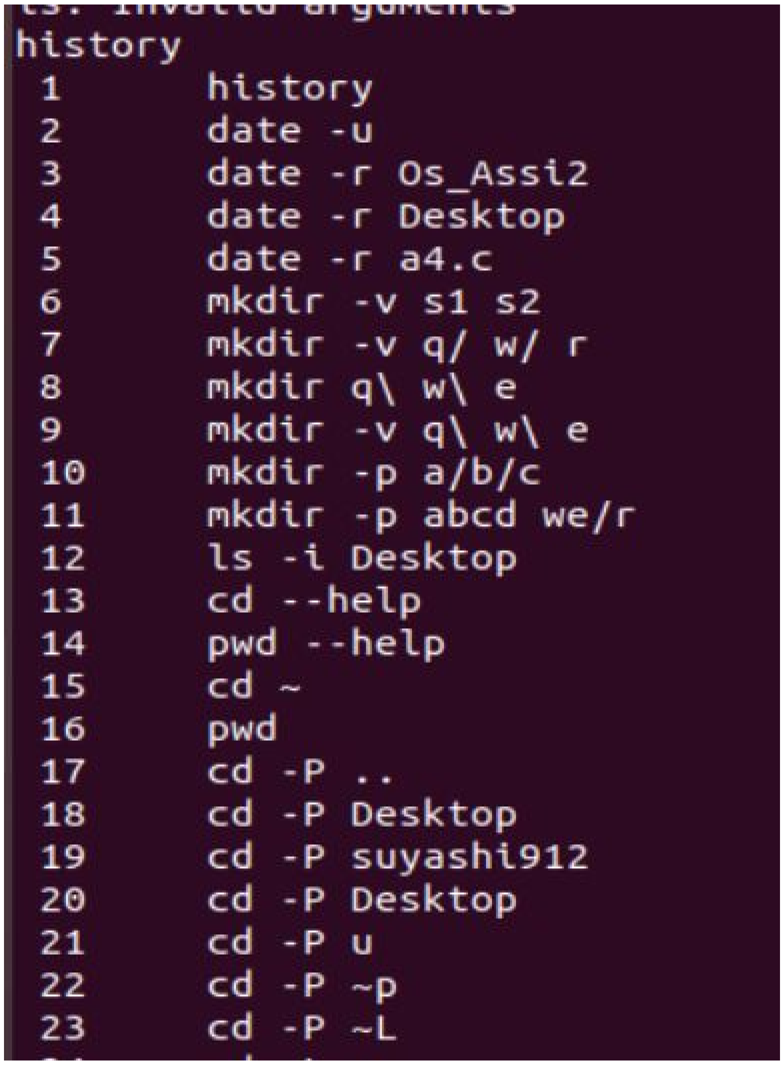
```
history
 1      history
 2      history -we
 3      exit
 4      history -we
 5      history -q e de
 6      history ok
 7      history
history -d 200
history: 200: history position out of range
history -d 2
 1      history
 2      exit
 3      history -we
 4      history -q e de
 5      history ok
 6      history
 7      history -d 200
 8      history -d 2
```

The -d flag above deletes the line at the second position in history.



```
 201    exit
 202    ls -o dede
 203    ls -q
 204    ls -2 f w w
 205    history
 206    history -d 202
 207    history -d 2
history -c
history
 1      history
```

The -c flag clears the entire history.

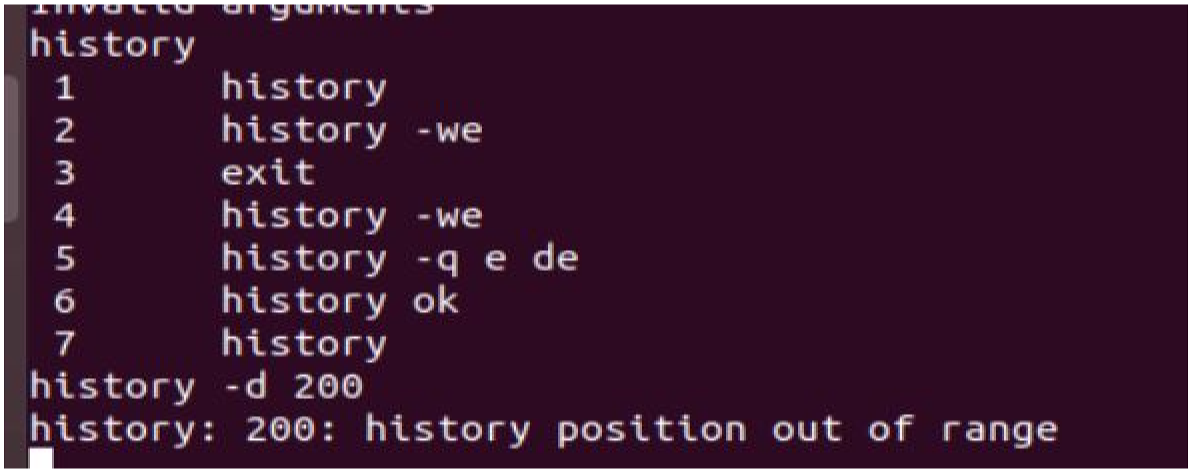


```
history
 1      history
 2      date -u
 3      date -r Os_Assi2
 4      date -r Desktop
 5      date -r a4.c
 6      mkdir -v s1 s2
 7      mkdir -v q/ w/ r
 8      mkdir q\ w\ e
 9      mkdir -v q\ w\ e
 10     mkdir -p a/b/c
 11     mkdir -p abcd we/r
 12     ls -i Desktop
 13     cd --help
 14     pwd --help
 15     cd ~
 16     pwd
 17     cd -P ..
 18     cd -P Desktop
 19     cd -P suyashi912
 20     cd -P Desktop
 21     cd -P u
 22     cd -P ~p
 23     cd -P ~L
```

History command displays the entire history stored in the file.

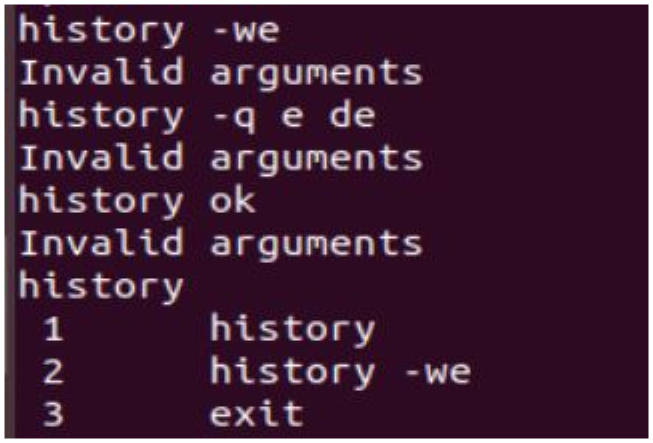
Error handling and corner cases

1) If the line number to be deleted is more than than the total lines in history, then an error message is printed using the strerror() function.



```
history
 1      history
 2      history -we
 3      exit
 4      history -we
 5      history -q e de
 6      history ok
 7      history
history -d 200
history: 200: history position out of range
```

2) **In case the correct arguments / correct number of arguments aren't entered by the user then it displays an error message of invalid arguments.**



```
history -we
Invalid arguments
history -q e de
Invalid arguments
history ok
Invalid arguments
history
 1      history
 2      history -we
 3      exit
```

b) **pwd -**

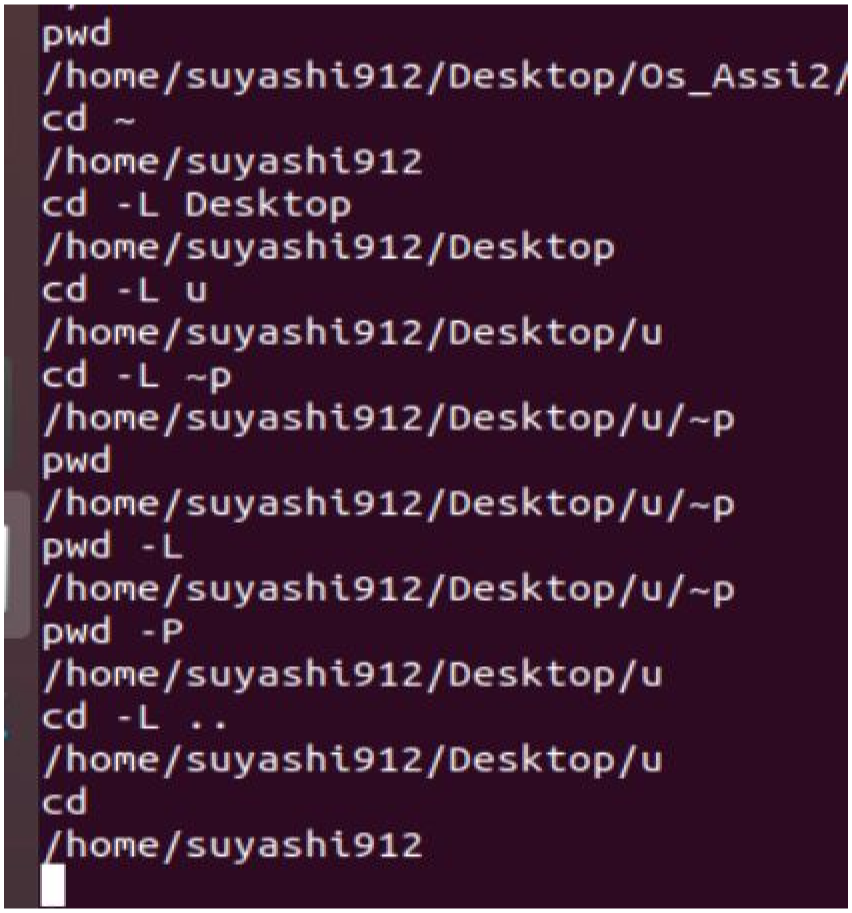
This command is used to print the current directory in which the user is working at the moment.

Flags used

1) **'-P' -**
    - **This is used to display the physical path ignoring any symlinks present in it. It avoids the links.**
    - **In my code I use getcwd() function to get the physical path of the current working directory.**
2) **'-L' -**
    - **It does not remove the symlinks. It displays the logical path of the current working directory.**
    - **In my code I have stored the logical path I got from cwd in a temporary string and I display it whenever this flag is invoked.**
3) **pwd - it functions the same as -L flag**
4) **'--help'**

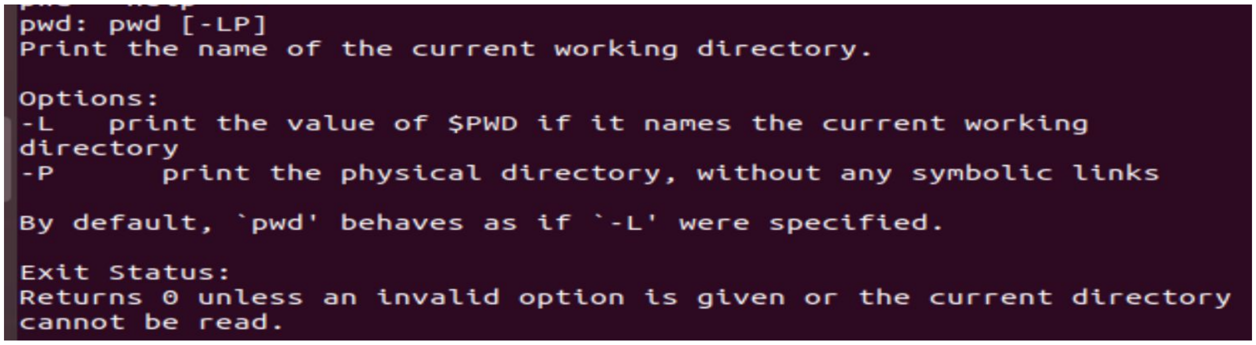
   This flag is used to display the manpages of pwd.

Test Cases



```
pwd
/home/suyashi912/Desktop/Os_Assi2/
cd ~
/home/suyashi912
cd -L Desktop
/home/suyashi912/Desktop
cd -L u
/home/suyashi912/Desktop/u
cd -L ~p
/home/suyashi912/Desktop/u/~p
pwd
/home/suyashi912/Desktop/u/~p
pwd -L
/home/suyashi912/Desktop/u/~p
pwd -P
/home/suyashi912/Desktop/u
cd -L ..
/home/suyashi912/Desktop/u
cd
/home/suyashi912
```

Here we can see that pwd -L prints the logical path i.e the one with the symlink '~p' while -P just displays the logical path.

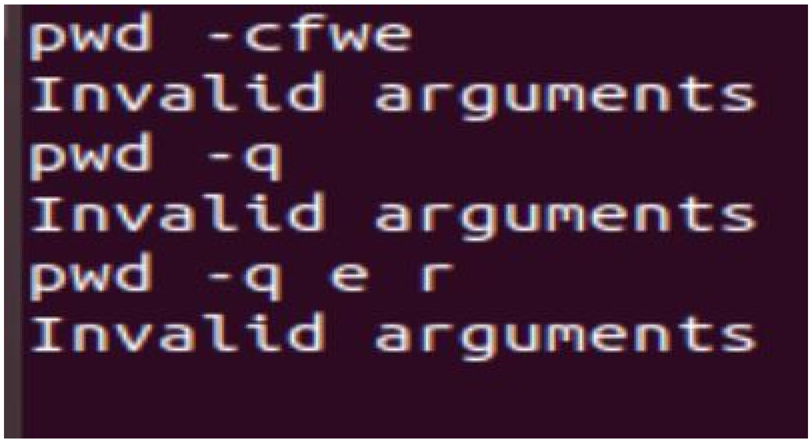


```
pwd: pwd [-LP]
Print the name of the current working directory.

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:
Returns 0 unless an invalid option is given or the current directory
cannot be read.
```

It displays the help section for the pwd command.

Error handling and corner cases

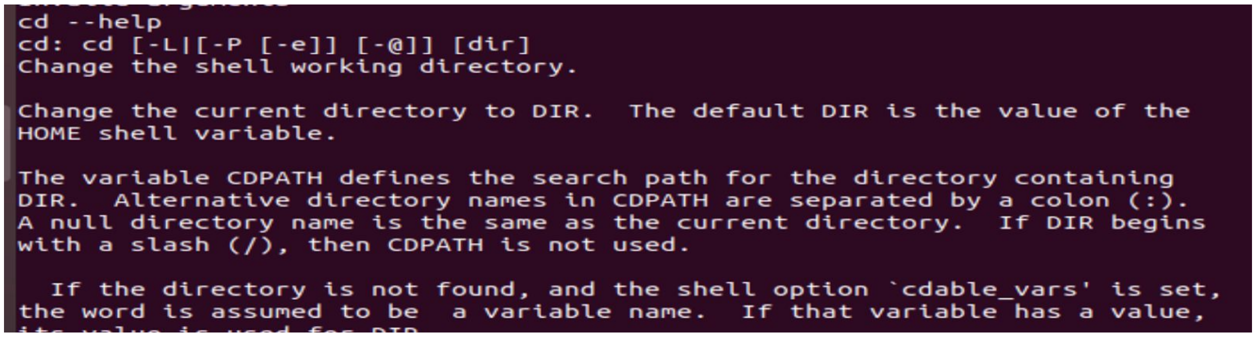
1) **In case the correct arguments / correct number of arguments aren't entered by the user then it displays error message of invalid arguments.**



```
pwd -cfwe
Invalid arguments
pwd -q
Invalid arguments
pwd -q e r
Invalid arguments
```

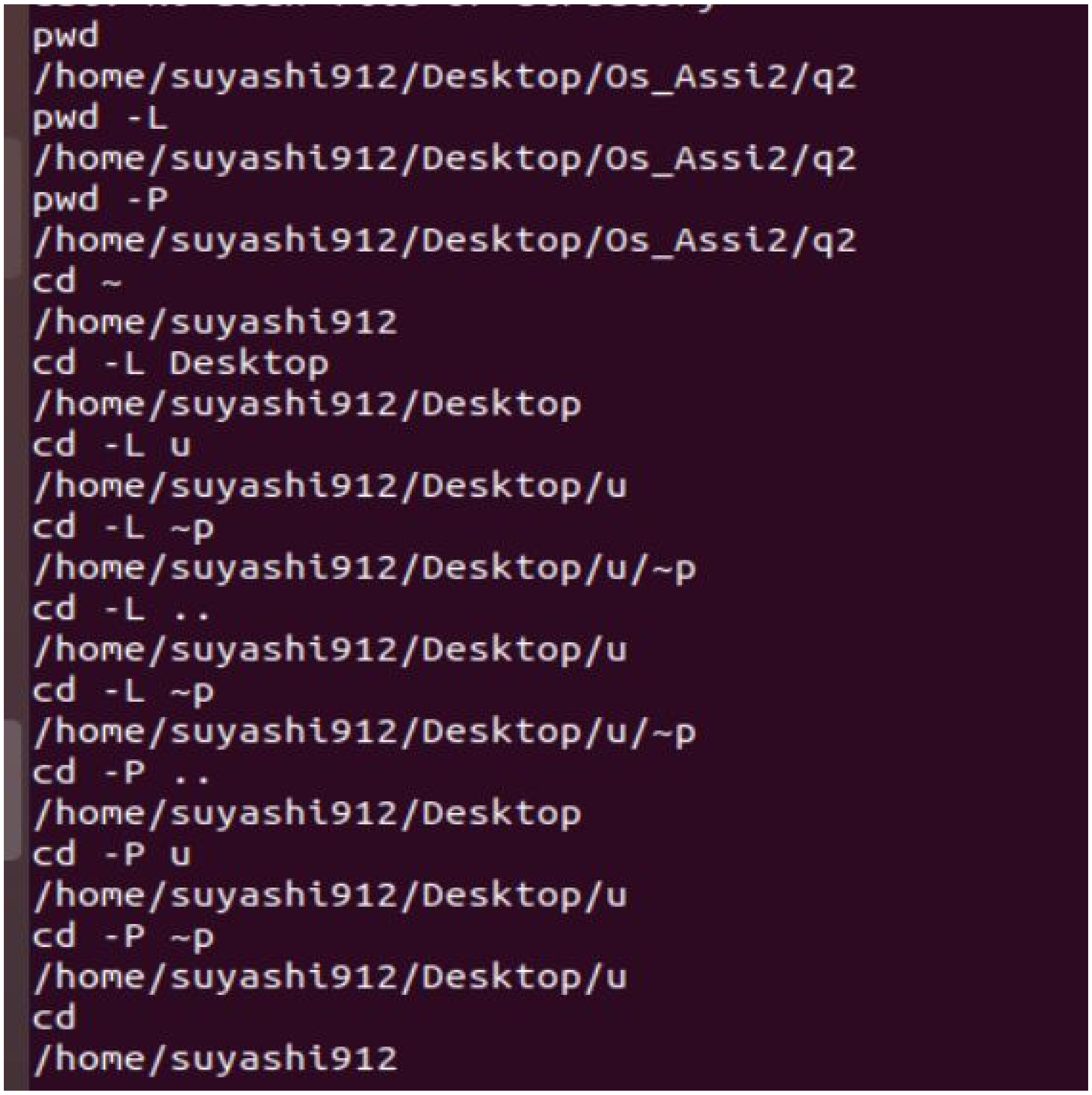
c) **cd - This command is used to change the current working directory of user.**
   Flags used
   1) **'-P' -**
      - **It avoids the symbolic links and only considers the absolute physical path.**
      - **It resolves symbolic links.**
      - **In my code I just specify the path to the directory that we want to navigate to in chdir() function. It takes us the the required directory.**
   2) **'-L' -**
      - **The symbolic links are followed in this case. I have stored the symbolic link in a temporary array and change it as and when required.**
   3) **cd ~ - it takes the user to its home directory**
   4) **cd - it has the same functionality as cd ~**
   5) **'--help' - this command displays the man pages/help section for cwd**

Test cases



```
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.

  If the directory is not found, and the shell option `cdable_vars' is set,
the word is assumed to be  a variable name.  If that variable has a value,
```

This is the --help function used to display the help section of cd.

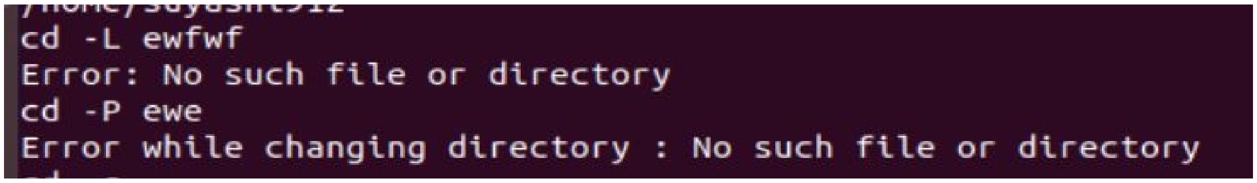


```
pwd
/home/suyashi912/Desktop/Os_Assi2/q2
pwd -L
/home/suyashi912/Desktop/Os_Assi2/q2
pwd -P
/home/suyashi912/Desktop/Os_Assi2/q2
cd ~
/home/suyashi912
cd -L Desktop
/home/suyashi912/Desktop
cd -L u
/home/suyashi912/Desktop/u
cd -L ~p
/home/suyashi912/Desktop/u/~p
cd -L ..
/home/suyashi912/Desktop/u
cd -L ~p
/home/suyashi912/Desktop/u/~p
cd -P ..
/home/suyashi912/Desktop
cd -P u
/home/suyashi912/Desktop/u
cd -P ~p
/home/suyashi912/Desktop/u
cd
/home/suyashi912
```

In the above test case ~p is a symbolic link inside u. We can see that -L gives us the logical path while -P takes us to the physical paths after removing the symbolic links.

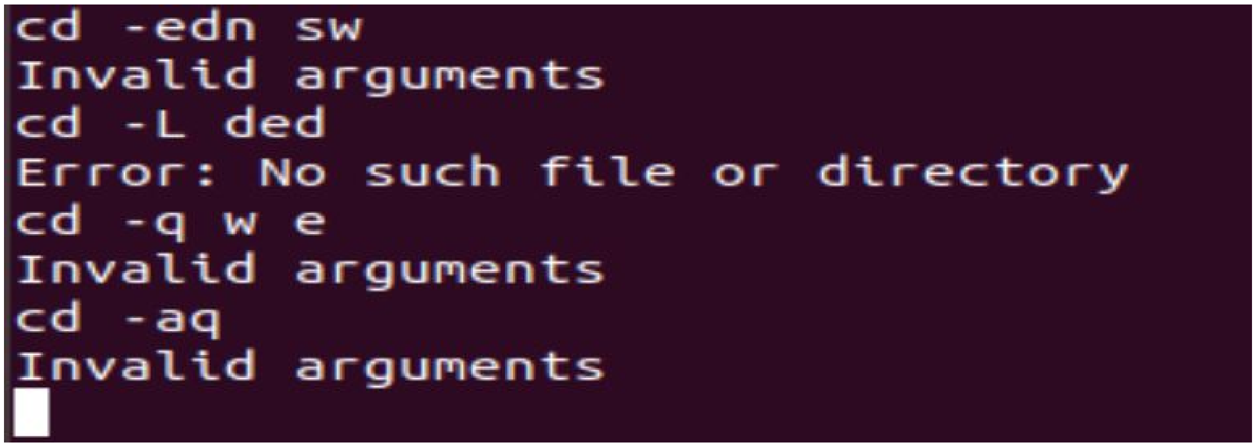
Error handling and corner cases

1) It displays an error when it cannot find a file or directory to which it can go to .



```
/home/suyashi912
cd -L ewfwf
Error: No such file or directory
cd -P ewe
Error while changing directory : No such file or directory
```

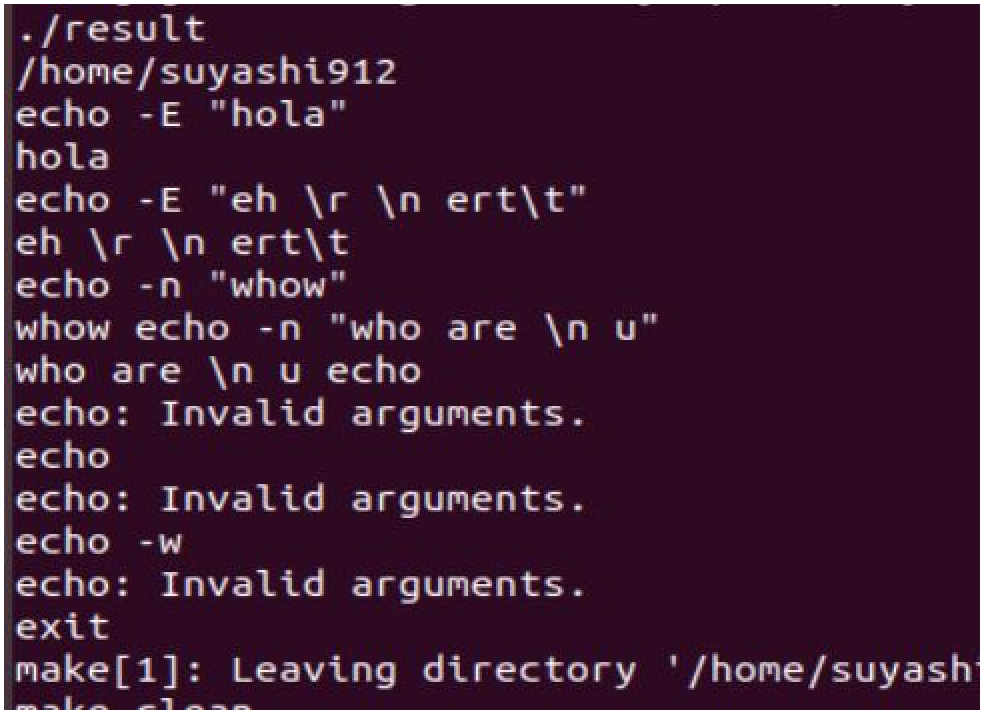
2) In case the correct arguments / correct number of arguments aren’t entered by the user then it displays error message of invalid arguments.



```
cd -edn sw
Invalid arguments
cd -L ded
Error: No such file or directory
cd -q w e
Invalid arguments
cd -aq
Invalid arguments
```

d) echo → This command displays the arguments specifies to it on the terminal
   Flags used -
   1) ‘-n’ →
      - This flag is used to print the argument without appending a new line.
      - In my code I read the argument word by word. I remove the quotes and concatenate them together. I then print them without appending a line to them
   2) ‘-E’ →
      - This is the default flag which is used to ignore the escape characters / backslash - escaped character.
      - In my code I read the argument word by word. I remove the quotes and concatenate them together. I then print them
      - while suppressing the backslash characters.

Test cases

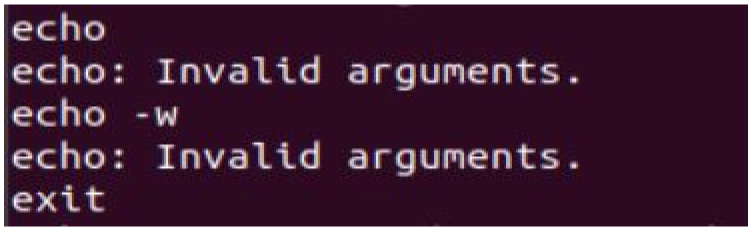


```
./result
/home/suyashi912
echo -E "hola"
hola
echo -E "eh \r \n ert\t"
eh \r \n ert\t
echo -n "whow"
whow echo -n "who are \n u"
who are \n u echo
echo: Invalid arguments.
echo
echo: Invalid arguments.
echo -w
echo: Invalid arguments.
exit
make[1]: Leaving directory '/home/suyash
```

Above is a basic test case for echo

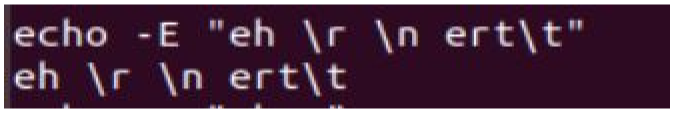
Error handling and corner cases

1) **In case the correct arguments / correct number of arguments aren't entered by the user then it displays error message of invalid arguments.**

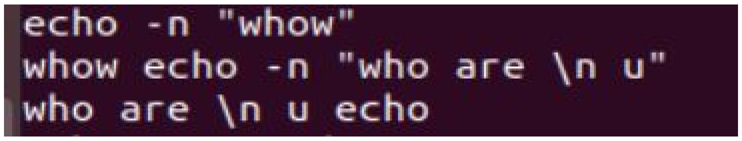


```
echo
echo: Invalid arguments.
echo -w
echo: Invalid arguments.
exit
```

2) **It can handle space separated message to be echoed within quotes without any problem.**



```
echo -E "eh \r \n ert\t"
eh \r \n ert\t
```



```
echo -n "whow"
whow echo -n "who are \n u"
who are \n u echo
```

e) **exit**
This command is used to exit from a program.

2. External commands

a) **cat → It is used to print the details of files on the terminal or to concatenate files together.**

Flags used -

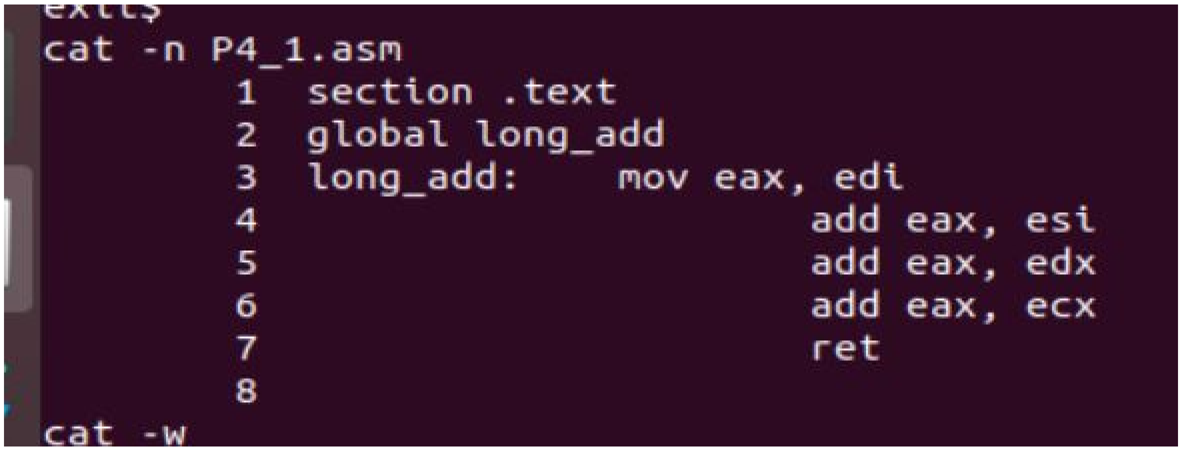
1) **'-n'**

   This flag numbers the lines in the output. I just take a counter in my code and increase it by 1 after printing each line.

2) **'-E'**

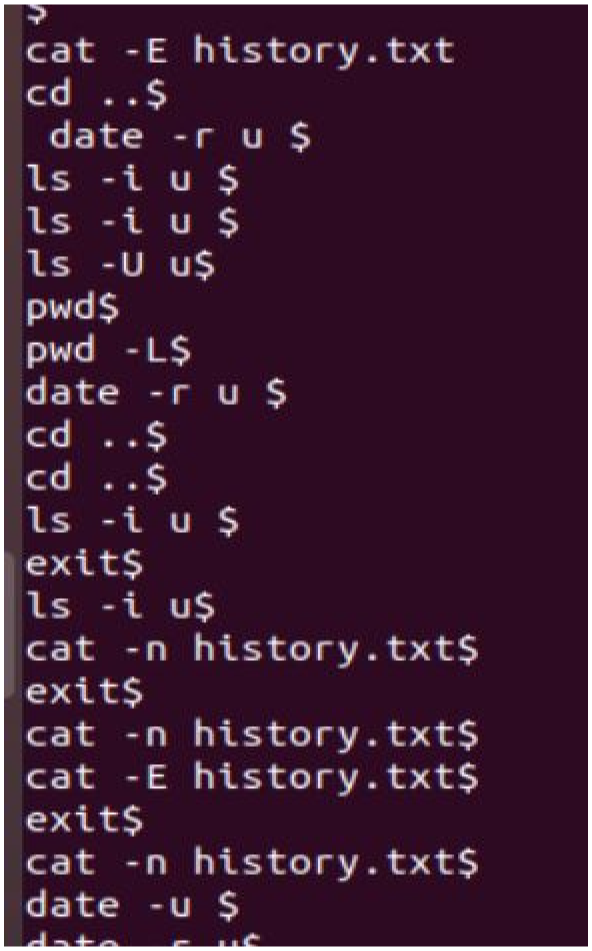
   This flag is used to append a $ character at the end of each line. In my code I print each line followed by $ character.

Test cases



```
exit$
cat -n P4_1.asm
        1  section .text
        2  global long_add
        3  long_add:    mov eax, edi
        4                       add eax, esi
        5                       add eax, edx
        6                       add eax, ecx
        7                       ret
        8
cat -w
```

Using -n flag of cat command

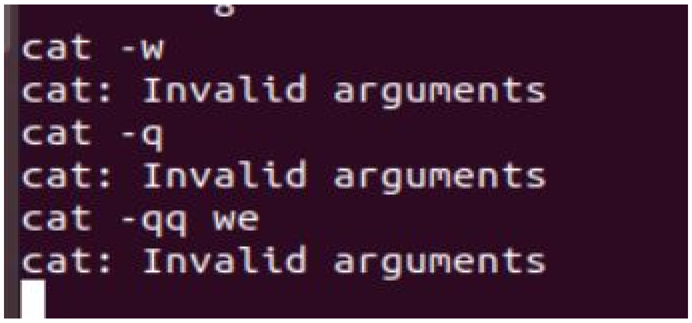


```
$
cat -E history.txt
cd ..$
 date -r u $
ls -i u $
ls -i u $
ls -U u$
pwd$
pwd -L$
date -r u $
cd ..$
cd ..$
ls -i u $
exit$
ls -i u$
cat -n history.txt$
exit$
cat -n history.txt$
cat -E history.txt$
exit$
cat -n history.txt$
date -u $
```

Using -E flag of cat command

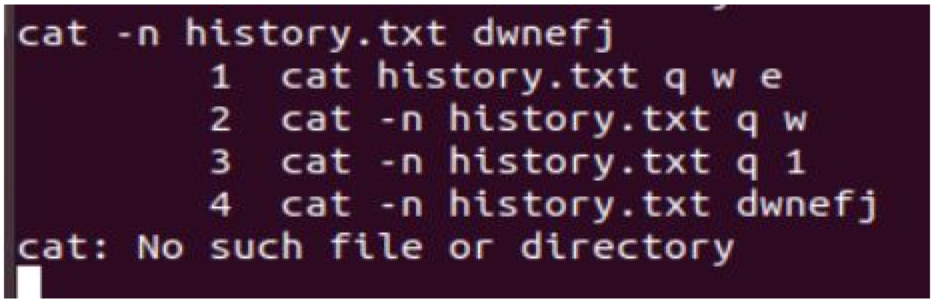
Error handling and corner cases

1) **If the user enters wrong arguments then an error message is displayed saying "Invalid arguments"**



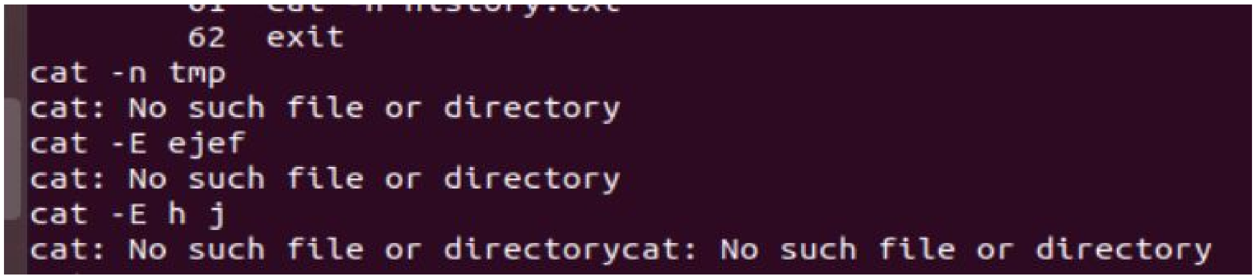
```
cat -w
cat: Invalid arguments
cat -q
cat: Invalid arguments
cat -qq we
cat: Invalid arguments
```

2) **The code handles multiple files. We can display the data of multiple files one after another using cat command**



```
cat -n history.txt dwnefj
        1  cat history.txt q w e
        2  cat -n history.txt q w
        3  cat -n history.txt q 1
        4  cat -n history.txt dwnefj
cat: No such file or directory
```

3) **In case the file from which we want cat to read data doesnt exists it displays an error message using the strerror() function that takes errno as the parameter.**



```
       62  exit
cat -n tmp
cat: No such file or directory
cat -E ejef
cat: No such file or directory
cat -E h j
cat: No such file or directorycat: No such file or directory
```

b) **Ls → It is used to print the content of the directory on the terminal.**
   Flags used
   1) **'-a'-**
- **This is the -a flag to display hidden files.**
- **I have used schdir() to get the names of the files and sort them using alphasort.**
- **I display all the file names row wise without ignoring the files that begin with '.'.**
    - ■
   2) **'-U'**
    - **This flag is used to display the contents of the directory in an unsorted order i.e in the order in which they are stored in the directory.**
   3) **'-1'**
- **This is used to display one file in each line.**
- **I have used schdir() to get the names of the files and sort them using alphasort.**

- I display all the file names in subsequent row while also ignoring the files that begin with ‘.’.

4) ‘-s’

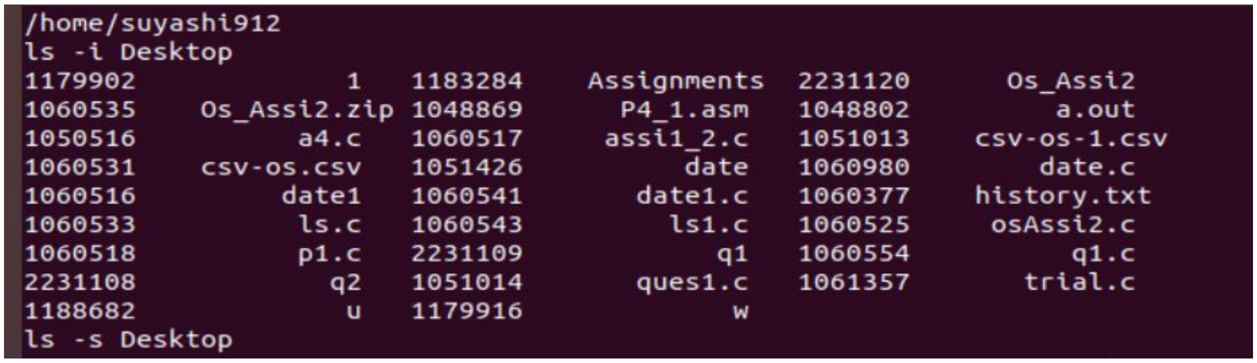
- This is the -s command that displays the size allocated to each file in blocks.
- I have used schdir() to get the names of the files and sort them using alphasort.
- I display all the file names and file size row wise while also ignoring the files that begin with ‘.’.

5) ‘-i’

- This is the -i flag that shows the index number of each file
- I have used schdir() to get the names of the files and sort them using alphasort.
- I display all the file names and index numbers row wise while also ignoring the files that begin with ‘.’.

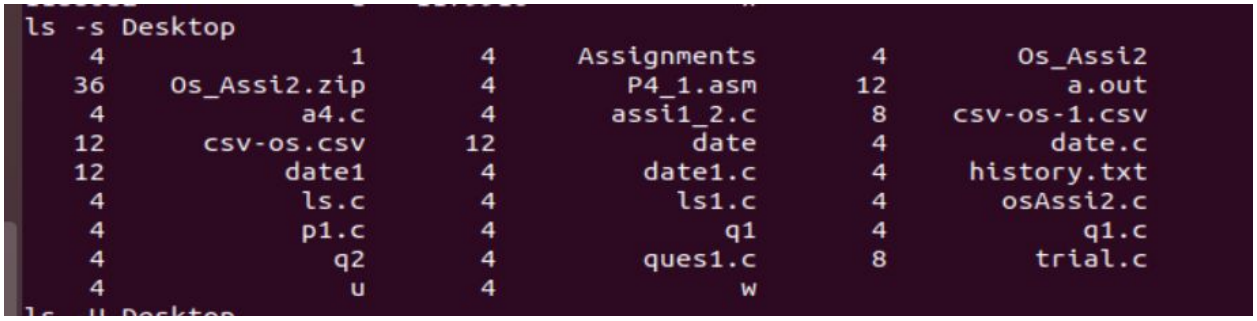
Test cases

I am calling each of the 5 above ls flags on my desktop and the obtain the following output -



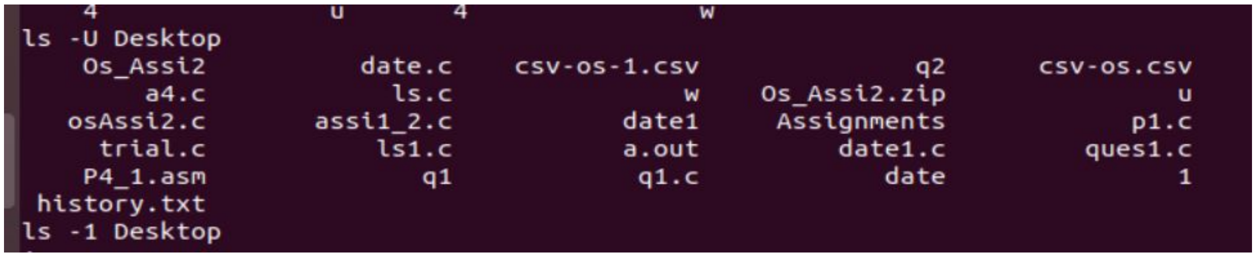
```
/home/suyashi912
ls -i Desktop
1179902            1   1183284    Assignments  2231120      Os_Assi2
1060535    Os_Assi2.zip 1048869      P4_1.asm   1048802         a.out
1050516          a4.c   1060517      assi1_2.c  1051013    csv-os-1.csv
1060531    csv-os.csv   1051426           date  1060980        date.c
1060516         date1   1060541        date1.c  1060377    history.txt
1060533          ls.c   1060543          ls1.c  1060525     osAssi2.c
1060518          p1.c   2231109             q1  1060554          q1.c
2231108            q2   1051014        ques1.c  1061357       trial.c
1188682             u   1179916              w
ls -s Desktop
```

This is the -i flag that shows the index number of each file



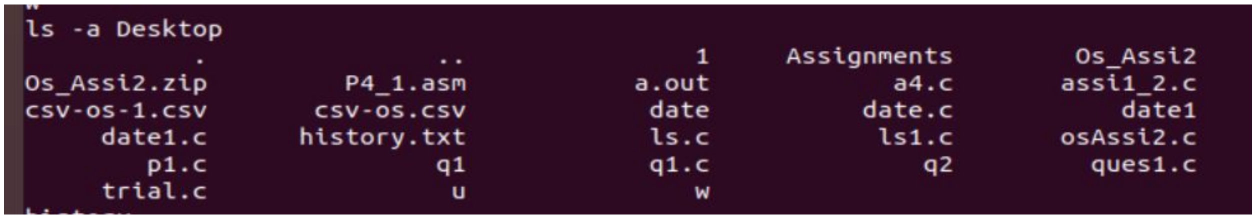
```
ls -s Desktop
   4             1       4     Assignments       4          Os_Assi2
  36    Os_Assi2.zip     4        P4_1.asm      12             a.out
   4            a4.c     4       assi1_2.c       8      csv-os-1.csv
  12      csv-os.csv    12            date       4            date.c
  12           date1     4         date1.c       4       history.txt
   4            ls.c     4           ls1.c       4        osAssi2.c
   4            p1.c     4              q1       4             q1.c
   4              q2     4         ques1.c       8          trial.c
   4               u     4               w
```

This is the -s command that displays the size allocated to each file in blocks.



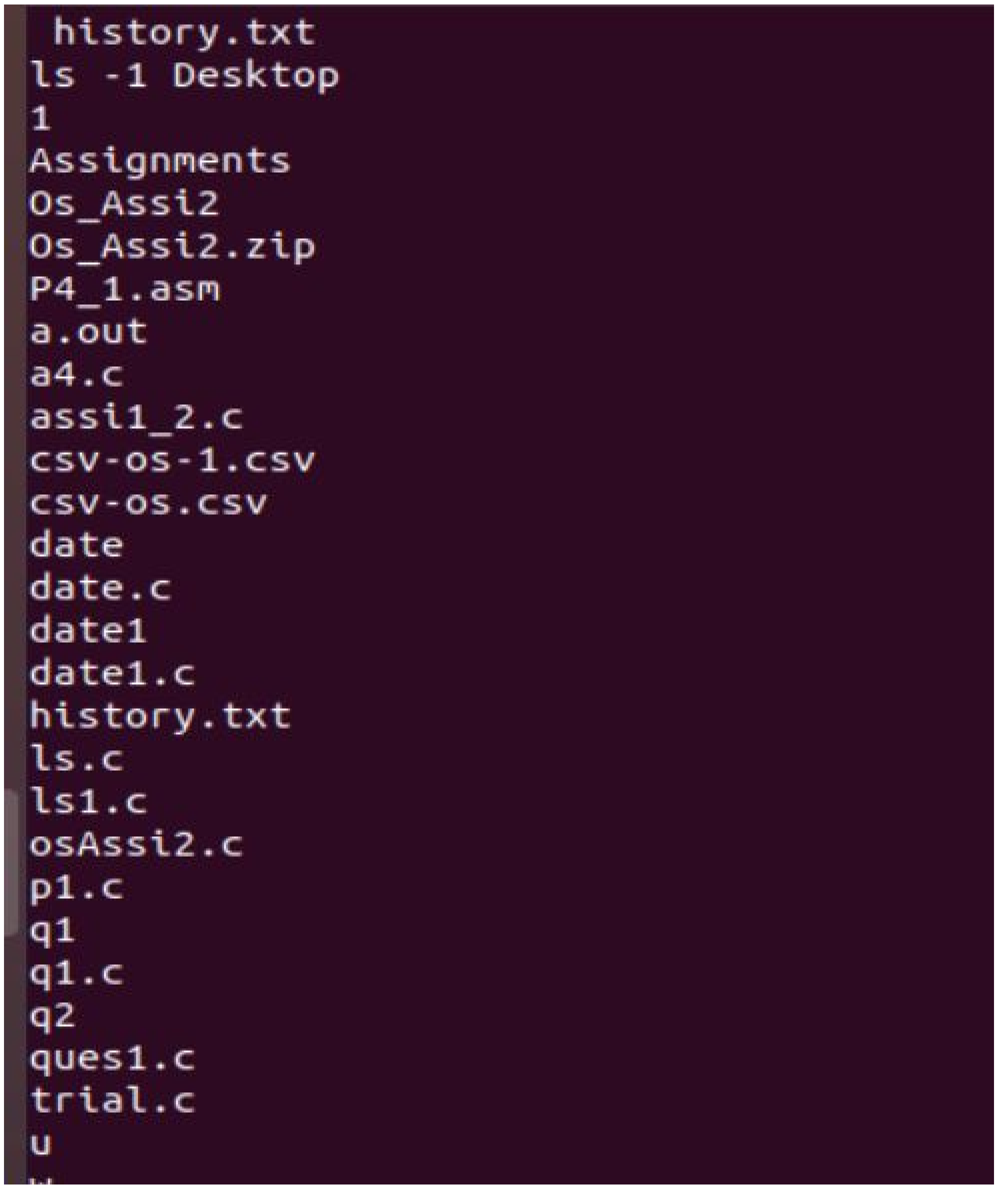
```
ls -U Desktop
    Os_Assi2        date.c   csv-os-1.csv             q2     csv-os.csv
        a4.c          ls.c              w   Os_Assi2.zip              u
   osAssi2.c     assi1_2.c          date1    Assignments           p1.c
     trial.c         ls1.c          a.out        date1.c        ques1.c
    P4_1.asm            q1           q1.c           date              1
 history.txt
ls -1 Desktop
```

This is the -U flag to display the files in their natural order in which it is present in the directory.



```
ls -a Desktop
           .            ..              1    Assignments       Os_Assi2
Os_Assi2.zip      P4_1.asm          a.out           a4.c      assi1_2.c
csv-os-1.csv    csv-os.csv           date         date.c          date1
     date1.c   history.txt           ls.c          ls1.c      osAssi2.c
        p1.c            q1           q1.c             q2        ques1.c
     trial.c             u              w
```

This is the -a flag to display hidden files.

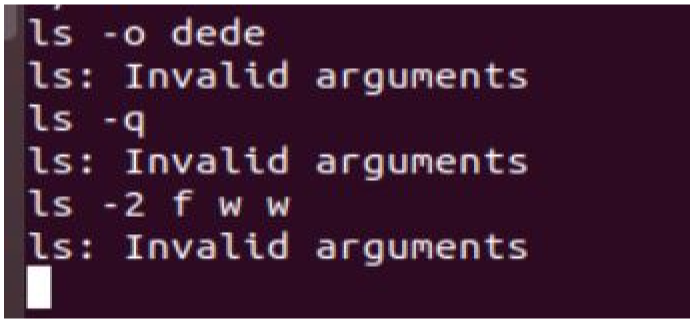


```
 history.txt
ls -1 Desktop
1
Assignments
Os_Assi2
Os_Assi2.zip
P4_1.asm
a.out
a4.c
assi1_2.c
csv-os-1.csv
csv-os.csv
date
date.c
date1
date1.c
history.txt
ls.c
ls1.c
osAssi2.c
p1.c
q1
q1.c
q2
ques1.c
trial.c
u
```

This is the -1 flag that displays the flags row wise.

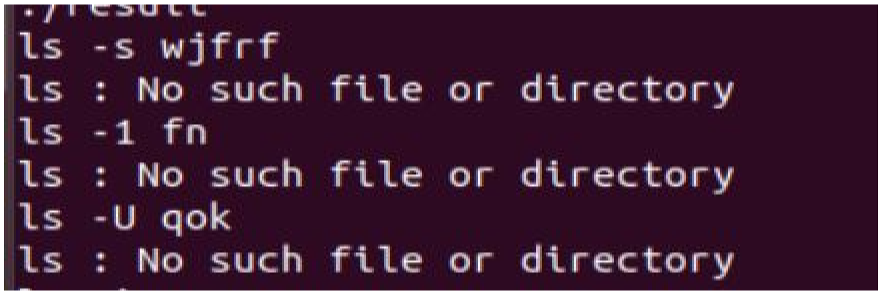
ERROR HANDLING AND CORNER CASES

1) **If the user enters wrong arguments then an error message is displayed saying “Invalid arguments”**



```
ls -o dede
ls: Invalid arguments
ls -q
ls: Invalid arguments
ls -2 f w w
ls: Invalid arguments
```

2) If the file on which we call ls does not exist in that directory, then an error message is shown that no such file or directory exists.
   This is done by using the strerror() function which takes errno as its parameter.

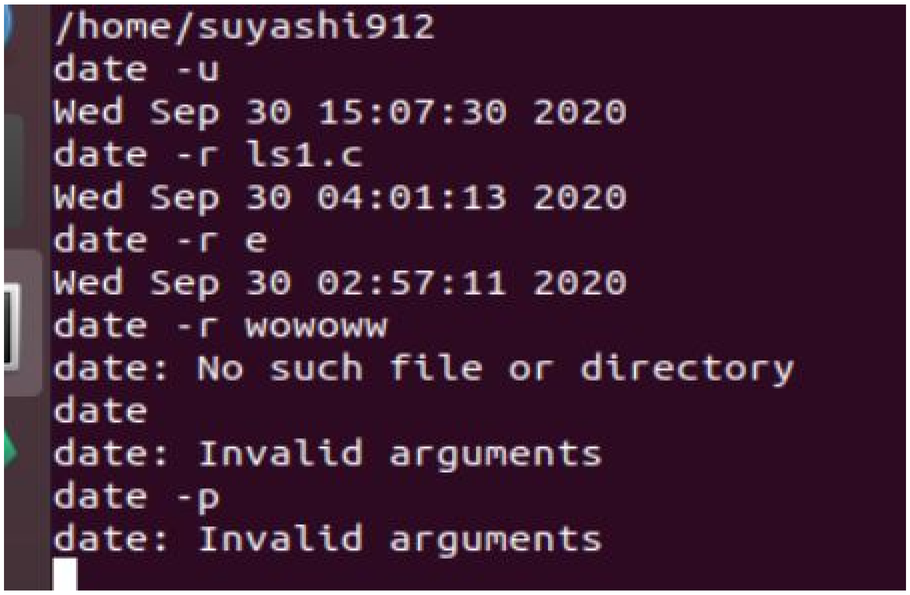


```
ls -s wjfrf
ls : No such file or directory
ls -1 fn
ls : No such file or directory
ls -U qok
ls : No such file or directory
```

c) date - This is used to access various formats and kinds of dates. Using it we can set the date and time of a system or print dates and time etc.
   Flags used
   1) '-u' -
      - This flag is used to print the UTC i.e the Coordinated universal time.
      - For this command I used the time.h header files which contains function gmtime() to get the UTC time.
   2) '-r'
      - This flag takes a file as its argument.
      - It prints the last modification date of the file stored in the system. It prints the time and date in the timeline of the location.
      - For eg in my case it prints the IST time.

- **Here again I am using the time.h header file. I am also using the stat struct that stores the time of the last modification of the file.**

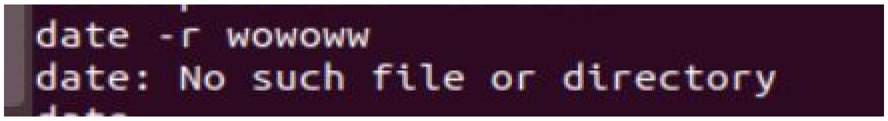
Test Cases



```
/home/suyashi912
date -u
Wed Sep 30 15:07:30 2020
date -r ls1.c
Wed Sep 30 04:01:13 2020
date -r e
Wed Sep 30 02:57:11 2020
date -r wowoww
date: No such file or directory
date
date: Invalid arguments
date -p
date: Invalid arguments
```

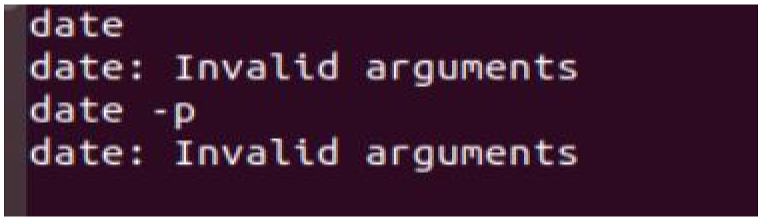
Error handling and corner cases

1. **In case the file entered with the command -r which prints the local time of last modification, then error is printed using errno value passed in strerror() function.**



```
date -r wowoww
date: No such file or directory
```

2. **In case the user doesn't enter atleast two arguments (minimum) or doesn't enter the correct flag then the "invalid arguments " error is displayed.**



```
date
date: Invalid arguments
date -p
date: Invalid arguments
```

3. **In case the utc time fails to be displayed in -u flag , then we print an error message.**

d) rm → This command is used to remove specific files from the system.
Flags -

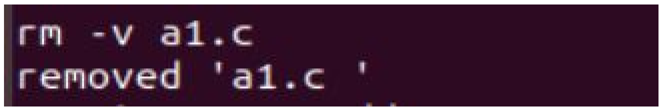
1) '-i' →

- **This flag prompts the user for a “yes” or “no” before deleting the file. If the user enters “yes” or “y” then the file is removed from the system.**
  Implementation -
- **In my code the if statement first checks if the flag ‘-i’ is selected by the user or not.**
- **If yes, then it checks the arguments. The while loop takes the arguments till it encounters a null argument and stops. We can thus take multiple files or a single file with spaces in it.**
- **The code then prompts the user about whether to delete the file or not.**
- **If the user enters “yes” or “y”, it tries to remove the file. In case the file does not exist or is a non empty directory it shows an error.**
- **If the user says no, then it ignores the rm command and moves to the next argument.**
- **In case the user enters “yes” for non-existent file names or non empty directories, it prints the error message instead of deleting them .**
- **These error messages are printed using strerror() function which takes in errno as the parameter. Errno is set whenever an error is encountered.**

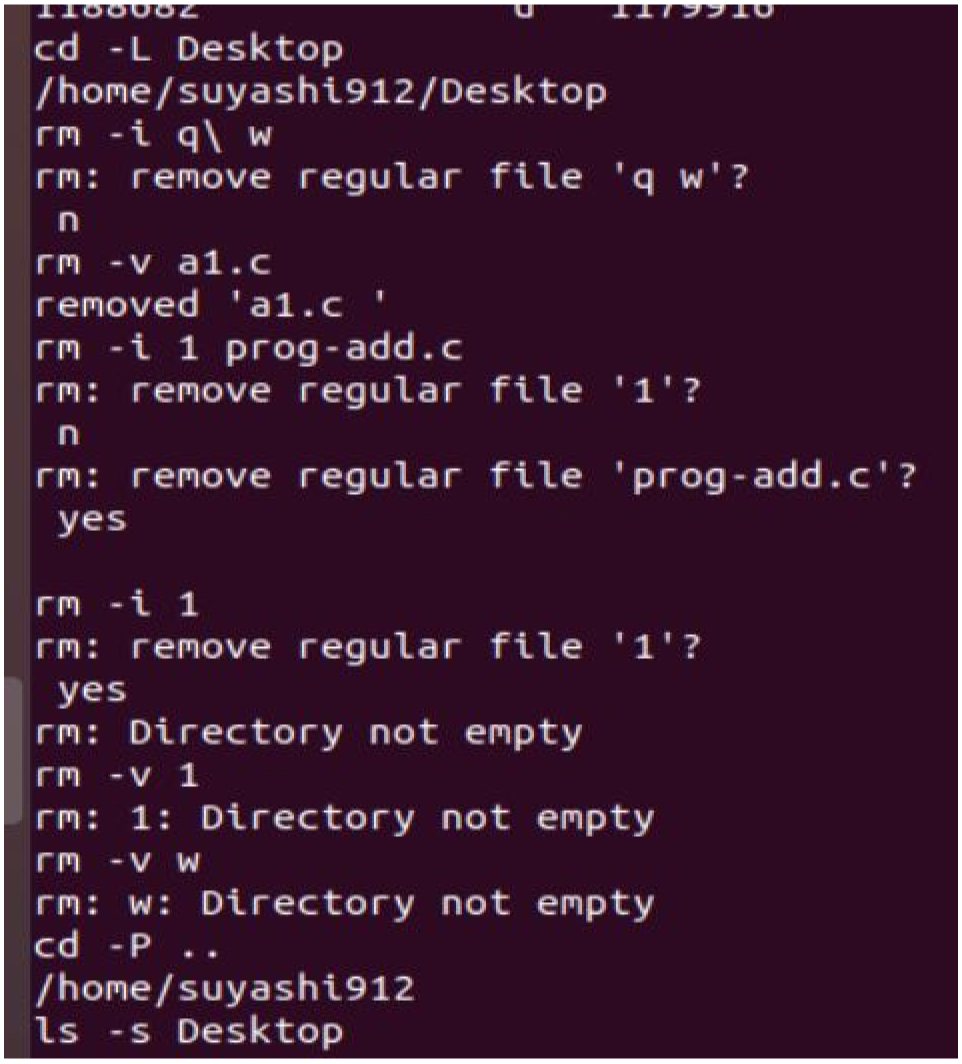
2) -v → This flag provides a verbose of the fact that a given file / directory is removed from the system. In case the file does not exist or the directory isn’t empty it displays an error message.
  Implementation -
- **The if statement within the code firstly checks if the flag is ‘-v’ or not. This flag basically gives the user a written statement about the file he wants to remove.**
- **Again while loop takes the arguments till it encounters a null argument and stops. We can thus take multiple files or a single file with spaces in it or both.**
- **The code then tries to delete all the specified files and prints the delete message or error message on by one.**

- **In case the user enters non-existent file names or non empty directories, it prints the error message instead of deleting them .**
- **These error messages are printed using strerror() function which takes in errno as the parameter. Errno is set whenever an error is encountered.**

Test cases



```
rm -v a1.c
removed 'a1.c '
```



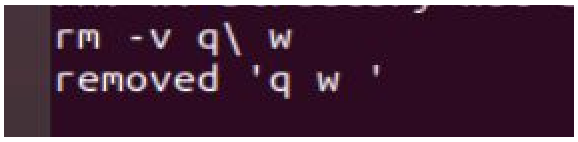
```
cd -L Desktop
/home/suyashi912/Desktop
rm -i q\ w
rm: remove regular file 'q w'?
 n
rm -v a1.c
removed 'a1.c '
rm -i 1 prog-add.c
rm: remove regular file '1'?
 n
rm: remove regular file 'prog-add.c'?
 yes

rm -i 1
rm: remove regular file '1'?
 yes
rm: Directory not empty
rm -v 1
rm: 1: Directory not empty
rm -v w
rm: w: Directory not empty
cd -P ..
/home/suyashi912
ls -s Desktop
```

Error and corner case handling

1) **When the name of the file contains spaces in it.**

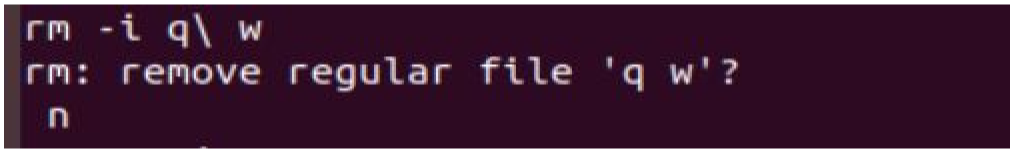
I have used '\' as the delimiter to consider spaces . A file named "q w" can also be written as q\ w and would be considered a single file with a space in it.

It shows that the file 'q w' has been deleted.



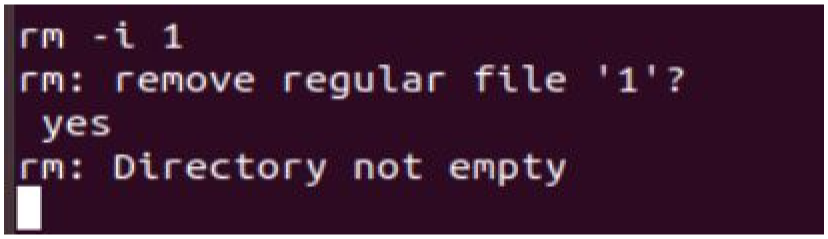
```
rm -v q\ w
removed 'q w '
```

-i prompts the user asking him whether to delete the file or not. Since I have written 'n', it won't be deleted.

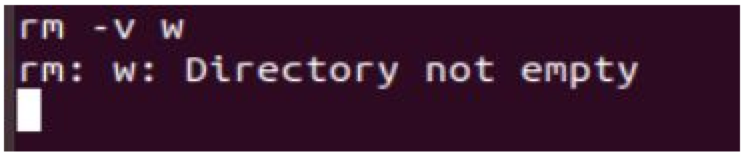


```
rm -i q\ w
rm: remove regular file 'q w'?
 n
```

2) **The rm command doesn't remove non empty directories. Hence it prints an error message whenever it encounters such problems.**

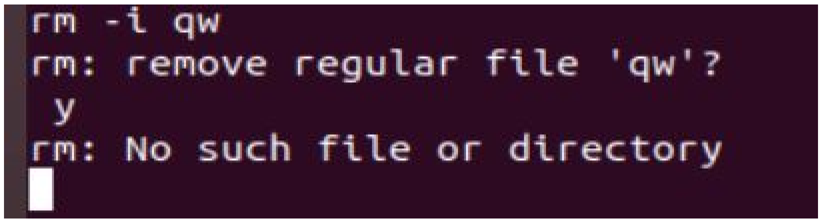


```
rm -i 1
rm: remove regular file '1'?
 yes
rm: Directory not empty
```

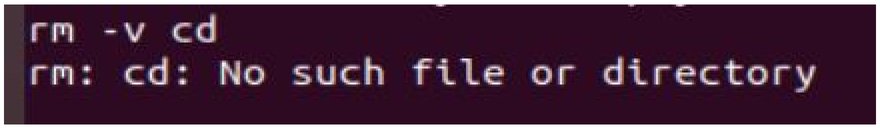


```
rm -v w
rm: w: Directory not empty
```

3) **In the case when the file / directory that the user enters doesn't exist, then it gives the following error.**

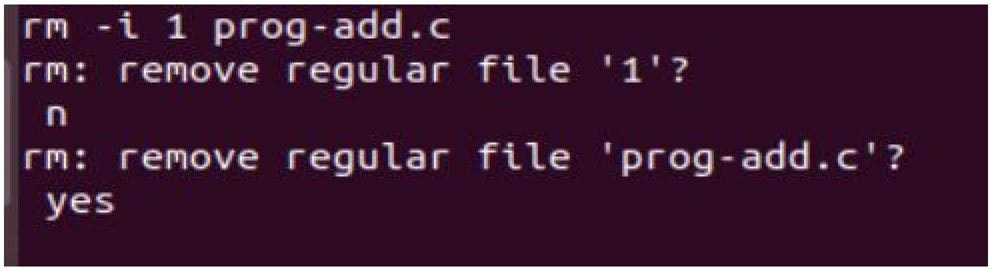


```
rm -i qw
rm: remove regular file 'qw'?
 y
rm: No such file or directory
```



```
rm -v cd
rm: cd: No such file or directory
```

4) **I have handled the commands in such a way that they can take multiple files to be deleted -**



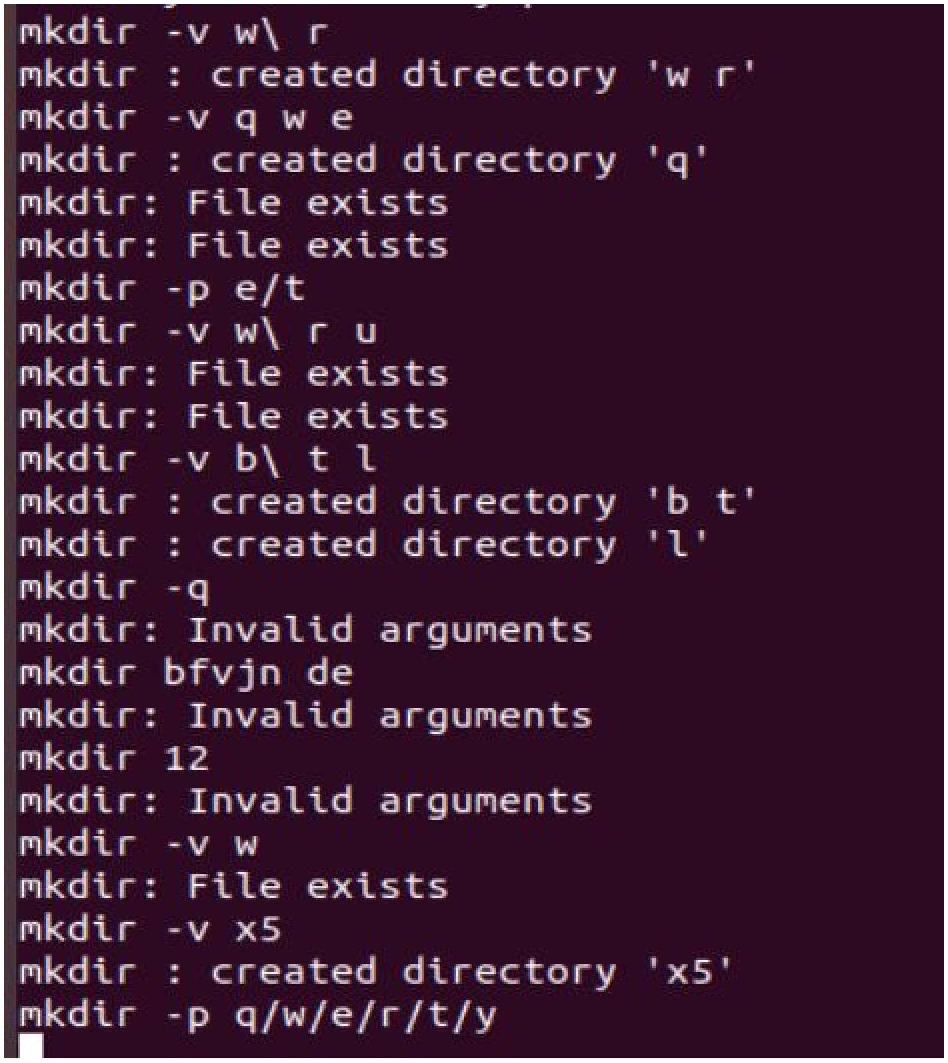
```
rm -i 1 prog-add.c
rm: remove regular file '1'?
 n
rm: remove regular file 'prog-add.c'?
 yes
```

e) **mkdir - It is used to make directories in the system.**

Flags used

1) **'-v'**
    - **This flag prints a message every time a directory is created. If the directory already exists, it also prints that.**
    - **I have used the c function mkdir() to create directories**
    - **I have handled the multiple file condition as well as the file name with spaces in it.**
2) **'-p'**
- **This flag is used to make directories recursively one within the other. The outer directory is called the parent directory.**
- **In case the parent directory already exists it doesn't display any error.**
- **I have used the c function mkdir() to create directories. After creating the parent directory I go into it using chdir() and create a new directory in it and so on.**
- **At the end I return back to the parent directory in which I was actually present at the beginning.**

Test case

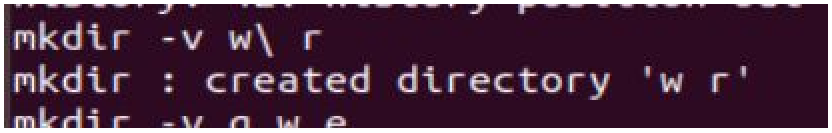


```
mkdir -v w\ r
mkdir : created directory 'w r'
mkdir -v q w e
mkdir : created directory 'q'
mkdir: File exists
mkdir: File exists
mkdir -p e/t
mkdir -v w\ r u
mkdir: File exists
mkdir: File exists
mkdir -v b\ t l
mkdir : created directory 'b t'
mkdir : created directory 'l'
mkdir -q
mkdir: Invalid arguments
mkdir bfvjn de
mkdir: Invalid arguments
mkdir 12
mkdir: Invalid arguments
mkdir -v w
mkdir: File exists
mkdir -v x5
mkdir : created directory 'x5'
mkdir -p q/w/e/r/t/y
```

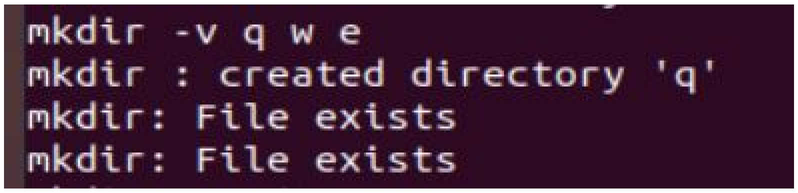
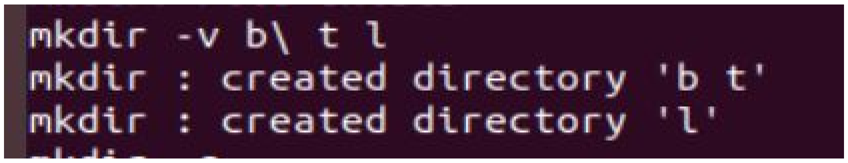
It shows the various cases of -v and -p flags.
It also shows the error messages when the directory already exists or when the arguments aren't valid.

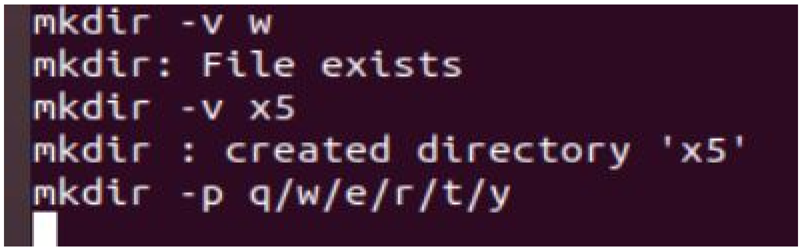
Error handling and corner cases

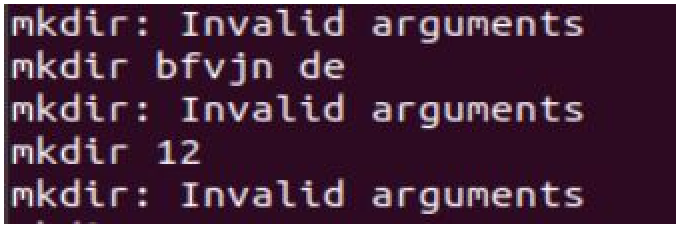
1) **When the name of the file contains spaces in it.**
   I have used '\' as the delimiter to consider spaces . A file named "q w" can also be written as q\ w and would be considered a single file with a space in it.
   I have also handled multiple file case. Hence the user can enter more than one file.



```
mkdir -v w\ r
mkdir : created directory 'w r'
```


```
mkdir -v q w e
mkdir : created directory 'q'
mkdir: File exists
mkdir: File exists
```

```
mkdir -v b\ t l
mkdir : created directory 'b t'
mkdir : created directory 'l'
```
2) **In case the directory already exists, mkdir gives a warning that the file already exists. In case of -p if the parent directory exists, it doesnt give an error and uses that directory itself.**


```
mkdir -v w
mkdir: File exists
mkdir -v x5
mkdir : created directory 'x5'
mkdir -p q/w/e/r/t/y
```
3) **In case the user doesn’t enter the correct set of arguments or doesn’t enter the correct flag then the “invalid arguments “ error is displayed.**


```
mkdir: Invalid arguments
mkdir bfvjn de
mkdir: Invalid arguments
mkdir 12
mkdir: Invalid arguments
```