**Ubuntu Commands**

**1.gedit:** If you want to edit a file graphically—even a system

file—the gedit text editor makes it painless and easy.

**Syntax:**

**$ gedit filename**

**2.cal:** It displays current month calendar.

**Syntax:**

**$ cal**

**3.touch command:** It is used to create a file without any content.

The file created using touch command is empty. This command

can be used when the user doesn’t have data to store at the time of

file creation.

**Syntax:**

**$ touch filename**

**4.mv\*:** Moves all files from one location to another.

**Syntax:**

**$ mv \* destinationfile**

5.**passwd:** The **passwd** command changes passwords for user

accounts. A normal user may only change the password for his/her

own account, while the superuser may change the password for

any account.

**Syntax:**

**$ passwd**

**6.Cp \*:** Copies all files from one location to another. It usually

overwrites the content of destination file

**Syntax:**

**$ cp \* destinationfile**

**7.mkdir** : Creates a directory or subdirectory. Command extensions,

which are enabled by default, allow you to use a single **mkdir**

command to create intermediate directories in a specified path.

**Syntax:**

**$ mkdir filename**

**8.Cp file1 file2:** Copies one or more files from one location to

another. It usually overwrites the content of destination file

**Syntax:**

**$ cp sourcefile destinationfile**

**9.rmdir:** This command is used to delete the empty folder only. The

**rmdir** command can also run from the Windows Recovery

Console, using different parameters.

**Syntax:**

**$ rmdir filename**

**10.rm:** This command is used to delete the file in the specific

folder.

**Syntax:**

**$ rm filename**

**11.Cp –r folder1 folder2:** Copies one or more files from one

folder to another. The destination folder can be anywhere.

**Syntax:**

**$ cp –r sourceFolder destinationFolder**

**12.mv file1 file2:** Moves one or more files from one location to

another. It usually overwrites the content of destination file

**Syntax:$ mv sourcefile destinationfile**

**13.Sort:** SORT command is used to sort a file, arranging the

records in a particular order. By default, the sort command sorts

file assuming the contents are ASCII.

**Syntax:**

**$ sort filename**

**14.vim:** This command is used for editor in command line.

Using this, you can write or read any file or also append in existing

file.

**Syntax:**

**$ vim filename**

**14.chmod: chmod** command is used to change the access mode

of a file. The name is an abbreviation of **change mode**

**15.Tree:** tree command is a recursive directory listing program that

produces a depth-indented listing of files.

**Syntax:**

**$ tree**

**16.tail:** The head command, as the name implies, print the last N

amount of data of the given input. By default, it prints the last 10

lines of the specified files.

**Syntax:**

**$ tail filename**

**17.tail-v:** By using this option, data from the specified file is

always preceded by its file name.

**Syntax:**

**$ tail-v filename**

**18.tail-n:** The head command, print the last N amount of data of

the given input. By default, it prints the last 10 lines of the specified

files. Prints the last ‘num’ lines instead of last 10 lines. **num** is

mandatory to be specified in command otherwise it displays an

error.

**Syntax:**

**$ tail-n filename**

**19.tail-c:** Prints the last ‘num’ bytes from the file specified.

Newline count as a single character, so if head prints out a newline,

it will count it as a byte. **num** is mandatory to be specified in

command otherwise displays an error.

**Syntax:**

**$ tail-c filename**

**20.head:** The head command, as the name implies, print the top

N amount of data of the given input. By default, it prints the first 10

lines of the specified files.

**Syntax:**

**$ head filename**

**21.Head-n:** The head command, print the top N amount of data

of the given input. By default, it prints the first 10 lines of the

specified files. Prints the first ‘num’ lines instead of first 10 lines. **num** is mandatory to be specified in command otherwise it displays

an error.

**Syntax:**

**$ head-n filename**

**22.Head-c:** Prints the first ‘num’ bytes from the file specified.

Newline count as a single character, so if head prints out a newline,

it will count it as a byte. **num** is mandatory to be specified in

command otherwise displays an error.

**Syntax:**

**$ head-c filename**

**23.Head-v:** By using this option, data from the specified file is

always preceded by its file name.

**Syntax:**

**$ head-v filename**

**24.Head-c:** When you want to print n top lines from two different

files, you can use this command.

**Syntax:**

**$ head-c filename**

**25.more: more** command is used to view the text files in the

command prompt, displaying one screen at a time in case the file is

large (For example log files). The more command also allows the

user do scroll up and down through the page.

**Syntax:**

**$ more filename**

**26.More-p:** This option clears the screen and then displays the

text.

**Syntax:**

**$ more-p filename**

**27.More-f:** This option does not wrap the long lines and displays

them as such.

**Syntax:**

**$ more-f filename**

**28.More-c:** This command is used to display the pages on the

same area by overlapping the previously displayed text. We cannot

scroll down in this.

**Syntax:**

**$ more-c filename**

**29.Tr:** It is used to translate, like new lines into spaces, spaces

to new lines or tabs. This is also used with ‘echo’ command.

**30.Tr:** The command 'tr' stands for **'translate'**. It is used to

translate, like from lowercase to uppercase and vice versa. We can

use ‘[:lower:]’ ‘[:upper:]’ also for this.

**Syntax:**

**$ command | tr <'old'> <'new'>**

**31.a-w: It grants write command to all user (u,g,o)**

**32.u=r: It revokes previous permission and assign only read**

**permission to owner/user.**

**33.u+w: grant write permission to owner/user.**

**34.000: No permission to any user**

**35.777: All permission to all user**