**Problem Statement**

Explain the below Linux commands with an example. Share the screenshot of each command with the output:

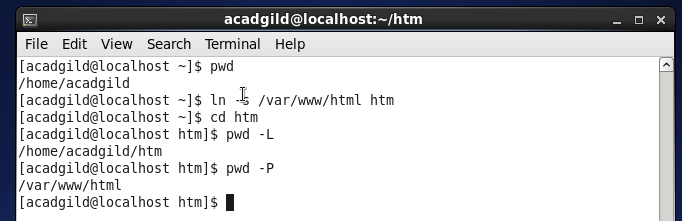
1. **pwd**

It stands for ‘**Print Working Directory**. It prints the current working directory or simply the directory user is, at present. It prints the current directory name with the complete path starting from root (/). It is built in shell command.

**pwd command options:**

$ pwd [options]

* -L (Logical) Use pwd from environment even if it contains symbolic links
* - P (Physical) Prints actual current working directory by resolving all symbolic links



1. **vi**

It stands for **Visual Editor**. It is a default editor which comes with LINUX operating system. It is a full screen editor.

**Modes of Operation:-**

* Command Mode-

Every character typed is a command that does something to the text file being edited, a character typed in the command mode may even cause the vi editor to the enter the insert mode.

* Insert Mode-

Every character typed is added to the text in the file; pressing the <esc> key turns off the Insert mode.

#### To Start vi:

* vi filename :- edit filename starting at line 1
* vi –r filename :- recover filename that was being edited when the system crashed

***To Exit vi:***

* :q!<Return :- Quit Vi even though the last changes are not saved

### *To Move the Cursor:*

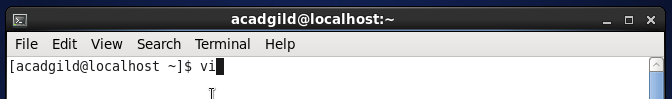
* \* j:- Move cursor down one line

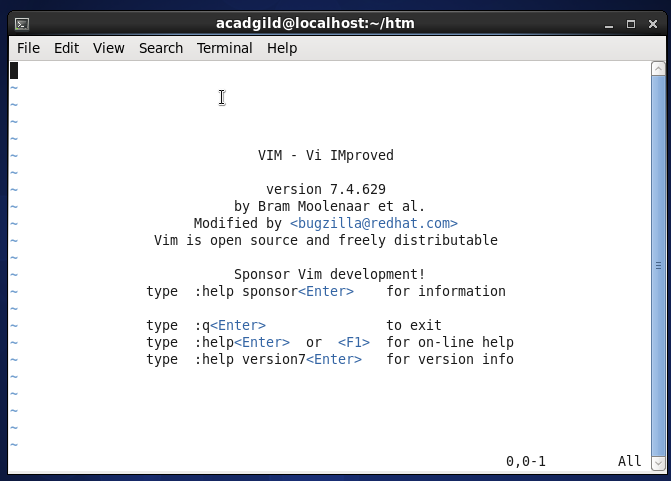
### *Screen Manipulation:*

### ^f: move forward one screen

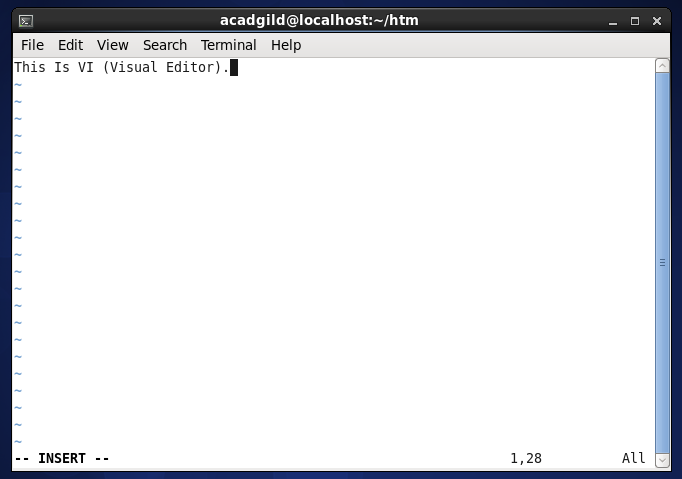
* ^b : move backward one screen
* ^d : move down one screen
* ^u : move up one screen
* ^l : redraw the screen
* ^r : redraw the screen removing deleted lines
* \*u : undo whatever you did
* \*I : insert text before cursor
* \*a : append text after cursor

**Opening vi in Command Mode**:-





**Vi editor in Insert Mode:-**

****

1. **touch**

touch is a standard Linux program used to change a file’s access and modification timestamps. It is also used to create a new empty file. By using touch command we can create single file or more than one file with zero bytes.

**Syntax** touch [-a] [-c] [-m] [-r ref\_file | -t time] file

**touch command options :**

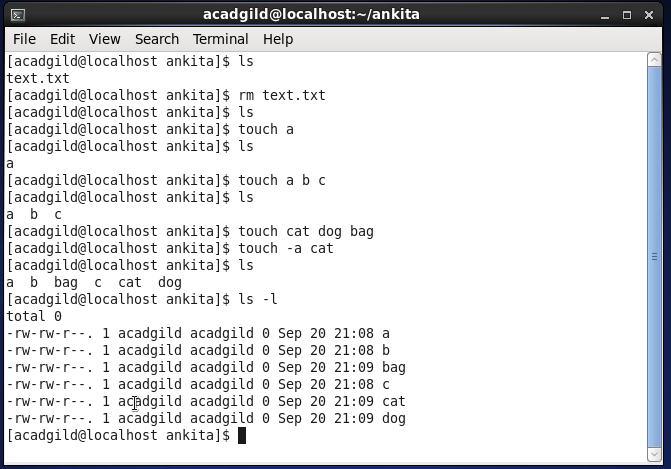
* **-a**

Change the access time of file. Does not change the modification time unless -m is also specified.

* **-c**  
  do not create a specified file if it does not exist. Do not write any diagnostic messages concerning this condition.
* **-m**  
  Change the modification time of file. Do not change the access time unless -a is also specified.
* **-r**  
  ref\_file Use the corresponding times of the file named by ref\_file instead of the current time.
* **-t**  
  time Use the specified time instead of the current time. time will be a decimal number of the form:  
  [[CC]YY]MMDDhhmm [.SS]

MM – The month of the year [01-12].  
DD – The day of the month [01-31].  
hh – The hour of the day [00-23].  
mm – The minute of the hour [00-59].  
CC – The first two digits of the year.  
YY – The second two digits of the year.  
SS – The second of the minute [00-61].

* **-f**  
  ref\_file Use the corresponding times of the file named by ref\_file instead of the current time.
* **file**  
  A path name of a file whose times are to be modified.



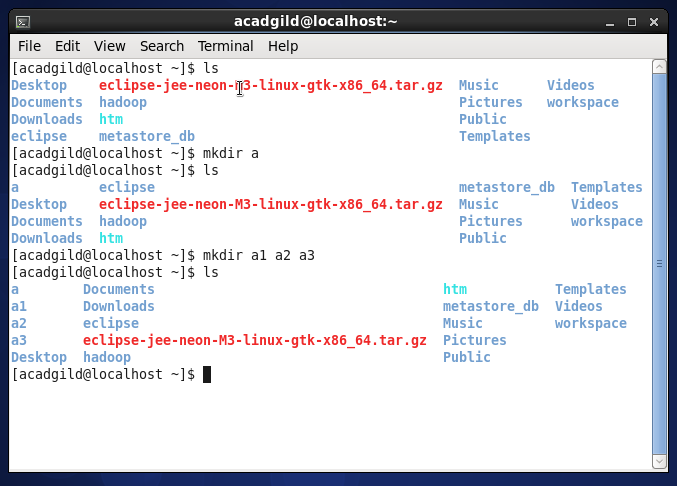
1. **mkdir**

It stands for **Make Directory**. It allows the user to make a new directory. Just like making a new directory within a PC or Mac desktop environment, the mkdir command makes new directories in a Linux environment. This command is used for creating single or multiple directories on a single line of command.

**Syntax :** $ mkdir [Options]

**mkdir command options:**

* - v : Verbose output, print a message for each created directory
* -m mode : sets the access mode for the new directory
* -p : If the parent directory do not exist, this command creates the parent directory

****

1. **rm**

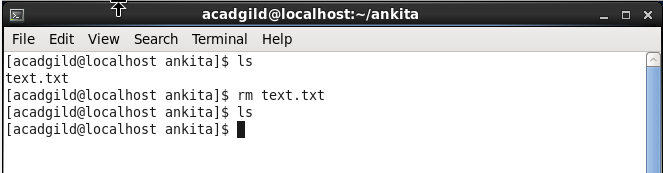
It stands for **Remove** - like the rmdir command is meant to remove files from your Linux OS. Whereas the rmdir command will remove directories and files held within, the rm command will delete created files. An example of the rm command:

**rm testfile.txt**

The aforementioned command removed testfile.txt. Interestingly, whereas the rmdir command will only delete an empty directory, the rm command will remove both files and directories with files in it. This said the rm command carries more weight than the rmdir command and should be used with more specificity.

**rm Command options:**

* (-f): force delete
* (\*): remove all( rm \*) Removes the entire files from the working directory. If it is write protected, you will be prompted before deleting.
* rm –f\* :- Removes all the file from the working directory, even if it is write protected.

****

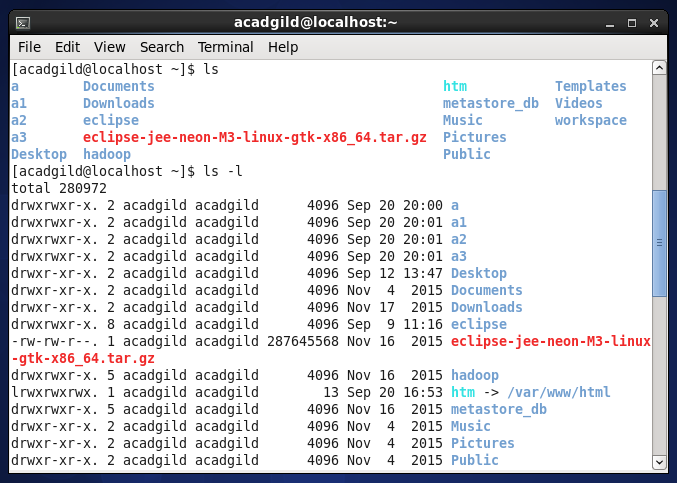
1. **ls**

It stands for **List Command.** It shows all of the major directories filed under a given file system. It is used for viewing files, folders and directories.

**Syntax:** $ ls [options]

**ls command options:**

* ls -S: Sort by file size
* ls –s: list the file size
* ls –r: list in reverse order
* ls –a: list all file including hidden file starting with ‘.’
* ls – l: list with long format show permissions
* ls –t: Sort by time and date
* ls –X: Sort by extension name
* ls ~: list users home directory
* ls..: list parent directory
* ls /: list root directory



1. **echo**

The echo command writes character strings to standard output. Strings are separated by spaces, and a new-line character follows the last String parameter specified. If no String parameter is specified, a blank line (new-line character) is displayed.

**Syntax** echo [OPTIONS] [String …]

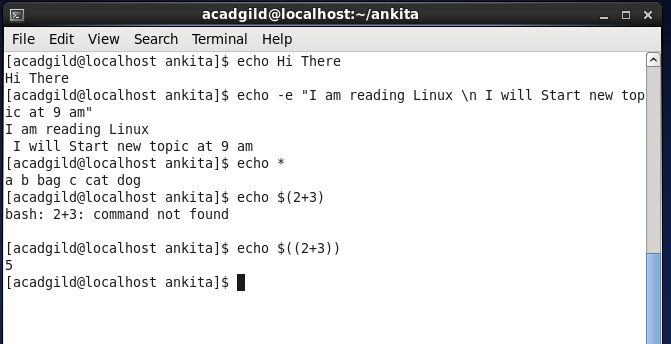
**echo command options:**

* **-n**  
  On BSD and some variants derived from BSD does not begin a new line after the echoed text.
* **-e**  
  on some variants this option is necessary to enable recognition of escape characters in string

**Escape Characters**

The echo command recognizes the following escape conventions:

* **\a** -Displays an alert character.
* **\b -**Displays a backspace character.
* **\c**-Suppresses the new-line character that otherwise follows the final argument in the output. All characters following the \c sequence are ignored.
* **\f-**Displays a form-feed character.
* **\n-**Displays a new-line character.
* **\r-**Displays a carriage return character.
* **\t-**Displays a tab character.
* **\v-**Displays a vertical tab character.
* **\\-**Displays a backslash character.
* **\0Number**  
  Displays an 8-bit character whose ASCII value is a 0-, 1-, 2-, or 3-digit octal number.



1. **cat**

cat” command concatenate files and print on the standard output.

**Syntax**: cat *[OPTION]… [FILE]…*

cat command can be used for the following purposes under UNIX or Linux:  
\* Copy the text-files  
\* Create and combine text files  
\* Show the files on the screen

* Create a new file using the cat command

Syntax: $ cat >filename

Awaits input from the user, type desired text and press ctrl + d to save the file.

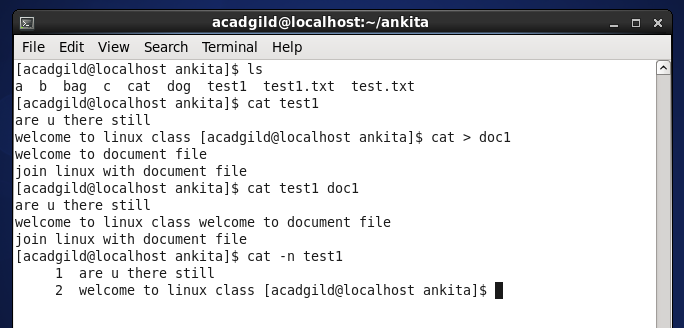
* To show Contents a file using cat command

Syntax : $ cat filename

* Display Line Numbers In file

With –n option we can see the line numbers of the file.

Syntax: $ cat –n filename

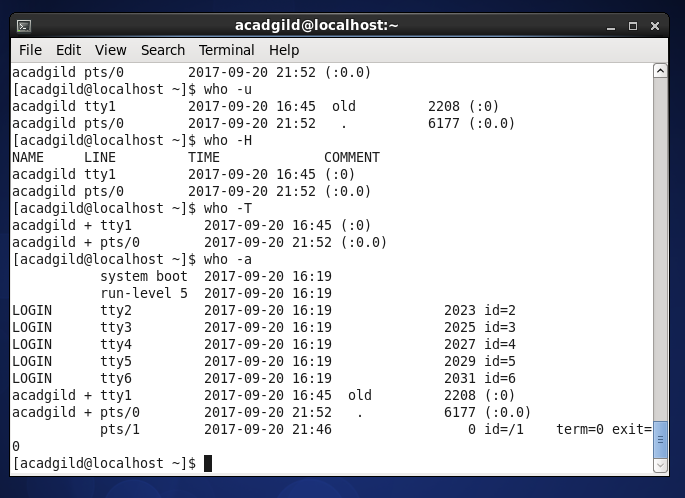


1. **who**

This commands displays about all the users who logged into your system at the time of last system boot or at current run time.

**Syntax:** who [options]

* $ who -To show a list of all the users currently logged in to the system
* -H To display line of column headings
* - m To show only hostname and user associated with stdin (usually keyboard)
* -p To show active processes spawned by init
* -T To show user’s message status as +, – or ?
* -u Show or list users logged in
* -b to display time of last system boot
* -q to count all login names and number of users logged on
* -r to display current run level
* -a to display all information



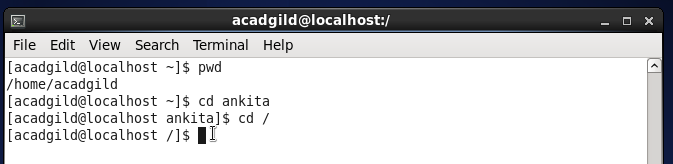
1. **cd**

It stands for **Change Directory.** It allows the user to change between file directories.

**Syntax:** cd [options]

***Cd command options:***

* Change to Home Directory - $ cd $HOME
* Change to root directory- $ cd /



1. **date**

**This command** prints or set the system date and time

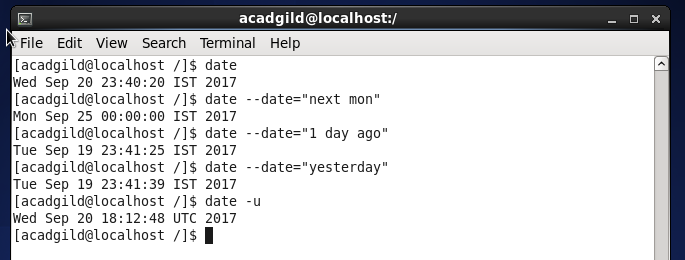
**Syntax:** date [OPTION]... [+FORMAT]

date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

**date commands options:**

|  |  |
| --- | --- |
| **-d, --date=STRING** | display time described by STRING, not 'now'. |
| **-f, --file=DATEFILE** | like --date once for each line of DATEFILE |
| **-r, --reference=FILE** | display the last modification time of FILE |
| **-R, --rfc-2822** | output date and time in RFC 2822 format. Example: Mon, 07 Aug 2006 12:34:56 -0600 |
| **--rfc-3339=TIMESPEC** | output date and time in RFC 3339 format. TIMESPEC='date', 'seconds', or 'ns' for date and time to the indicated precision. Date and time components are separated by a single space: 2006-08-07 12:34:56-06:00. |
| **-s, --set=STRING** | set time described by STRING. |
| **-u, --utc, --universal** | print or set Coordinated Universal Time |

* To Print current system date and time: $ date
* To print date of next Monday: $ date --date="next mon"
* To display past date: $ date --date="1 day ago" OR $ date --date="yesterday"
* To display future date: $ date --date="1 day" OR $ date --date="tomorrow" OR $ date --date="10 day"
* To set date: $ date -s "Sun Dec 18 21:00:00 PDT 2017"
* To display Universal Time: $ date –u
* To display Weekday name: $ date +%a OR $ date +%A
* To display Month name: $ date +%b OR $ date +%B
* To display current day of month: $ date +%d
* To display Current Date in MM/DD/YY format: $ date +%D
* To display date in YYYY-MM-DD format: $ date +%F
* To display time as HH:MM:SS, Note: Hours in 24 Format: $ date +%T



1. **cal**

This command displays a calendar.

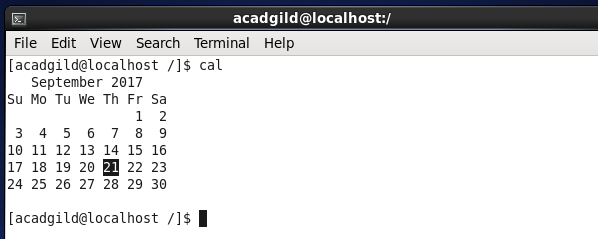
**Syntax:** cal [-mjy][[month]year]

**-**m -- Display Monday as the first day of the week.

**-**j -- Display Julian dates (days one-based, numbered from January 1).

**-**y-- Display a calendar for the current year.

* To display complete year calendar : $ date –y



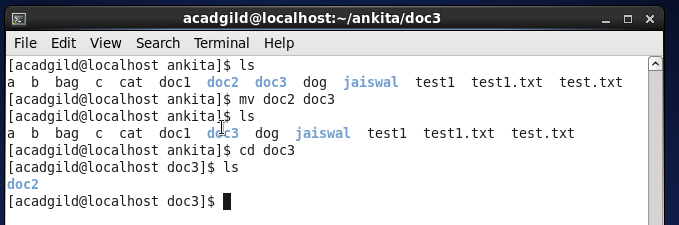
1. **Mv**

It stands for **Move Command** and is used to move file and directories

**Syntax:** $mv [options] source dest

**Mv command options:**

* mv –f ---Force move by overwriting destination file without prompt
* mv – u---Update – move when source is newer than destination
* mv –v ---Verbose – print source and destination files

****

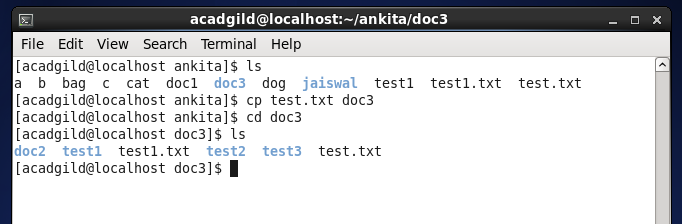
1. **cp**

It stands for **Copy command**. It is used to copy files and directories.

**Syntax:** $cp [options] source dest

**cp command options:**

* cp –a: archive files
* cp – f: force copy by removing destination file if needed
* cp –i: interactive –ask before overwrite
* cp – l: Link files instead of copy
* cp –n: no file overwrites

****

1. **which**

This command is used to find the location of a program

**Syntax:** $ which <programname>

