

## Unix Miscellaneous commands

- **ls**
- **pwd**
- **ln**
- **head**
- **tail**
- **cal**
- **ps**
- **kill**
- **who**
- **whoami**
- **uptime**
- **ut**

**ls** : This command displays files and directories in columnar format.

### Example:

\$ ls

\$ ls -S

Arrange the files based on the size(S is upper letter)

\$ ls -l

long listing the files

\$ ls -a

Displays hidden files

\$ ls -i

Displays inodes for each file

\$ ls -R

Displays all directories along with subdirectories in current working directory.

### Creating a hidden file:

\$ cat >.employee.txt

### Wild card characters using with ls command.

? Represents single character

\* Represents group of characters

[ ] Represents searching pattern

### Examples:

\$ ls ?	Displays files with one letter
\$ ls x*	Displays files which are starting with 'x'
\$ ls ???	Displays files with 3 letters
\$ ls *.out	Displays all the files with extension 'out'
\$ ls [a-z]	Displays single character files which are from a to z
\$ ls [a-z]*	Displays files starting with a to z
\$ rm ?	Removes the files with single character
\$ rm *.c	Removes the files with extension 'c'
\$ cp ? chennai	Single digit files will be copied into directory 'chennai'

**pwd :** This command shows current working directory in unix

\$ pwd

**head :** Used to display First lines of the file

Syntax:

\$ head -n filename

**Example:**

\$ head paypal.txt

This command displays default 10 lines of the file paypal.txt

\$ head -n15 paypal.txt

This command displays 15 lines of the file paypal.txt

\$ head -n3 paypal.txt

This command displays 3 lines of the file paypal.txt

**tail :** Used to display last lines of the file.

Ex:

\$ tail paypal.txt

This command displays default last 9 lines.

```
$ tail -n5 paypal.txt
```

This command displays last 4 lines of the file.

```
$ tail -n15 paypal.txt
```

This command displays last 14 lines of the file.

**cal:** Used for display the calendar.

Syntax:

```
$ cal
```

```
$ cal [Month No] [Year]
```

Ex:

```
$ cal 2011           Displays 2011 calendar
```

```
$ cal 2 2011         Displays Feb month in 2011 year
```

**ln :**

- 1.This will create link between 2 files.(2nd file should new one)
2. If one file modified another one affected.

Ex:

```
$ln paypal.txt  funpal.txt
```

**ps :** Knowing background process running /stopped

```
$ ps pid
```

**kill :** Used for terminating the process

Ex:

```
$ kill pid
```

**who :** Used for to display who are working in the system

Ex:

```
$ who
```

**whoami:** Display my user name/account

Ex:

```
$ whoami
```

**uptime:** This command used for display load in the server.

```
$ uptime
```

**ut:** displays current unix time

**advance\_date :** Used to time forward the unix server.

```
$ advance_date
```

**reset\_time.py :** Used to reset time and date in unix server.

```
$ reset_time.py
```

## UNIX File Access Permissions

**chmod:** It provides permissions over a file in 3 categories.

- 1) owners
- 2) groups
- 3) others

Permissions which can be granted are read, write and execute

- 1) read (r)
- 2) write (w)
- 3) execute (e)

These permissions are represented with numeric values

```
r - 4
w - 2
e - 1
-----
    7
-----
```

**Owners** are users whose files gets referred from their respective accounts.

**Groups** are users whose accounts are dependent on the other accounts.

**Others** are users who can access the files of other users.

**Chmod** command is used to change the permissions for a file or directory.

Syntax:

\$ chmod FAP Filename

\* FAP is file access permissions

### **Examples:**

**\$ chmod 000 paypal.txt**

No permissions to owners,groups and others

**\$ chmod 777 paypal.txt**

All permissions to owners,groups and others

**\$ chmod 444 paypal.txt**

Read permission to owners,groups and others (4 - read)

**\$ chmod 600 paypal.txt**

Read(4),write(2) permissions to owners, no permissions to groups and others

**\$ chmod 664 paypal.txt**

Read,write permissions to owners,groups and read permission to others

**\$ chmod 111 paypal.txt**

Execute permission to owners,groups and others

### **Change permissions using name of the permission:**

#### **Examples:**

**\$ chmod u-w g-w o-r paypal.txt**

- \* write permission cancelled from owner
- \* write permission cancelled from groups
- \* read permission cancelled from others

**\$ chmod u+rw g+rw o+r paypal.txt**

- \* read,write,execute permissions added to owners
- \* read,write,execute permissions added to groups
- \* read permission added to others

- \*\* Note:** + used for giving permissions.  
- is used for removing permissions.

## UNIX Filter Commands

- **grep**
- **sort**
- **more**
- **cut**
- **wc**
- **uniq**

### **grep:** (Global Regular Expression Pattern)

**This command is used for searching a required pattern in a file.**

#### **Syntax:**

```
$ grep [- option] "search pattern" Filename [redirection symbol newfilename]
```

#### **Options:**

- i** Ignores case sensitiveness in searching pattern
- n** displays line numbers for those lines which gets matched and un matched with the pattern
- c** counts number of times a searching pattern exists and does not exists
- v (verbose)** Displays those lines that does not match with the pattern

#### **Example:**

```
$ cat > paypal.txt
welcome to unix
paypal welcomes you
unix multi user os
WELCOME to the world of unix
```

```
$ grep "welcome" paypal.txt
$ grep -i "welcome" paypal.txt
$ grep -i -n "welcome" paypal.txt
$ grep -i -c "welcome" paypal.txt
$ grep -i -v "welcome" paypal.txt
$ grep -i -v -n "welcome" paypal.txt > funpal.txt
```

**Sort :** Used to arrange numbers/text in ascending/descending order.  
**\* by default it arranges ascending order.**

#### **Syntax:**

```
$ sort [-option] [redirection symbol ] filename [redirection symbol] [new filename]
```

### options:

**-r** Arrange data in reverse or descending order

**-n** Arrange data in ascending or descending order by considering whole number.

\* if n is not used then numbers gets arranged in order based on 1st digit.

### Ex1:

```
$ sort > paypal.txt
```

```
6
2
9
1
5
3
```

```
$ cat paypal.txt    Ascending order
```

```
$ sort -r paypal.txt > funpal.txt    Descending order
```

```
$cat funpal.txt
```

### Ex2:

```
$ sort >google.txt
```

```
176
2165
8
93
```

```
-----
```

```
----- [ctrl+d]
```

```
$cat google.txt    It considered first digit
```

```
$sort -n google.txt > yahoo.txt    It considered whole number.
```

**more: This filter command used to display information from multiple files based n page wise.**

It gives an identification of end of file for first file and beginning of next file.

Syn:

```
$more [-option] file1,file2,file3, etc...
```

**options:**

**-p** clears the screen and displays next file in the list of files

**\*note:**

Enter key retrieves next file based on %s  
spacebar retrieves complete data from next file

Ex: \$more - p paypal.txt funpal.txt

**cut:** Used to cut the required text from a file. It can cut the data on the columns and fields.

**Syntax:**

\$ cut [-option] filename [redirection symbol new filename]

**-c** To cut the data in columns

**-f** To cut the data in fields that is that data which is separated by tab.

**Ex1:**

```
$ cat>paypal.txt
Hyderabad
Secunderabad
Andhra [ctrl+d]
```

```
$ cut -c1 paypal.txt [Enter]
H
S
A
```

```
$cut -c3 paypal.txt [Enter]

d
c
d
```

```
$cut -c1 -3 paypal.txt [Enter]

Hyd
Sec
And
```

**Ex2:**



```
$cat > funpal.txt [Enter]
```

```
India    Delhi
```

```
Andhra   Hyderabad
```

```
Peers    Net
```

```
[ctrl+d]
```

```
$ cut -f1 funpal.txt [Enter]
```

```
India
```

```
Andhra
```

```
Peers
```

```
$ cut -f2 funpal.txt [Enter]
```

```
Delhi
```

```
Hyderabad
```

```
Net
```

**wc:** It will count number of lines, worlds, characters in a file.

### Syntax:

```
$ wc [-option] filename
```

### options:

**-l** count number of lines

**-w** count number of words

**-c** count number of characters

### Ex:

```
$ wc -l paypal.txt
```

```
$ wc -w paypal.txt
```

```
$ wc -c paypal.txt
```

**uniq:** This filter is used to get the uniq or duplicate lines from a file.  
Data should be in order.

Syntax: \$ uniq [-option] filename

### options:

**-d** Display duplicate lines

**-u** Display uniq lines

**-c** Counts number of times each word has occurred in a file

### Ex:

```
$ cat > city.txt
ameerpet
ameerpet
peers
bhel
hyderabad
ameerpet
peers
secunderabad [ctrl+d]
```

```
$ sort city.txt > city1.txt
$ cat city1.txt    Ascending order of data displaying
```

```
$ uniq -u city1.txt
$ uniq -d city1.txt
$ uniq -c city1.txt
```

## Unix File compare commands

- **cmp**
- **diff**
- **comm**

**cmp:** It compares 2 files. If files are same it returns prompt or else it returns the message where the difference encountered.

Syn: \$ cmp file1 file2

Ex: \$ cmp paypal.txt funpal.txt

**diff:** This command compares 2 files like cmp. If any difference found in 2 files it displays those lines.

Ex:

```
$ diff paypal.txt funpal.txt
```

**comm :** Used to compare 2 sorted files .It provides output in 3 columns.

- \* In first column displays uniq lines of first file.
- \* In second column displays uniq lines of second file
- \* In third column displays common lines in 2 files.

**Ex:-**

**Step1 :** Create two files with some data

```
$ cat> paypal.txt  
risk  
payments  
ebay  
uv  
norkom
```

```
$ cat> funpal.txt  
football  
cricket  
crems  
ebay  
payments
```

**Step2:** Sort above files and store the data into another 2 different files

```
$ sort paypal.txt paypal1.txt [Enter]  
$ sort funpal.txt funpal1.txt [Enter]
```

**Step3:** Use comm command

```
$ comm paypal1.txt funpal1.txt [Enter]  
** output shows in 3 different columns.
```

### [FAQ's](#)

#### **1) How to find hidden files in current directory?**

```
$ ls -a
```

#### **2) How to find current running processes in Unix server?**

```
$ ps -ef
```

and if we want to find specific process we can use 'grep' with pipe

```
$ ps -ef | grep -i 'application'
```

#### **3) How to find process which is taking maximum memory in server?**

```
$ top
```

top command tell us about cpu usage , process id and other details.

**4) How to find Exception in log files available in current directory and how to find number of occurrence?**

```
$ grep 'Exception' log1.txt | wc -l
```

**5) find all files in current and subdirectories which contains 'log' name?**

```
$ find . -name 'log'
```

**6) How do you access command line arguments from within a shell script?**

Arguments passed from the command line to a shell script can be accessed within the shell script by using a \$ (dollar sign) immediately followed with the argument's numeric position on the command line.

**7) How to tails last 200 lines of any log file?**

```
$ tail -200f filename.txt
```

**8) How to find remaining disk space in unix/linux server?**

```
$ df -kl
```

**9) How to make any script file executable?**

```
$chmod 755 *.sh
```

**10) How to kill process in unix server?**

```
$ kill -9 #pid
```

these #pid can be found using ps -ef command.

**11) Command to run a job say job.sh in background?**

```
.job.sh &
```

**12) How to check a process named Informatica is running or not?**

```
Ps -ef|grep Informatica
```

**13) Command to force any process to stop execution?**

```
Kill -9 pid (process id)
```

**14) Command to get the present directory you are working?**

Pwd

**15) Ipconfig equivalent in unix or command which gives all the network related details?**

Ifconfig