**1) cat**  
: Used to view files or concatenate files.  
  
**Cat /etc/passwd** - this command displays the “/etc/passwd/” file on the screen.  
**Cat file1 file2 file3 > file4** - Combines the contents of the first 3 files into 4th file.  
  
The **drawback** of the "cat"command (when displaying file contents to your screen) is that the contents of the file may scroll off of the screen. In cases where a file is to large to fit on the screen, you're better off using the "more" command to display the file. In fact, it's probably easier to use the more command all the time, and just use the cat command to concatenate (merge) files.  
  
**To append file1 onto the end of file2, enter:**  
cat file1 >> file2  
  
**To combine several text files into a single file in Unix, use the cat command:**  
  
cat file1 file2 file3 > newfile  
  
**If you want to add one or more files to an existing document, use the format:**  
  
cat file1 file2 file3 >> destfile  
  
**Note: If you use > instead of >>, you will overwrite destfile rather than add to it.**  
  
**2) more**  
: It displays files one screenful at a time.  
  
When more has control of your screen, you can hit the spacebar to display the next screenful of information, use the '/' to search for text, or type 'q' to quit.  
  
**more /etc/passwd** - this command displays the /etc/passwd file one screenful at a time.  
  
**ps -ef more**  
This command displays the output of the ps -ef command one screenful at a time.  
  
**3) cd**  
: "cd" stands for change directory. It is the primary command for moving around the file system.  
**cd /usr**  
This command moves you to the "/usr" directory. "/usr" becomes your current working directory.  
  
**cd /usr/fred**  
Moves you to the "/usr/fred" directory.  
  
**cd /u\*/f\***  
Moves you to the "/usr/fred" directory - if this is the only directory matching this wildcard pattern.  
  
**cd**  
Issuing the "cd" command without any arguments moves you to your home directory.  
  
**cd -**  
Using the Korn shell, this command moves you back to your previous working directory. This is very useful when you're in the middle of a project, and keep moving back-and-forth between two directories.  
  
**4) cp**  
: The "cp" command is used to copy files and directories.  
Note that when using the cp command, you must always specify both the source and destination of the file(s) to be copied.  
  
  
**cp .profile .profile.bak**  
This command copies your ".profile" to a file named ".profile.bak".  
  
**cp /usr/fred/Chapter1 .**  
This command copies the file named "Chapter1" in the "/usr/fred" directory to the current directory. This example assumes that you have write permission in the current directory.  
  
**cp /usr/fred/Chapter1 /usr/prasad**  
This command copies the "Chapter1" file in "/usr/fred" to the directory named "/usr/prasad". This example assumes that you have write permission in the "/usr/prasad" directory.  
  
**5) file**  
: The "file" command shows you information about the file type of the files you specify - i.e., whether the file contains text or binary data, whether the file is an executable file, etc.  
  
**file \***  
This command displays file type information for every file in the current directory.  
  
**file /etc/\***  
This command tells you information about every file in the /etc directory.  
  
**6) ls**  
: "ls" stands for list. It is used to list information about files and directories.  
  
**ls**  
This is the basic "ls" command, with no options. It provides a very basic listing of the files in your current working directory. Filenames beginning with a decimal are considered hidden files, and they are not shown.  
  
**ls -a**  
The -a option tells the ls command to report information about all files, including hidden files.  
  
**ls -l**  
The -l option tells the "ls" command to provide a long listing of information about the files and directories it reports. The long listing will provide important information about file permissions, user and group ownership, file size, and creation date.  
  
**ls -al**  
This command provides a long listing of information about all files in the current directory. It combines the functionality of the -a and -l options. This is probably the most used version of the ls command.  
  
**ls -al /usr**  
This command lists long information about all files in the "/usr" directory.  
  
**ls -alR /usr more**  
This command lists long information about all files in the "/usr" directory, and all sub-directories of /usr. The -R option tells the ls command to provide a recursive listing of all files and sub-directories.  
  
  
**ls -ld /usr**  
Rather than list the files contained in the /usr directory, this command lists information about the /usr directory itself (without generating a listing of the contents of /usr). This is very useful when you want to check the permissions of the directory, and not the files the directory contains.  
  
**options**  
**ls** will list all the files in your home directory, this command has many options.  
**ls -l** will list all the file names, permissions, group, etc in long format.  
**ls -a** will list all the files including hidden files that start with . .  
**ls -lt** will list all files names based on the time of creation, newer files bring first.  
**ls –Fx** will list files and directory names will be followed by slash.  
**ls –R** will lists all the files and files in the all the directories, recursively.  
**ls -R more** will list all the files and files in all the directories, one page at a time.  
**-d** Shows directory names, but not contents  
**-F** Marks special files with symbols to indicate what they are: / for directories, @ for symbolic links, \* for executable programs  
**If you want to mail a list of the files in your directory to a user named tom, you would use the following combination:**ls Mail tom  
  
**7) mkdir**  
: The "mkdir" command is used to create new directories (sub-directories).  
  
**mkdir tmp**This command creates a new directory named "tmp" in your current directory. (This example assumes that you have the proper permissions to create a new sub-directory in your current working directory.)  
  
**mkdir memos letters e-mail**  
This command creates three new sub-directories (memos, letters, and e-mail) in the current directory.  
  
**mkdir /usr/fred/tmp**  
This command creates a new directory named "tmp" in the directory "/usr/fred". "tmp" is now a sub-directory of "/usr/fred". (This example assumes that you have the proper permissions to create a new directory in /usr/fred.)  
  
**mkdir -p /home/joe/customer/acme**This command creates a new directory named /home/joe/customer/acme, and creates any intermediate directories that are needed. If only /home/joe existed to begin with, then the directory "customer" is created, and the directory "acme" is created inside of customer.  
  
**8) mv**  
: The "mv" command is used to move and rename files.  
  
**mv Chapter1 Chapter1.bad**  
This command renames the file "Chapter1" to the new name "Chapter1.bad".  
  
**mv Chapter1 garbage**This command renames the file "Chapter1" to the new name "garbage". (Notice that if "garbage" is a directory, "Chapter1" would be moved into that directory).  
  
**mv Chapter1 /tmp**  
This command moves the file "Chapter1" into the directory named "/tmp".  
  
**mv tmp tmp.old**Assuming in this case that tmp is a directory, this example renames the directory tmp to the new name tmp.old.  
  
**mv -f oldfile newfile** will force the rename even if target file exists.  
  
**9) pwd**  
: "pwd" stands for print working directory. It displays your current position in the UNIX file system.  
  
**10) rm**  
: The "rm" command is used to remove files and directories.  
  
**rm Chapter1.bad**  
This command deletes the file named "Chapter1.bad" (assuming you have permission to delete this file).  
  
**rm Chapter1 Chapter2 Chapter3**  
This command deletes the files named "Chapter1", "Chapter2", and "Chapter3".  
  
**rm -i Chapter1 Chapter2 Chapter3**  
This command prompts you before deleting any of the three files specified. The -i option stands for inquire. You must answer y (for yes) for each file you really want to delete. This can be a safer way to delete files.  
  
**rm \*.html**  
This command deletes all files in the current directory whose filename ends with the characters ".html".  
  
**rm index\***  
This command deletes all files in the current directory whose filename begins with the characters "index".  
  
**rm -r dir1**  
This command deletes the directory named "dir1". This directory, and all of it’s' contents, are erased from the disk, including any sub-directories and files.  
  
**rm -f** option will remove write-protected files without prompting.  
  
insert a backslash ( \ ) before the meta-character in your filename. The backslash causes the character that follows to be interpreted literally. For example, to remove the file named my$project, enter:  
**rm my\$project**  
To remove a file whose name begins with a dash ( - ) character, refer to the file with the following syntax:  
**rm ./-filename**  
**11) wc**  
: The "wc" command stands for "word count". It counts the number of characters, words, and lines that are contained in a text stream.  
  
**wc /etc/passwd**This command tells you the number of characters, words, and lines in the /etc/passwd file.  
  
**wc -l /etc/passwd**  
This command tells you the number of lines (only) in the /etc/passwd file.  
  
**wc -w MyStory**This command counts the number of words in the file named MyStory (which can be useful if you're paid by the word!).  
  
**who wc -l**  
This command counts the number of users logged into your computer system. The output of the who command is piped into the wc command, which counts the number of lines in the who output.  
  
**ps -e wc -l**  
This command counts the number of processes running on your computer system.  
  
**12) compress**: The "compress" command is used to compress files. It's function is similar to PkZIP in the DOS/Windows world.  
  
**compress /tmp/myfiles.tar**  
This command compresses the file /tmp/myfiles.tar, and replaces it with a file named /tmp/myfiles.tar.Z.  
  
**compress -v /tmp/myfiles.tar**This command works just like the previous example, but gives more verbose output during the compression process. This is useful if you like to see the compression ratio you're getting with this utility.  
  
**13) cancel**  
: The "cancel" command lets you stop print requests from printing.  
  
Requests can be cancelled by (1) using the printer-id or (2) by specifying the printer name.  
  
**cancel laser-101**  
This command terminates (or cancels) the print request identified as "laser-101".  
  
**cancel -u student1**  
This command cancels all of the print jobs that are queued for the user named "student1".  
  
**14) lp**  
: "lp" is the command used to print files. "lp" stands for line printer.  
  
**lp /etc/passwd**  
This command prints the "/etc/passwd" file to the default printer. If you do not use the "-d" option, the file is printed to the default printer destination.  
  
**lp -dSales .profile**This command prints the ".profile" file to the printer named "Sales". The -d option specifies the destination.  
  
**lp -dSales file1 file2 file3**This command prints the three files "file1", "file2", and "file3" to the printer named "Sales".  
  
**lp -i Sales-101 -H hold**  
This command places the print request Sales-101 on hold.  
  
**lp -i Sales-101 -H resume**This command resumes the print request Sales-101. The print request starts printing from page one unless you instruct it otherwise.  
  
**lp -i Sales-101 -H resume -P 4-**  
This command resumes the print request Sales-101, starting with page 4 of the print job. Note that the final hyphen (following the "4") is required.  
The "lp" command can also be used as part of a pipeline. For instance, the following command will print the output of the "ps -ef" command to the default printer:  
  
**ps -ef lp**  
Print the output of the "ps -ef" command to the default printer.  
  
**The lpr command is used on**[**BSD**](https://kb.iu.edu/data/agom.html)**systems, and the lp command is used in**[**System V**](https://kb.iu.edu/data/agjs.html)**. Both commands may be used on the**[**UITS**](https://kb.iu.edu/data/ahaw.html)**systems.**  
  
  
**15) find**  
: The "find" command is very powerful. It can search the entire file system for one or more files that you specify to look for. This is very helpful when a file has been "lost".  
You can also use the find command to locate files, and then perform some type of action on the files after they've been located. With this capability, you can locate files using powerful search criteria, and then run any UNIX command you want on the files you locate.  
  
**find / -name Chapter1 -type f -print**  
This command searches through the root file system ("/") for the file named "Chapter1". If it finds the file, it prints the location to the screen.  
  
**find /usr -name Chapter1 -type f -print**This command searches through the "/usr" directory for the file named "Chapter1".  
  
**find /usr -name "Chapter\*" -type f -print**  
This command searches through the "/usr" directory for all files that begin with the letters "Chapter". The filename can end with any other combination of characters.  
  
**find /usr/local -name "\*.html" -type f -print**  
This command searches through the "/usr/local" directory for files that end with the extension ".html". These file locations are then printed to the screen.  
  
**find /usr/local -name "\*.html" -type f  
-exec chmod 644 {} \;**  
This command searches through the "/usr/local" directory for files that end with the extension ".html". When these files are found, their permission is changed to mode 644 (rw-r--r--).  
  
**find htdocs cgi-bin -name "\*.cgi" -type f  
-exec chmod 755 {} \;**  
This command searches through the "htdocs" and "cgi-bin" directories for files that end with the extension ".cgi". When these files are found, their permission is changed to mode 755 (rwxr-xr-x). This example shows that the find command can easily search through multiple sub-directories (htdocs, cgi-bin) at one time.  
  
Some Examples:  
find $HOME -print will lists all files in your home directory.  
find /work -name chapter1 -print will list all files named chapter1 in /work directory.  
find / -type d -name 'man\*' -print will list all manpage directories.  
find / -size 0 -ok rm {} \; will remove all empty files on system.  
  
  
To find all of the files named myfile.txt in your current directory and all of its subdirectories, enter:  
**find . -name myfile.txt -print**  
  
To look in your current directory and its subdirectories for all of the files that end in the extension .txt , enter:  
**find . -name "\*.txt" -print**  
  
In these examples, the . (period) represents your current directory.  
  
  
**conditions of find**  
-**atime +n -n n** will find files that were last accessed more than n or less than -n days or n days.  
**-ctime +n or -n** will find that were changed +n -n or n days ago.  
**-depth** descend the directory structure, working on actual files first and then directories. You can use it with cpio command.  
**-exec commad {} \;** run the Unix command on each file matched by find. Very useful condition.  
**-print** print or list to standard output (screen).  
**-name** pattern find the pattern.  
**-perm** nnnfind files whole permission flags match octal number nnn.  
**-size n** find files that contain n blocks.  
**-type c** Find file whole type is c. C could be b or block, c Character special file, d directory, p fifo or named pipe, l symbolic link, or f plain file.  
  
  
**16) grep**  
: "general regular expression parser";  
It is used to search for text strings within one or more files.  
  
**grep 'fred' /etc/passwd**  
This command searches for all occurrences of the text string 'fred' within the "/etc/passwd" file. It will find and print (on the screen) all of the lines in this file that contain the text string 'fred', including lines that contain usernames like "fred" - and also "alfred".  
  
**grep '^fred' /etc/passwd**  
This command searches for all occurrences of the text string 'fred' within the "/etc/passwd" file, but also requires that the "f" in the name "fred" be in the first column of each record (that's what the caret character tells grep). Using this more-advanced search, a user named "alfred" would not be matched, because the letter "a" will be in the first column.  
  
**grep 'joe' \***  
This command searches for all occurrences of the text string 'joe' within all files of the current directory.  
  
It can also be used in conjunction with other commands as in this following example, output of ps command is passed to grep command, here it means search all processes in system and find the pattern sleep.  
**ps -ef grep sleep** will display all the sleep processes running in the system.  
  
**Options:**  
**-b** option will precede each line with its block number.  
**-c** option will only print the count of matched lines.  
**-i** ignores uppercase and lowercase distinctions.  
**-l** lists filenames but not matched lines.  
  
  
  
  
**17) df**  
: "df" stands for disk free. This command is used to show the amount of free space on one or more file systems.  
  
df  
A basic display of the free disk space (showing blocks and i-nodes) on all mounted file systems.  
  
**df /**  
Displays the free disk space in the / file system.  
  
**df -kvi**  
A more verbose display showing the free disk space and free i-node space of all file systems, measured in kilobytes.  
  
**18) du**  
: "du" stands for disk usage. This command is used to show the amount of disk space consumed by one or more directories (or directory trees).  
  
**du**  
Shows the disk usage of the current working directory. Each file in the directory and it's usage (in blocks) is displayed.  
  
**du -k**  
Shows the usage of the current directory in 1024-byte units (kilobytes).  
  
**du /home/fred**  
Shows the disk usage of the /home/fred subdirectory.  
  
**du -ks /home/fred**Shows only a summary of the disk usage of the /home/fred subdirectory (measured in kilobytes).  
  
**du -ks /home/fred/\***  
Shows a summary of the disk usage of each subdirectory of /home/fred (measured in kilobytes).  
  
**19) ps**  
: The "ps" command (process statistics) lets you check the status of processes that are running on your Unix system.  
  
**ps**  
The ps command by itself shows minimal information about the processes you are running. Without any arguments, this command will not show information about other processes running on the system.  
  
**ps -f**  
The -f argument tells ps to supply full information about the processes it displays. In this example, ps display full information about the processes you are running.  
  
**ps -e**  
The -e argument tells the ps command to show every process running on the system.  
  
  
  
**ps -ef**  
The -e and -f arguments are normally combined like this to show full information about every process running on the system. This is probably the most often-used form of the ps command.  
  
**ps -ef more**  
Because the output normally scrolls off the screen, the output of the ps -ef command is often piped into the more command. The more command lets you view one screenful of information at a time.  
  
**ps -fu fred**  
This command shows full information about the processes currently being run by the user named fred (the -u option lets you specify a username).  
  
**20) who**: The "who" command lets you display the users that are currently logged into your Unix computer system.  
  
who  
This is the basic who command with no command-line arguments. It shows the names of users that are currently logged in, and may also show the terminal they're logged in on, and the time they logged in.  
  
**who more**In this example the output of the who command is piped into the more command. This is useful when there are a lot of users logged into your computer system, and part of the output of the who command scrolls off the screen.  
  
The w and who commands are similar programs that list all users logged into the computer. If you use w, you also get a list of what they are doing. If you use who, you also get the [**IP numbers**](https://kb.iu.edu/data/aakl.html) or computer names of the terminals they are using.  
  
**21) banner**  
: banner prints characters in a sort of ascii art poster  
  
banner prasad  
  
**22) cal**  
: cal command will print the calander on current month by default. If you want to print calander of august of 1965. That's eightht month of 1965.  
  
**cal 8 1965**  
  
**23) clear**: clear command clears the screen and puts cursor at beginning of first line.  
  
**24) calendar**: calendar command reads your calendar file and displays only lines with current day.  
For example in your calendar file if you have this  
  
12/20 Test new software.  
1/15 Test newly developed 3270 product.  
1/20 Install memory on HP 9000 machine.  
On dec 20th the first line will be displayed.  
  
  
**25) nohup**  
  
: nohup command if added in front of any command will continue running the command or process even if you shut down your terminal or close your session to machine. For example, if I want to run a job that takes lot of time and must be run from terminal and is called update\_entries\_tonight .  
**nohup update\_entries\_tonight** will run the job even if terminal is shut down in middle of this job.  
  
**26) tty**  
: Tty command will display your terminal. Syntax is  
tty options  
  
**Options**  
**-l** will print the synchronous line number.  
**-s** will return only the codes: 0 (a terminal), 1 (not a terminal).  
  
**27) head**: head filename by default will display the first 10 lines of a file.  
If you want first 50 lines you can use head -50 filename or for 37 lines head -37 filename and so forth.  
  
**28) tail**  
: tail filename by default will display the last 10 lines of a file.  
If you want last 50 lines then you can use tail -50 filename.  
  
**29) rcp**  
: rcp command will copy files between two unix systems and works just like cp command (-p and -i options too).  
For example you are on a unix system that is called Cheetah and want to copy a file which is in current directory to a system that is called lion in /usr/john/ directory then you can use rcp command  
**rcp filename lion:/usr/john**  
You will also need permissions between the two machines.  
  
**30) ln**  
: Instead of copying you can also make links to existing files using ln command.  
If you want to create a link to a file called coolfile in /usr/local/bin directory then you can enter this command.  
**ln mycoolfile /usr/local/bin/coolfile**  
**ln -s** fileone filetwo will create a symbolic link and can exist across machines.  
**ln -n** option will not overwrite existing files.  
**ln -f** will force the link to occur.  
  
**A symbolic link, also termed a soft link, is a special kind of file that points to another file, much like a shortcut in Windows**  
Unlike a hard link, a symbolic link does not contain the data in the target file. It simply points to another entry somewhere in the file system. This difference gives symbolic links certain qualities that hard links do not have, such as the ability to link to directories, or to files on remote computers networked through NFS. Also, when you delete a target file, symbolic links to that file become unusable, whereas hard links preserve the contents of the file.  
  
  
**To create a symbolic link in Unix, at the Unix prompt, enter:**  
  
ln -s source\_file myfile  
  
After you've made the symbolic link, you can perform an operation on or execute myfile, just as you could with the source\_file. You can use normal file management commands (e.g., cp, rm) on the symbolic link.  
  
If you delete the source file or move it to a different location, your symbolic file will not function properly.  
  
  
**31) rmdir**  
: rmdir command will remove directory or directories if a directory is empty.  
  
**32) diff**  
: diff command will compare the two files and print out the differences between.  
  
**diff fileone filetwo**  
Contents of fileone are  
  
This is first file  
this is second line  
this is third line  
this is different as;lkdjf  
this is not different  
  
filetwo contains  
This is first file  
this is second line  
this is third line  
this is different xxxxxxxas;lkdjf  
this is not different  
  
**diff fileone filetwo** will give following output  
4c4  
<> this is different xxxxxxxas;lkdjf  
  
  
  
  
**33) cmp**  
: cmp command compares the two files.  
  
**cmp fileone filetwo** will give me  
  
fileone filetwo differ: char 80, line 4  
  
if I run cmp command on similar files nothing is returned.  
**-s** command can be used to return exit codes. i.e. return 0 if files are identical, 1 if files are different, 2 if files are inaccessible.  
This following command prints a message 'no changes' if files are same  
**cmp -s fileone file1 && echo 'no changes'**  
no changes  
  
**34) dircmp**  
: dircmp command compares two directories. If i have two directories in my home directory named  
dirone and dirtwo and each has 5-10 files in it. Then  
dircmp dirone dirtwo will return this  
  
Dec 9 16:06 1997 dirone only and dirtwo only Page 1  
  
./cal.txt ./fourth.txt  
./dohazaar.txt ./rmt.txt  
./four.txt ./te.txt  
  
**35) cut**  
: cut command selects a list of columns or fields from one or more files.  
**Option -c** is for columns and**-f** for fields. It is entered as  
**cut options [files]**  
for example if a file named testfile contains  
  
this is firstline  
this is secondline  
this is thirdline  
Examples:  
**cut -c1,4 testfile** will print this to standard output (screen)  
ts  
ts  
ts  
It is printing columns 1 and 4 of this file which contains t and s (part of this).  
  
  
  
Options:  
-c list cut the column positions identified in list.  
-f list will cut the fields identified in list.  
-s could be used with -f to suppress lines without delimiters.  
  
**36) paste**: paste command merge the lines of one or more files into vertical columns separated by a tab.  
for example if a file named testfile contains  
this is firstline  
  
and a file named testfile2 contains  
this is testfile2  
  
then running this command  
**paste testfile testfile2 > outputfile**  
will put this into outputfile  
this is firstline this is testfile2  
  
it contains contents of both files in columns.  
**who paste** - - will list users in two columns.  
  
**Options:**  
**-d'char'** separate columns with char instead of a tab.  
**-s** merge subsequent lines from one file.  
  
**37) sort**  
: sort command sort the lines of a file or files, in alphabetical order.  
  
**Options:**  
**-b** ignores leading spaces and tabs.  
**-c** checks whether files are already sorted.  
**-d** ignores punctuation.  
**-i** ignores non-printing characters.  
**-n** sorts in arithmetic order.  
**-ofile** put output in a file.  
**+m[-m]** skips n fields before sorting, and sort up to field position m.  
**-r** reverse the order of sort.  
**-u** identical lines in input file appear only one time in output.  
  
**38) uniq**: uniq command removes duplicate adjacent lines from sorted file  
  
**sort names uniq -d** will show which lines appear more than once in names file.  
  
**Options:  
-c** print each line once, counting instances of each.  
**-d** print duplicate lines once, but no unique lines.  
**-u** print only unique lines.  
  
**39) chmod**: This command changes the permission information associated with a file. Every file (including directories, which Unix treats as files) on a Unix system is stored with records indicating who has permission to read, write, or execute the file, abbreviated as r, w, and x. These permissions are broken down for three categories of user: first, the owner of the file; second, a group with which both the user and the file may be associated; and third, all other users. These categories are abbreviated as u for owner (or user), g for group, and o for other.  
  
To allow yourself to execute a file that you own named myfile, enter:  
**chmod u+x myfile**  
To allow anyone who has access to the directory in which myfile is stored to read or execute myfile, enter:  
**chmod o+rx myfile**  
  
**40) less and more**: Both less and more display the contents of a file one screen at a time, waiting for you to press the Spacebar between screens. This lets you read text without it scrolling quickly off your screen. The less utility is generally more flexible and powerful than more, but more is available on all Unix systems while less may not be.  
  
To read the contents of a file named textfile in the current directory, enter:  
**less textfile**  
  
The less utility is often used for reading the output of other commands. For example, to read the output of the ls command one screen at a time, enter:  
**ls -la less**  
  
In both examples, you could substitute more for less with similar results. To exit either less or more, press q .  
  
**41) set**  
: This command displays or changes various settings and options associated with your Unix session.  
  
To see the status of all settings, enter the command without options:  
Set