wc → Word Count

wc stands for word count. As the name implies, it is mainly used for counting purposes.

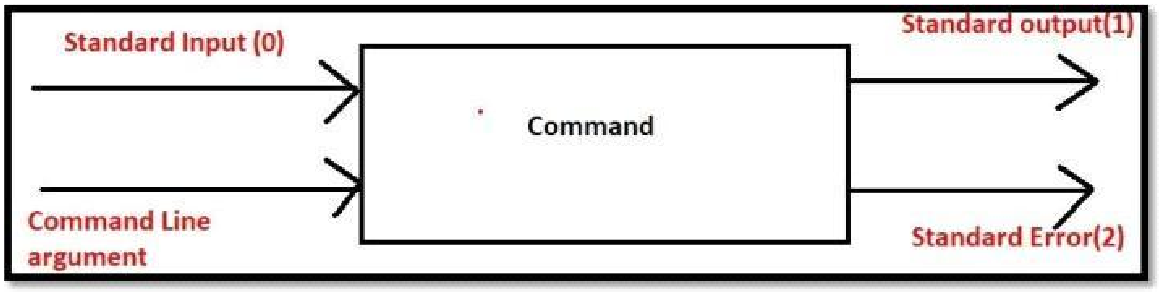
It is used to find out the number of lines, word count, byte and characters count in the files specified in the file arguments.
By default it displays four-columnar output.
First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which is given as argument.

Syntax: wc [Options] [File]

Echo

The echo command in Linux is a built-in command that allows users to display lines of text or strings that are passed as arguments. It is commonly used in shell scripts and batch files to output status text to the screen or a file.
Syntax: echo Argument

---Input and Output of Commands and Redirection------



Commands can take input, perform required operations and produce some output.
While executing commands if anything goes wrong then we will get an error message.

Command can take input either from standard Input or from command line arguments.
Command will produce results to either Standard output or Standard Error.

Standard Input, Standard Output and Standard Error are Data Streams and can flow from one place to another place. Hence redirection and piping are possible.

Command Line arguments are static and these are not streams. Hence redirection and piping concepts are not applicable to command line arguments.

These data streams are associated with some numbers.

Standard Input associated with 0.
Standard Output associated with 1.
Standard Error associated with 2.

By Default Standard input connected with keyboard, Standard output and Standard Error connected with Terminal. But we can redirect.

Standard Input from the keyboard and output to Standard Output Device:
```
$cat
```
read required input from the keyboard this data will be displayed to the standard output. Ctrl+d

Note: For the cat command if we are not providing any arguments, then the input will be taken from the standard input device (keyboard) and display the output to the standard output device (Terminal).
```
$ cat
```

This is data provided from Standard Input This is data provided from Standard Input

Input from command line arguments and error messages

Standard Error:

```
$ rm file100
rm: cannot remove 'file100': No such file or directory
```

We are providing filename as command line argument to the rm command.
Specified file not available and hence this command will produce an error message to the Standard Error device (Terminal).

Note: Some commands may accept Standard Input and Some commands may accept command line arguments.

1) rm command will always accept command line arguments only.
 rm file1 file2
2) echo command will always accept command line arguments only.
 echo "jitendra"
3) cat command can accept input either from standard input or from command line arguments.

Redirection :-

As Standard Input, Standard Output and Standard Error are Data streams, we can redirect these streams.

Redirecting Standard Output:

We can redirect standard output by using > and >> symbols.
> will perform overwriting of existing data
>> will perform appending to existing data

Eg 1: To redirect the standard output of cat command from terminal to output.txt

```
$cat 1> output.txt
sample data ctrl+d
```

sample data won't be displayed to the terminal and will write to output.txt
Redirection symbol > is always associated with 1 by default. Hence we are not required to specify 1 explicitly.

Redirecting Standard Error:
We can redirect error messages from the terminal to our own file by using > and >> symbols.

```
$ cal 34 w3892384208342 2>> error.txt
```

Now the error message won't be displayed to the console and will be written to error.txt.
For error redirection 2 is mandatory.

Redirecting Standard Input:
We can redirect standard input from the keyboard to our required file.
We can perform input redirection by using < symbol.

```
$cat 0< a.txt 1>>output.txt 2>>error.txt
```

< symbol is always associated with 0 by default. Hence we can remove.

```
$cat < a.txt >>output.txt 2>>error.txt
```

Note: To redirect both standard output and standard error to the same destination we can use shortcut as follows

```
$ cat < a.txt &> output.txt
```

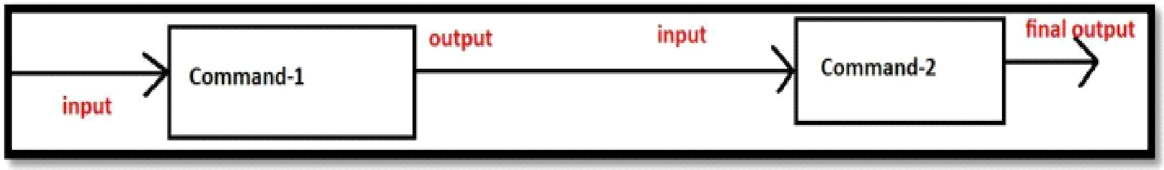
&> means both standard output and standard error.

Q1) In How Many Ways Command can get Input?

2 ways. Either from Standard Input or from command line arguments.

--------------------Piping----------------

Sometimes we can use the output of one command as input to another command. This concept is called piping.
By using piping, multiple commands will work together to fulfill our requirement.



We can implement piping by using vertical bar (|).

```
$ ls -l /etc | wc
215 1940 11872
```

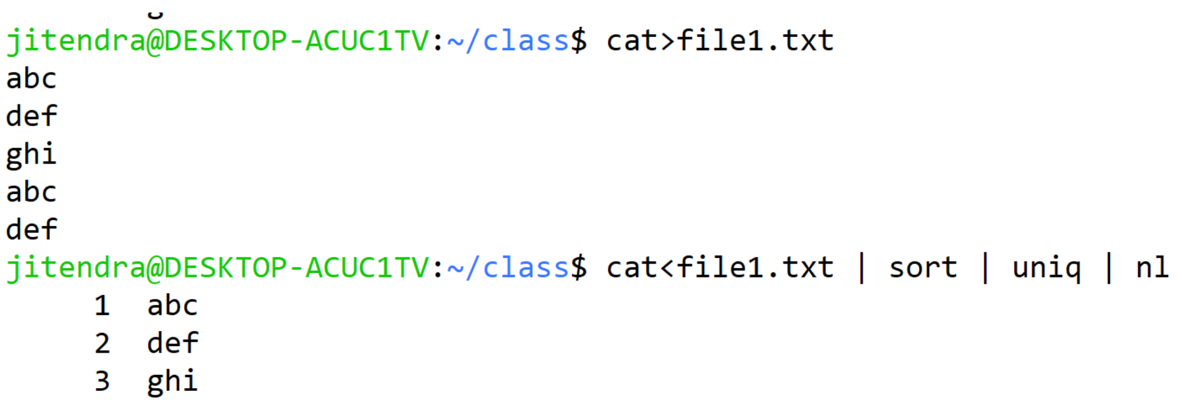
First ls got executed and the output of this command will become input to wc command

Eg 2: $ ls -l /etc | more

Eg 3: $ ls -l /etc | wc |wc -l The output is: 1

Eg 4: $ ls -l /etc | head -5

ex:- given a text file sort it , remove duplicates, give line numbering



```
jitendra@DESKTOP-ACUC1TV:~/class$ cat>file1.txt
abc
def
ghi
abc
def
jitendra@DESKTOP-ACUC1TV:~/class$ cat<file1.txt | sort | uniq | nl
     1  abc
     2  def
     3  ghi
```

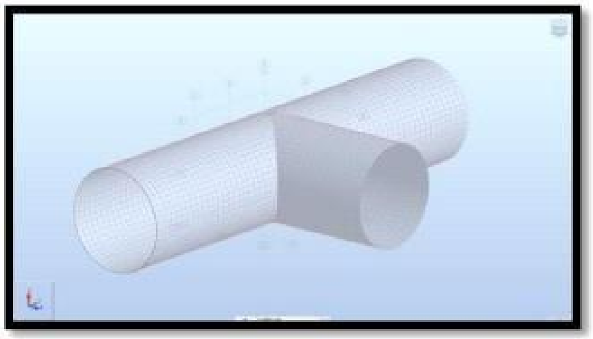
tee :-

Requirement:
The output of the ls command should be saved to output.txt and should be provided as input to wc command:
ls -l 1>output.txt | wc

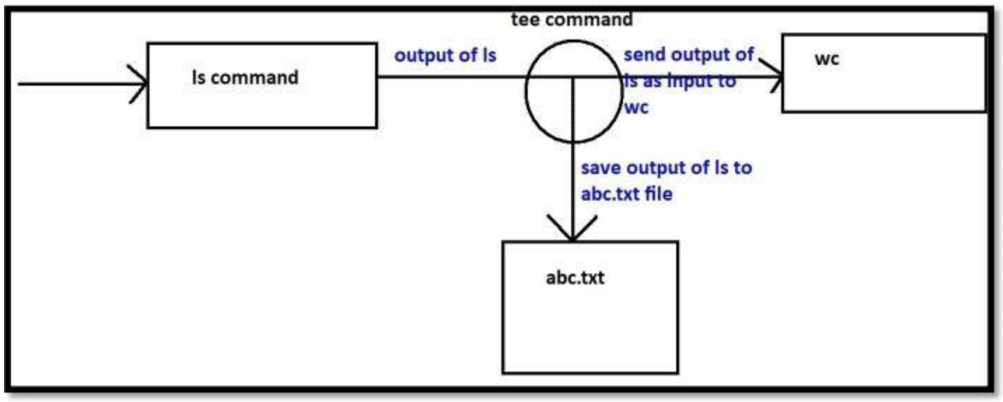
This command won't work because if we are using redirection in the middle of piping, it will break piping concept.

In piping, if we want to save the output of one command to a file and if we want to pass that output as input to next command simultaneously, then we should go for tee command

tee command is just like T-Junction or T-Pipe. It will take one input but provides two outputs.



Eg 1: To save the output of ls command to a file and to display to the terminal simultaneously.
$ ls –l →It will display to the terminal
$ ls -l > abc.txt → ▯ It will save to the abc.txt but won't display to the terminal.
$ ls -l | tee abc.txt It will save to the abc.txt and display to the terminal.


tee command
ls command
output of ls
send output of ls as input to wc
wc
save output of ls to abc.txt file
abc.txt

chmod

chmod means change mode. We can use the chmod command to change file or directory permissions.

Syntax: chmod [options] [mode] [File_name]

Options	Description
`-R`	Apply the permission change recursively to all the files and directories within the specified directory.
`-v`	It will display a message for each file that is processed. while indicating the permission change that was made.
`-c`	It works the same as `-v` but in this case it only displays messages for files whose permission is changed.
`-f`	It helps in avoiding display of error messages.
`-h`	Change the permissions of symbolic links instead of the files they point to.

Two Modes in chmod:

1) Symbolic: Use Symbols.
2) Octal: Use numbers

Symbolic:

The file permissions are a total of 9 permissions. First 3 are user permissions, next 3 are group permissions and next 3 are others permissions.
user permissions: rw → user can perform both read and write operations but not execute operation
group permissions: r-- →group members can perform only read operation and cannot perform write and execute operations
others permissions: r-- → other members can perform only read operation and cannot perform write and execute operations.
User Permissions + Group Permissions + Others Permissions

Operators	Definition
`+`	Add permissions
`-`	Remove permissions
`=`	Set the permissions to the specified values

Letters	Definition
`r`	Read permission
`w`	Write permission
`x`	Execute permission

Reference	Class
u	Owner
g	Group
o	Others
a	All (owner,groups,others)

Eg: For user add execute permission,for group add write permission,for others remove read permission

$ chmod u+x,g+w,o-r demo.txt
Note: Only owner and super user (root) can change file permissions.
How to check Permissions of existing File:
By using ls -l command:
$ ls -l
total 0
-rw-r--r-- 1 jitendra jitendra 0 Nov 27 21:19 demo.txt
order is important
Eg 1: $ chmod u+x demo.txt
adding execute permission to the user
Eg 2: $ chmod u+w,g+rw,o+r demo.txt
adding write permission to the user
adding read and write permissions to the group
adding read permission to the others
Eg 3: $chmod u+x,g-w,o+w demo.txt
adding execute permission to the user
removing write permission from the group
adding write permission to the others
Eg 4: $ chmod u=rw,g=rw,o=r demo.txt
Now user permissions: rwgroup permission: rwothers permission: r--
Eg 5: $ chmod a=- demo.txt
Now user permissions: ---
group permission: ---
others permission: ---
Eg 6: $ chmod a=rwx demo.txt
Now user permissions: rwx
group permission: rwx
others permission: rwx

Octal

It is also a method for specifying permissions. In this method we specify permission using three-digit number. Where..

First digit specify the permission for Owner.
Second digit specify the permission for Group.
Third digit specify the permission for Others. The digits

NOTE: The digits are calculated by adding the values of the individual permissions.

Value	Permission
4	Read Permission
2	Write Permission
1	Execute Permission

Examples of Using the Octal mode:
Suppose we give read and write permission to the file Owner. Read, write and executable permission to the Group. Read-only permission to the Other. Their command would be.

```
chmod 674 [file_name]
```

Here.

6 represent permission of file Owner which are (rw).
7 represent permission of Group which are (rwx).
4 represent permission of Other which is (r).

Read Permission to the File:-
If the file not having read permission then we are not allowed to view content of the file. Hence cat, head, tail, more, less commands won't work.
Write Permission to the File:-
If the file not having write permission, then we cannot modify the content of the file.
Execute Permission to the File:-
If the user not has executed permission on any file, then he cannot execute that file as a program.
Read Permission to the Directory:-
If the user has read permission on any directory, then he can list out the contents of that directory. i.e he can use ls command.
Write Permission on the Directory:-
If the user has write permission on any directory, then he is allowed to modify the content of that directory. i.e he can add new files and remove existing files.
Execute Permission to the Directory:-
If the user not has executed permission on any directory, then he is not allowed to enter into that directory. i.e he cannot use cd command.

Note:- If the user not having read permission on any file, then he cannot execute that file even though he has executed permission.