**Experiment 4**

**Aim: Study basic & User status Unix/Linux Commands: Purpose commands:**  man, help, history, who, whoami, id, uname, uptime, free, tty, cal, date, hostname, reboot, clear, bc.

**1. man command in Linux with Examples**

***man*** command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS and SEE ALSO, of which three are always there Name, Synopsis, and Description.

**Name:** One-line introduction to the command.

**Synopsis**: Syntax used by the command.

**Description**: Provides details related to the command.

**Synopsis:**

If a command argument is enclosed in rectangular brackets, then it is optional; otherwise, the argument is required. The ellipsis implies that there can be more instances of the preceding word. If there is a | character in any of the areas, it means that only one of the options shown on the either side of the pipe can be used.

**EXIT STATUS:**

Lists possible error conditions and their numeric representations.

man –k prints a one-line description of the command.

Every manual is divided into the following sections:

* Executable programs or shell commands
* System calls (functions provided by the kernel)
* Library calls (functions within program libraries
* Games
* Special files (usually found in /dev)
* File formats and conventions eg /etc/passwd
* Miscellaneous (including macro packages and conventions), e.g. groff(7)
* System administration commands (usually only for root)
* Kernel routines [Non standard]

**Syntax :**

$man [OPTION]... [COMMAND NAME]...

**Options and Examples**

**1. No Option**: It displays the whole manual of the command.

**Syntax :**

$ man [COMMAND NAME]

**Example:**

$ man cat

In this example, manual pages of the command ‘*cat*‘ are simply returned.

**2. Section-num**: Since a manual is divided into multiple sections so this option is used to display only a specific section of a manual.

**Syntax :**

$ man [SECTION-NUM] [COMMAND NAME]

**Example:**

$ man 2 intro

In this example, the manual pages of command ‘*intro*‘ are returned which lies in the section 2.

**3. -f option**: One may not be able to remember the sections in which a command is present. So this option gives the section in which the given command is present.

**Syntax:**

$ man -f [COMMAND NAME]

**Example:**

$ man -f ls

In this example, the command ‘*ls*‘ is returned with its section number.

**4. -a option**: This option helps us to display all the available intro manual pages in succession.

**Syntax:**

$ man -a [COMMAND NAME]

**Example:**

$ man -a intro

In this example you can move through the manual pages(sections) i.e either reading(by pressing Enter) or skipping(by pressing ctrl+D) or exiting(by pressing ctrl+C).

**5. -k option**: This option searches the given command as a regular expression in all the manuals and it returns the manual pages with the section number in which it is found.

**Syntax:**

$ man -k [COMMAND NAME]

**Example:**

$ man -k cd

The command ‘*cd*‘ is searched in all the manual pages by considering it as a regular expression.

**6. -w option**: This option returns the location in which the manual page of a given command is present.

**Syntax:**

$ man -w [COMMAND NAME]

**Example:**

$ man -w ls

The location of command ‘*ls*‘ is returned.

**7. -I option**: It considers the command as case sensitive.

**Syntax:**

$ man -I [COMMAND NAME]

**Example:**

$ man -I printf

The command ‘*printf*‘ is taken as case-sensitive i.e ‘*printf*‘ returns the manual pages but ‘*Printf*‘ gives error.

**2. help**

Bash has a built-in help facility available for each of the shell built-ins. To use it, type “help” followed by the name of the shell built-in help command as told before just displays information about shell built-in commands. Here’s the syntax for it:

// syntax for help command

**$help [-dms] [pattern ...]**

The pattern specified in the syntax above refers to the command about which you would like to know and if it is matched with any shell built-in command then **help** give details about it and if it is not matched then **help** prints the list of help topics for your convenience. And the d, m and s here are options that you can use with the **help** command.

**Options for help command**

* **-d option :** It is used when you just want to get an overview about any shell built-in command *i.e* it only gives short description.
* **-m option :** It displays usage in pseudo-manpage format.
* **-s option :** It just displays only a short usage synopsis for each topic matching.

**3. History command in Linux with Examples**

***history***command is used to view the previously executed command. This feature was not available in the Bourne shell. Bash and Korn support this feature in which every command executed is treated as the event and is associated with an event number using which they can be recalled and changed if required. These commands are saved in a history file. In Bash shell **history** command shows the whole list of the command.

**Syntax:**

$ history

Here, the number(termed as event number) preceded before each command depends on the system. You may get different numbers while executing on your own system.

**Important Points**

* To show the limited number of commands that executed previously as follows:

$ history 5

**Note:** The command can be executed using event number also.

*Example:*

$ !1997

* To print command before executing so that a wrong command does not get executed use **:p** after event number of command.

**Example:**

!1997:p

* The most recent command can be viewed using **!!**

**Example:**

!!

* History can also be removed using **history -d event\_number**

**Example:**

history -d 1996

* The whole history can be removed using **history -c**option.

**Example:**

history –c

**4. who command in Linux**

**who** command is used to find out the following information :  
1. Time of last system boot  
2. Current run level of the system  
3. List of logged in users and more.

**Description :**The who command is used to get information about currently logged in user on to system.

**Syntax :** $who [options] [filename]

**Examples :**  
1. The who command displays the following information for each user currently logged in to the system if no option is provided :

1. Login name of the users
2. Terminal line numbers
3. Login time of the users in to system
4. Remote host name of the user

hduser@mahesh-Inspiron-3543:~$ who

hduser tty7 2018-03-18 19:08 (:0)

hduser@mahesh-Inspiron-3543:~$

2. To show list of users logged in to system

hduser@mahesh-Inspiron-3543:~$ who -u

hduser tty7 2018-03-18 19:08 01:16 3357 (:0)

3. To show time of the system when it booted last time

hduser@mahesh-Inspiron-3543:~$ who -b

NAME LINE TIME PID COMMENT

system boot 2018-03-18 19:07

4. To show details of all dead processes

hduser@mahesh-Inspiron-3543:~$ who -d -H (NO dead process in this case)

NAME LINE TIME IDLE PID COMMENT EXIT

NAME LINE TIME IDLE PID COMMENT EXIT

5. To show system login process details

hduser@mahesh-Inspiron-3543:~$ who -l

NAME LINE TIME IDLE PID COMMENT

LOGIN tty1 2018-03-18 19:07 3073 id=tty1

6. To count number of users logged on to system

hduser@mahesh-Inspiron-3543:~$ who -q

hduser

# users=1

7. To display current run level of the system

hduser@mahesh-Inspiron-3543:~$ who -r

run-level 5 2018-03-18 19:07

8. To display all details of current logged in user

hduser@mahesh-Inspiron-3543:~$ who -a

system boot 2018-03-18 19:07

LOGIN tty1 2018-03-18 19:07 3073 id=tty1

run-level 5 2018-03-18 19:07

hduser + tty7 2018-03-18 19:08 01:13 3357 (:0)

**5. whoami**

To display system’s username

hduser@mahesh-Inspiron-3543:~$ whoami

hduser

**6. w**

To display list of users and their activities

hduser@mahesh-Inspiron-3543:~$ w

20:39:20 up 1:32, 1 user, load average: 0.09, 0.06, 0.07

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

hduser tty7 :0 19:08 1:32m 38.95s 0.19s /sbin/upstart -

**7. uname command in Linux with Examples**

The command ‘*uname*‘ displays the information about the system.

**Syntax:**

**uname [OPTION]**

**Options and Examples**

**1. -a option**: It prints all the system information in the following order: *Kernel name*, *network node hostname*, *kernel release date*, *kernel version*, *machine hardware name*, *hardware platform*, *operating system*  
.  
**Syntax:**

$uname -a

**Output:**

**2. -s option**: It prints the kernel name.

**Syntax:**

$uname -s

**3. -n option**: It prints the hostname of the network node(current computer).

**Syntax:**

$uname -n

**4. -r option**: It prints the kernel release date.

**Syntax:**

**$uname -r**

**5. -v option**: It prints the version of the current kernel.

**Syntax:**

$uname -v

**6. -m option**: It prints the machine hardware name.

**Syntax:**

$uname -m

**7. -p option**: It prints the type of the processor.

**Syntax:**

$uname -p

**8. -i option**: It prints the platform of the hardware.

**Syntax:**

$uname -i

**9. -o option**: It prints the name of the operating system.

**Syntax :**

$uname -o

**8. id command in Linux with examples**

**id command** in Linux is used to find out user and group names and numeric ID’s (UID or group ID) of the current user or any other user in the server. This command is useful to find out the following information as listed below:

* User name and real user id.
* Find out the specific Users UID.
* Show the UID and all groups associated with a user.
* List out all the groups a user belongs to.
* Display security context of the current user.

**Synopsis:**

id [OPTION]… [USER]

**Options:**

* ***-g*** : Print only the effective group id.
* ***-G*** : Print all Group ID’s.
* ***-n*** : Prints name instead of number.
* ***-r*** : Prints real ID instead of numbers.
* ***-u*** : Prints only the effective user ID.
* ***–help*** : Display help messages and exit.
* ***–version***: Display the version information and exit.

**Note:** Without any OPTION it prints every set of identified information i.e. numeric ID’s.

**Examples:**

* **To print your own id without any Options:**

id

The output shows the ID of current user UID and GID.

* **To find a specific users id:** Now assume that we have a user named master, to find his UID we will use the command:

id -u master

* **To find a specific users GID:**Again assuming to find GID of master, we will use the command:

id -g master

* **To find out UID and all groups associated with a username:** In this case we will use the user “master” to find UID and all groups associated with it, use command:

id master

* **To find out all the groups a user belongs to:**Displaying the UID and all groups a user “master” belongs to:

id -G master

* **To display a name instead of numbers:** By default the **id** command shows us the UDI and GID in numbers which a user may not understand, with use of -n option with -u, -g and -G, use command(s)
* id -ng master

or

* id -nu master

or

* id -nG master
* **To display real id instead of effective id:** To show the real id with the use of -r option with -g, -u and -G, use command(s):
* id -r -g master
* id -r -u master
* id -r -G master

**9. Uptime command with examples**

* **Uptime Command In Linux:** It is used to find out how long the system is active (running). This command returns set of values that involve, the current time, and the amount of time system is in running state, number of users currently logged into, and the load time for the past 1, 5 and 15 minutes respectively.

**Syntax:**

* uptime [-options]

**Example:**   
Input

* [mistersubha@server-1 ~]$uptime
* 08:24:37 up 207 days, 11:10, 0 users, load average: 0.00, 0.03, 0.05

Just to make sure you get the uptime in human-readable format, you can use option **“p”**

[mistersubha@server-1 ~]$uptime -p

up 29 weeks, 4 days, 11 hours, 1 minute

Option **“s”** is used to get the starting time/specified time when the system started has been running.

[mistersubha@server-1 ~]$uptime -s

2017-11-10 20:14:15

**10. free Command in Linux with examples**

While using LINUX there might come a situation when you are willing to install a new application (*big in size*) and you wish to know for the amount of free memory available on your system. In LINUX, there exists a command line utility for this and that is **free**command which displays the total amount of free space available along with the amount of memory used and swap memory in the system, and also the buffers used by the kernel.

This is pretty much what free command does for you.  
**Syntax:**

$free [OPTION]

**OPTION :** refers to the options compatible with free command.

As free displays the details of the memory related to your system , its syntax doesn’t need any arguments to be passed but only options which you can use according to your wish.

**Using free Command**

You can use the free command as:

// using free command

**$free**

total used free shared buffers cached

Mem: 509336 462216 47120 0 71408 215684

-/+ buffers/cache: 175124 334212

Swap: 915664 11928 903736

/\*free command without any option shows the used and free space of swap and physical memory in **KB** \*/

When no option is used then free command produces the columnar output as shown above where column:

1. **total** displays the total installed memory (MemTotal and SwapTotal *i.e* present in /proc/meminfo).
2. **used** displays the used memory.
3. **free** displays the unused memory.
4. **shared** displays the memory used by tmpfs(Shmen *i.e*present in /proc/meminfo and displays zero in case not available).
5. **buffers** displays the memory used by kernel buffers.
6. **cached** displays the memory used by the page cache and slabs(Cached and Slab available in /proc/meminfo).
7. **buffers/cache** displays the sum of buffers and cache.

**Options for free command**

* **-b, – -bytes :** It displays the memory in bytes.
* **-k, – -kilo :** It displays the amount of memory in kilobytes(default).
* **-m, – -mega :** It displays the amount of memory in megabytes.
* **-g, – -giga :** It displays the amount of memory in gigabytes.
* **– – tera :** It displays the amount of memory in terabytes.
* **-h, – -human :** It shows all output columns automatically scaled to shortest three digit unit and display the units also of print out. The units used are **B**(bytes), **K**(kilos), **M**(megas), **G**(gigas), and **T**(teras).
* **-c, – -count :** It displays the output **c** number of times and this option actually works with **-s** option.
* **-l, – -lohi :** It shows the detailed low and high memory statistics
* **-o, – -old :** This option disables the display of the buffer adjusted line.
* **-s, – -seconds :** This option allows you to display the output continuously after **s** seconds delay. In actual, the **usleep**system call is used for microsecond resolution delay times.
* **-t, – -total :** It adds an additional line in the output showing the column totals.
* **– -help :** It displays a help message and exit.
* **-V, – -version :** It displays version info and exit.

**Using free command with options**

**1. Using -b :** It just displays the output in unit bytes.

//using free with -b

**$free -b**

total used free shared buffers cached

Mem: 521560064 474198016 47362048 0 73826304 220983296

-/+ buffers/cache: 179388416 342171648

Swap: 937639936 12210176 925429760

/\*everything now displayed is in bytes \*/

**2. Using -k :** This option displays the result in kilobytes.

//using free with -k

**$free -k**

total used free shared buffers cached

Mem: 509336 463084 46252 0 72104 215804

-/+ buffers/cache: 175176 334160

Swap: 915664 11924 903740

/\*no change in output if compared to only free command output cause this is the by default format that free uses for the result \*/

**3. Using -m :** This option displays the result in megabytes.

//using free with -m

**$free -m**

total used free shared buffers cached

Mem: 497 452 45 0 70 210

-/+ buffers/cache: 171 326

Swap: 894 11 882

/\*everything now displayed is in megabytes \*/

**4.using -g :** This option displays the result in gigabytes.

//using free with -g

**$free -g**

total used free shared buffers cached

Mem: 0 0 0 0 0 0

-/+ buffers/cache: 0 0

Swap: 0 0 0

/\*everything now displayed is in gigabytes \*/

**5. Using -t (total) :** This option displays an additional line containing the total of the total, used and free columns.

//using free with -t

**$free -t**

total used free shared buffers cached

Mem: 509336 463332 46004 0 72256 215804

-/+ buffers/cache: 175272 334064

Swap: 915664 11924 903740

Total: 1425000 475256 949744

/\*the line containing total is added to the output when -t is used\*/

**6. Using -o :** This option makes the buffer/cache line go away from the output as shown below.

//using free with -o

**$free -o**

total used free shared buffers cached

Mem: 509336 463588 45748 0 72376 215856

Swap: 915664 11924 903740

/\*now the output doesn't have the buffer line in it \*/

**11. tty command in Linux with examples**

Linux operating system represents everything in a file system, the hardware devices that we attach are also represented as a file. The terminal is also represented as a file. There a command exists called**tty**which displays information related to **terminal**. The **tty** command of terminal basically prints the file name of the terminal connected to standard input. **tty is short of teletype**, but popularly known as a terminal it allows you to interact with the system by passing on the data (you input) to the system, and displaying the output produced by the system.

**Syntax:**

tty [OPTION]....

**12. cal command in Linux with Examples**

If a user wants a quick view of calendar in Linux terminal, cal is the command for you. By default, cal command shows current month calendar as output.

**cal** command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.

**Syntax:**

cal [ [ month ] year]

Rectangular bracket means it is optional, so if used without option, it will display a calendar of current month and year.

* **cal :** Shows current month calendar on the terminal.
* **cal 08 2000 :** Shows calendar of selected month and year.
* **cal 2018 :** Shows the whole calendar of the year.
* **cal -3 :** Shows calendar of previous, current and next month

**13. date command in Linux with examples**

**date**command is used to display the system date and time. date command is also used to set date and time of the system. By default the date command displays the date in the time zone on which unix/linux operating system is configured.You must be the super-user (root) to change the date and time.

**Syntax:**

date [OPTION]... [+FORMAT]

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

**Options with Examples**  
1: **date (no option) :**With no options, the date command displays the current date and time, including the abbreviated day name, abbreviated month name, day of the month, the time separated by colons, the time zone name, and the year.

**Command:**

$date

**Output:**

Tue Oct 10 22:55:01 PDT 2017

**Note :**Here unix system is configured in pacific daylight time

.

2:**-u Option:**Displays the time in GMT(Greenwich Mean Time)/UTC(Coordinated Universal Time )time zone.

**Command:**

$date -u

**Output :**

Wed Oct 11 06:11:31 UTC 2017

3: **–date or -d Option:**Displays the given date string in the format of date. But this will not affect the system’s actual date and time value.Rather it uses the date and time given in the form of string.  
**Syntax:**

**$date --date=" string "**

**Command:**

$date --date="2/02/2010"

$date --date="Feb 2 2010"

**Output:**

Tue Feb 2 00:00:00 PST 2010

Tue Feb 2 00:00:00 PST 2010

4:**Using –date option for displaying past dates:**

 Date and time of 2 years ago.

**Command:**

$date --date="2 year ago"

**Output:**

Sat Oct 10 23:42:15 PDT 2015

 Date and time of 5 seconds ago.

**Command:**

$date --date="5 sec ago"

**Output:**

Tue Oct 10 23:45:02 PDT 2017

 Date and time of previous day.

**Command:**

$date --date="yesterday"

**Output:**

Mon Oct 9 23:48:00 PDT 2017

 Date and time of 2 months ago.

**Command:**

$date --date="2 month ago"

**Output:**

Thu Aug 10 23:54:51 PDT 2017

 Date and time of 10 days ago.

**Command:**

$date --date="10 day ago"

**Output:**

Sat Sep 30 23:56:55 PDT 2017

5:**Using –date option for displaying future date:**

 Date and time of upcoming particular week day.

**Command:**

$date --date="next tue"

**Output:**

Tue Oct 17 00:00:00 PDT 2017

 Date and time after two days.

**Command:**

$date --date="2 day"

**Output:**

Fri Oct 13 00:05:52 PDT 2017

 Date and time of next day.

**Command:**

$date --date="tomorrow"

**Output:**

Thu Oct 12 00:08:47 PDT 2017

 Date and time after 1 year on the current day.

**Command:**

$date --date="1 year"

**Output:**

Thu Oct 11 00:11:38 PDT 2018

6:**-s or –set Option:**To set the system date and time -s or –set option is used.  
**Syntax:**

**$date --set="date to be set"**

**Command:**

$date

**Output:**

Wed Oct 11 15:23:26 PDT 2017

**Command:**

$date --set="Tue Nov 13 15:23:34 PDT 2018"

$date

**Output:**

Tue Nov 13 15:23:34 PDT 2018

9: **List of Format specifiers used with date command:**

**%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.

**Syntax:**

**$date +%[format-option]**

**Examples:**

**Command:**

$date "+%D"

**Output:**

10/11/17

**Command:**

$date "+%D %T"

**Output:**

10/11/17 16:13:27

**Command:**

$date "+%Y-%m-%d"

**Output:**

2017-10-11

**Command:**

$date "+%Y/%m/%d"

**Output:**

2017/10/11

**Command:**

$date "+%A %B %d %T %y"

**Output:**

Thursday October 07:54:29 17

# 14. hostname command in Linux with examples

*hostname* command in Linux is used to obtain the DNS(Domain Name System) name and set the system’s hostname or NIS(Network Information System) domain name. A hostname is a name which is given to a computer and it attached to the network. Its main purpose is to uniquely identify over a network.

**Syntax :**

hostname -[option] [file]

**Example:** We obtain the system hostname by just typing hostname without any attributes.

**15. reboot command in Linux with Examples**

**reboot** command is used restart or reboot the system. In a Linux system administration, there comes a need to restart the server after the completion of some network and other major updates. It can be of software or hardware that are being carried on the server. The reboot is needed so that the changes that the user have done can be affected on the server. For example, if the user is re-compiling the server’s kernel that is going through some more advanced server administration, then he needs to restart the machine in order to complete the compilation and to have a new updated kernel version on the server. When Updating the server’s memory, IP allocation, NIC configuration are the key tasks that need to be done on the server restarted once leading to their successful implementation. Most of the Linux system administrators access their servers via shell or SSH to perform a bunch of administrative activities, server management, and monitoring. So they need to know the basic commands to restart the server from the shell.

**Syntax:**

reboot [OPTIONS...]

**Options:**

* **–help :** This option prints a short help text and exit.
* **–halt :** This option halt the machine, regardless of which one of the three commands is invoked.
  + **-p, –poweroff :** This option will going to power-off the machine, regardless of which one of the three commands is being invoked.
  + **–reboot :** This option reboot the machine, regardless of which one of the three commands is invoked.
  + **-f, –force :**This option force immediate halt, power-off, or reboot. When it is specified once, this results in the immediate but clean shutdown by the system manager. When it is specified twice, this results in immediate shutdown without contacting the system manager. See the description of the option –force in systemctl(1) for more details.
* **-w, –wtmp-only :**This option only writes *wtmp*shutdown entry, it do not actually halt, power-off, reboot.

**Examples:**

**Restart your system:** If you just need is a restart without going into any details just help yourself by using any one of the following commands:

$sudo reboot

$sudo shutdown –r now

**Note**that the usage of the reboot, halt and poweroff is almost similar in syntax and effect. Run each of these commands with –help to see the details.

**Scheduled a Restart:**Simple reboot command has limited usage. The shutdown command is being used instead of reboot command to fulfill much more advance reboot and shutdown requirements. One of those situations is a scheduled restart. Following is the syntax which is being used to reboot your system after time defines by the TIME.

$sudo shutdown –r [TIME] [MESSAGE]

Here the TIME has various formats. The simplest one is “now”, already been listed in the previous section, and tells the system to restart immediately. Other valid formats we have are *+m*, where m is the number of minutes we need to wait until restart and HH:MM which specifies the TIME in a 24hr clock. Below are the examples and their respective outputs. Optional MESSAGE argument can be used to intimate the users prior to reboot to prevent the possible loss of data.

**Command to reboot your system after 5 minutes:**

$sudo shutdown –r +5

**Reboot your system after 02:00 A.M :**

$sudo shutdown –r 02:00

**Checking your reboot logs:***/var/log/wtmp* is the file records in which all logins and logouts records are kept. One can parse this file with the last command in order to access log for the reboot. Below you can find the last command usage and its output on my system.

$last reboot

**16. clear command in Linux with examples**

*clear*is a standard Unix computer operating system command that is used to clear the terminal screen. This command first looks for a terminal type in the environment and after that, it figures out the *terminfo*database for how to clear the screen. And this command will ignore any command-line parameters that may be present. Also, the *clear*command doesn’t take any argument and it is almost similar to *cls* command on a number of other operating systems.

**Syntax:**

**$clear**

**17. bc command in Linux with examples**

**bc** command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations.  
Arithmetic operations are the most basic in any kind of programming language. Linux or Unix operating system provides the **bc command** and **expr command** for doing arithmetic calculations. You can use these commands in bash or shell script also for evaluating arithmetic expressions.

**Syntax:**

**bc [ -hlwsqv ] [long-options] [ file ... ]**