BAITUSSALAM TECH PARK

LINUX

BAITUSSALAM
—TECH PARK—

What is Linux…?

- Linux is a free and open-source operating system kernel initially developed by Linus Torvalds in 1991. It is a Unix-like operating system kernel that serves as the foundation for various Linux distributions, commonly referred to as "distros.”

- Linux is known for its stability, security, and flexibility. It is widely used in servers, desktop computers, mobile devices, embedded systems, and supercomputers. Additionally, it powers a significant portion of the internet infrastructure, with many web servers running on Linux-based operating systems.

- One of the key features of Linux is its open-source nature, which means its source code is freely available for anyone to view, modify, and distribute under the terms of the GNU General Public License (GPL) and other open-source licenses. This openness has led to a vibrant community of developers and enthusiasts contributing to its development and creating numerous distributions tailored to different use cases and preferences.

Why Linux…?

- Open Source
- Customizability
- Stability and Reliability
- Security
- Cost-Effectiveness
- Performance
- Community Support
- Compatibility

Linux Basic Commands…

ls

The ls command lists files and directories in your system. Here’s the syntax:

```
ls [/directory/folder/path]
```

If you remove the path, the ls command will show the current working directory’s content. You can modify the command using these options:

- -R – lists all the files in the subdirectories.
- -a – shows all files, including hidden ones.
- -lh – converts sizes to readable formats, such as MB, GB, and TB.

Linux Basic Commands…

pwd

The pwd command prints your current working directory path, like /home/directory/path. Here’s the command syntax:

```
ls [/directory/folder/path]
```

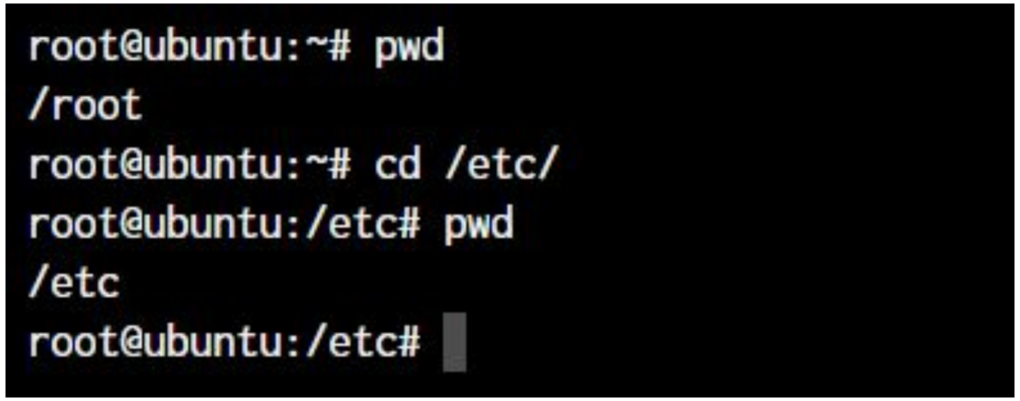
If you remove the path, the ls command will show the current working directory content. You can modify the command using these options:

Linux Basic Commands…

cd

While working within the terminal, moving around within directories is pretty much a necessity. The cd command is one of the important Linux commands you must know and it will help you to navigate through directories. Just type cd followed by directory as shown below.

```
root@ubuntu:~# cd <directory path>
```



```
root@ubuntu:~# pwd
/root
root@ubuntu:~# cd /etc/
root@ubuntu:/etc# pwd
/etc
root@ubuntu:/etc#
```

As you can see in the above command, I simply typed cd /etc/ to get into the /etc directory. We used the pwd command to print the current working directory.

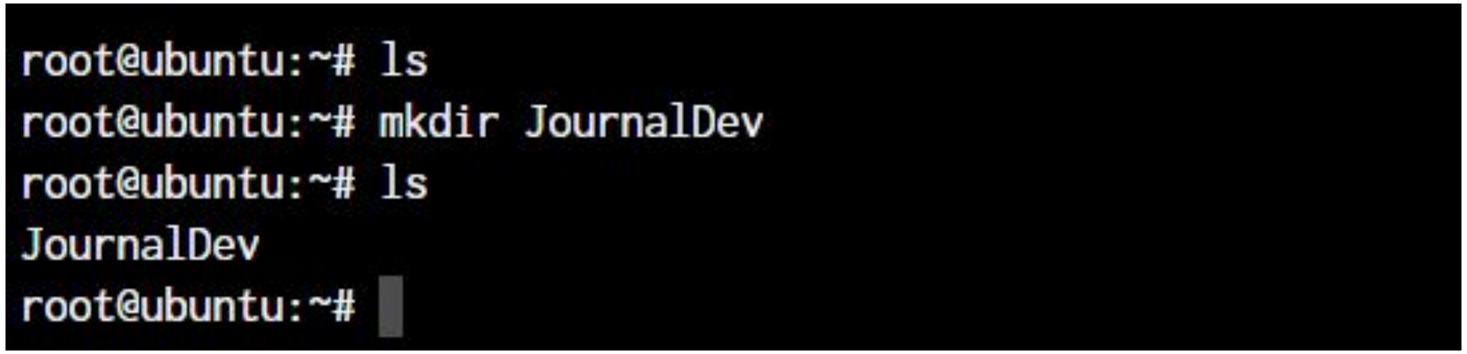
Linux Basic Commands…

mkdir

The mkdir command allows you to create directories from within the terminal.

The default syntax is mkdir followed by the directory name.

root@ubuntu:~# mkdir <folder name>



```
root@ubuntu:~# ls
root@ubuntu:~# mkdir JournalDev
root@ubuntu:~# ls
JournalDev
root@ubuntu:~#
```

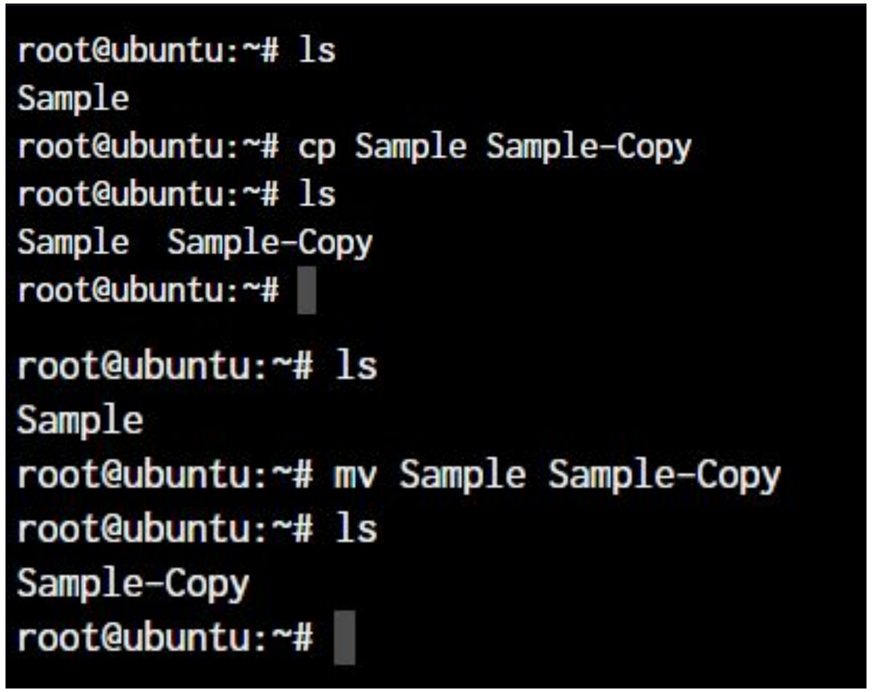
As you can see in the above screenshot, we created the JournalDev directory with just this simple command.

Linux Basic Commands…

cp and mv

The cp and mv commands are equivalent to the copy-paste and cut-paste in Windows. But since Linux doesn’t really have a command for renaming files, we also make use of the mv command to rename files and folders.

```
root@ubuntu:~# cp <source> <destination>
```



```
root@ubuntu:~# ls
Sample
root@ubuntu:~# cp Sample Sample-Copy
root@ubuntu:~# ls
Sample  Sample-Copy
root@ubuntu:~#
```

In the above command, we created a copy of the file named Sample. Let’s see how what happens if we use the mv command in the same manner. For this demonstration, I’ll delete the Sample-Copy file.

```
root@ubuntu:~# ls
Sample
root@ubuntu:~# mv Sample Sample-Copy
root@ubuntu:~# ls
Sample-Copy
root@ubuntu:~#
```

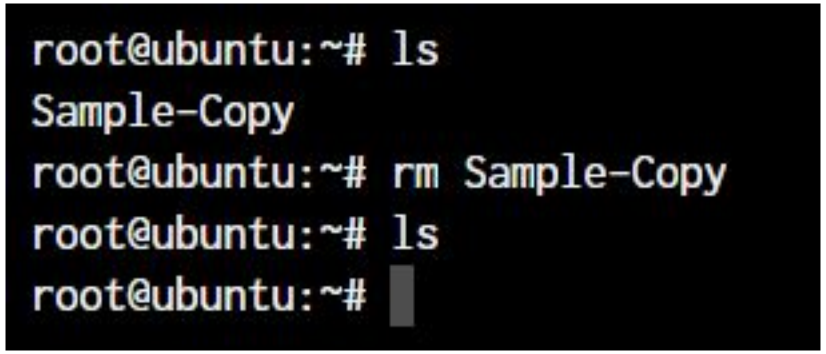
In the above case, since we were moving the file within the same directory, it acted as rename. The file name is now changed.

Linux Basic Commands…

rm

In the previous section, we deleted the Sample-Copy file. The rm command is used to delete files and folders and is one of the important Linux commands you must know.

```
root@ubuntu:~# rm <file name>
root@ubuntu:~# rm -r <folder/directory name>
```



```
root@ubuntu:~# ls
Sample-Copy
root@ubuntu:~# rm Sample-Copy
root@ubuntu:~# ls
root@ubuntu:~#
```

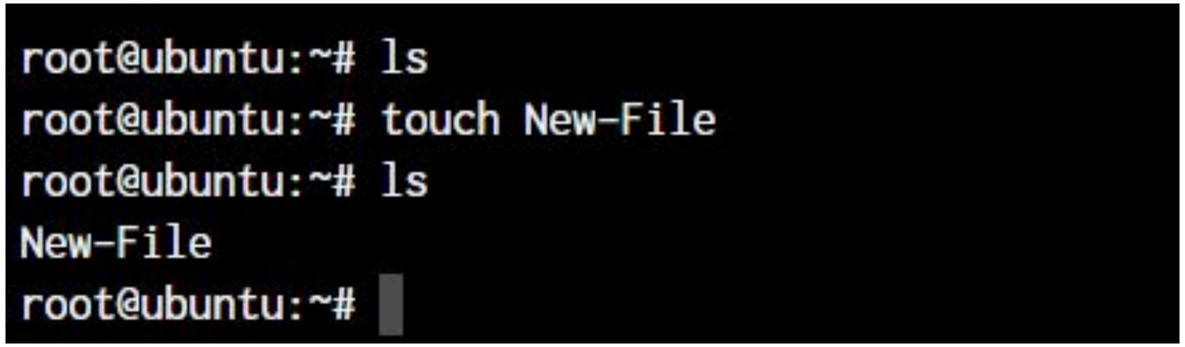
To delete a directory, you have to add the **-r** argument to it. Without the -r argument, rm command won’t delete directories.

Linux Basic Commands…

touch

To create a new file, the touch command will be used. The **touch** keyword followed by the file name will create a file in the current directory.

```
root@ubuntu:~# touch <file name>
```



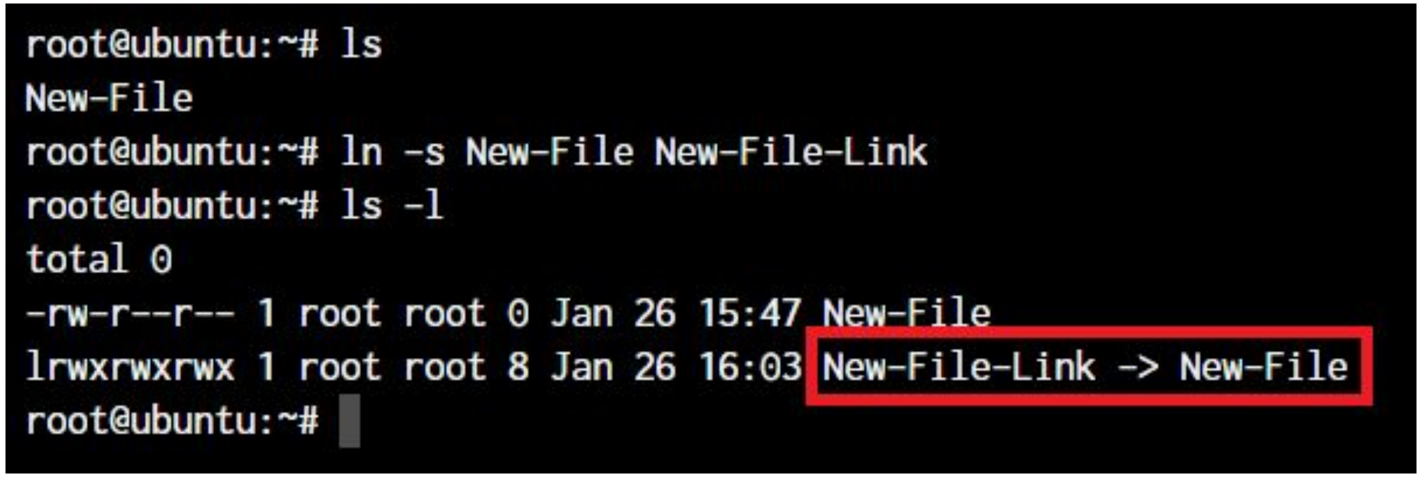
```
root@ubuntu:~# ls
root@ubuntu:~# touch New-File
root@ubuntu:~# ls
New-File
root@ubuntu:~#
```

Linux Basic Commands…

ln

To create a link to another file, we use the ln command. This is one of the important Linux commands that you should know if you're planning to work as a Linux administrator.

root@ubuntu:~# ln -s <source path> <link name>



```
root@ubuntu:~# ls
New-File
root@ubuntu:~# ln -s New-File New-File-Link
root@ubuntu:~# ls -l
total 0
-rw-r--r-- 1 root root 0 Jan 26 15:47 New-File
lrwxrwxrwx 1 root root 8 Jan 26 16:03 New-File-Link -> New-File
root@ubuntu:~#
```

The basic syntax involves using the **-s** parameter so we can create a symbolic link or soft link.

Linux Basic Commands…

Symbolic/Softlink vs Hardlink

Key differences:

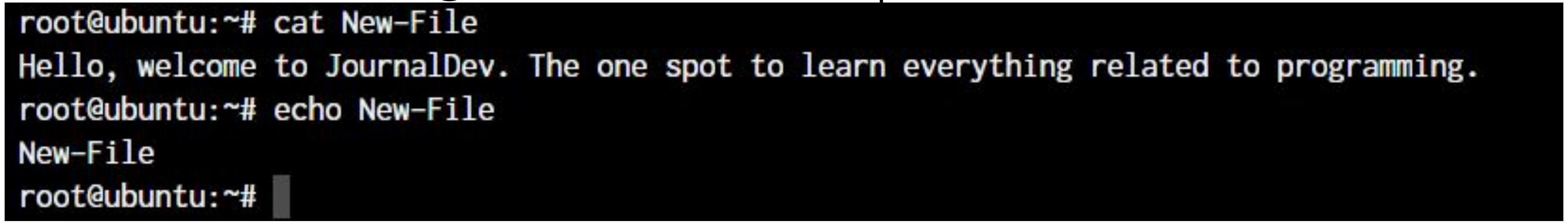
- If you delete the original file, a symbolic link becomes broken, whereas a hard link remains functional as it points to the same inode.
- Symbolic links can span file systems, while hard links cannot.
- Symbolic links have their own inode and data block, while hard links share the same inode and data block as the original file.
- Symbolic links can reference directories, while hard links can only reference files.
- In summary, symbolic links are more flexible and can point to directories or across file systems, but they are more fragile. Hard links are more robust, as they reference the same data blocks as the original file, but they have more limitations in terms of file system boundaries and cannot point to directories.

Linux Basic Commands…

cat, echo and less

When you want to output the contents of a file, or print anything to the terminal output, we make use of the cat or echo commands. Let's see their basic usage. I've added some text to our New-File that we created earlier.

```
root@ubuntu:~# cat <file name>

root@ubuntu:~# echo <Text to print on terminal>
```



```
root@ubuntu:~# cat New-File
Hello, welcome to JournalDev. The one spot to learn everything related to programming.
root@ubuntu:~# echo New-File
New-File
root@ubuntu:~#
```

The **less command** is used when the output printed by any command is larger than the screen space and needs scrolling. The less command allows use to break down the output and scroll through it with the use of the enter or space keys.
The simple way to do this is with the use of the pipe operator (**|**).

```
root@ubuntu:~# cat /boot/grub/grub.cfg  | less
```

Linux Basic Commands…

man

The man command is a very useful Linux command you must know. When working with Linux, the packages that we download can have a lot of functionality. Knowing it all is impossible. The man pages offer a really efficient way to know the functionality of pretty much all the packages that you can download using the package managers in your Linux distro.

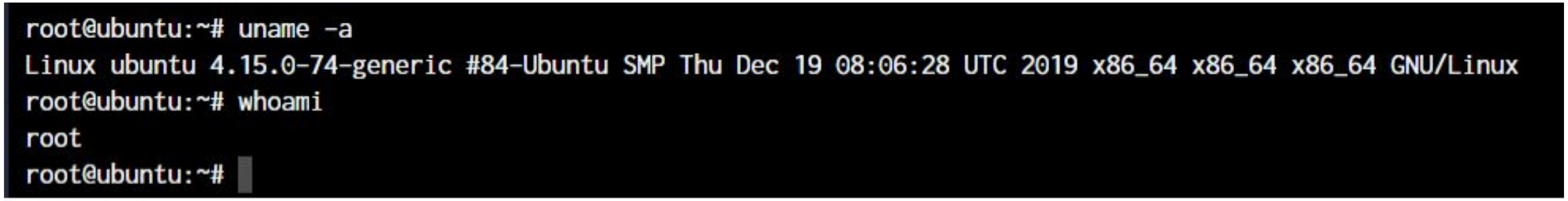
```
root@ubuntu:~# man <command>
```

Linux Basic Commands…

uname and whoami

The uname and whoami commands allow you to know some basic information which comes really handy when you work on multiple systems. In general, if you're working with a single computer, you won't really need it as often as someone who is a network administrator. Let's see the output of both the commands and the way we can use these.

root@ubuntu:~# uname –a



```
root@ubuntu:~# uname -a
Linux ubuntu 4.15.0-74-generic #84-Ubuntu SMP Thu Dec 19 08:06:28 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
root@ubuntu:~# whoami
root
root@ubuntu:~#
```

The parameter **-a** which I've supplied to uname, stands for "all". This prints out the complete information. Kernel version, Machine architecture, Operating system version etc.

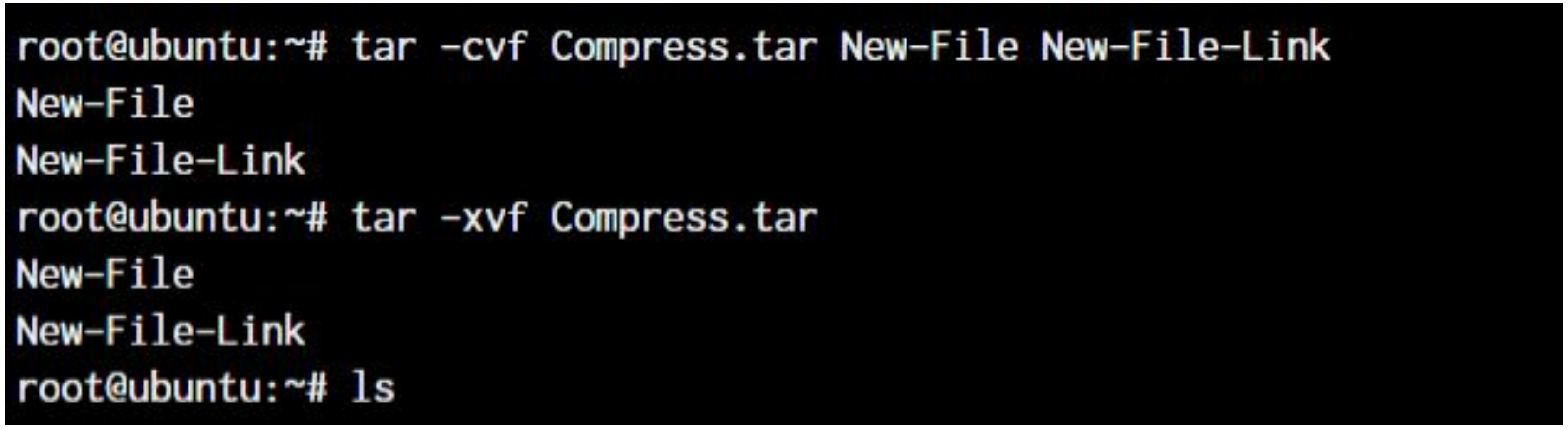
Linux Basic Commands…

tar, zip, and unzip

The tar command in Linux is used to create and extract archived files in Linux. We can extract multiple different archive files using the tar command.
To create an archive, we use the -c parameter and to extract an archive, we use the -x parameter. Let's see it working.

#Compress root@ubuntu:~# tar -cvf <archive name> <files separated by space>

#Extract root@ubuntu:~# tar -xvf <archive name>



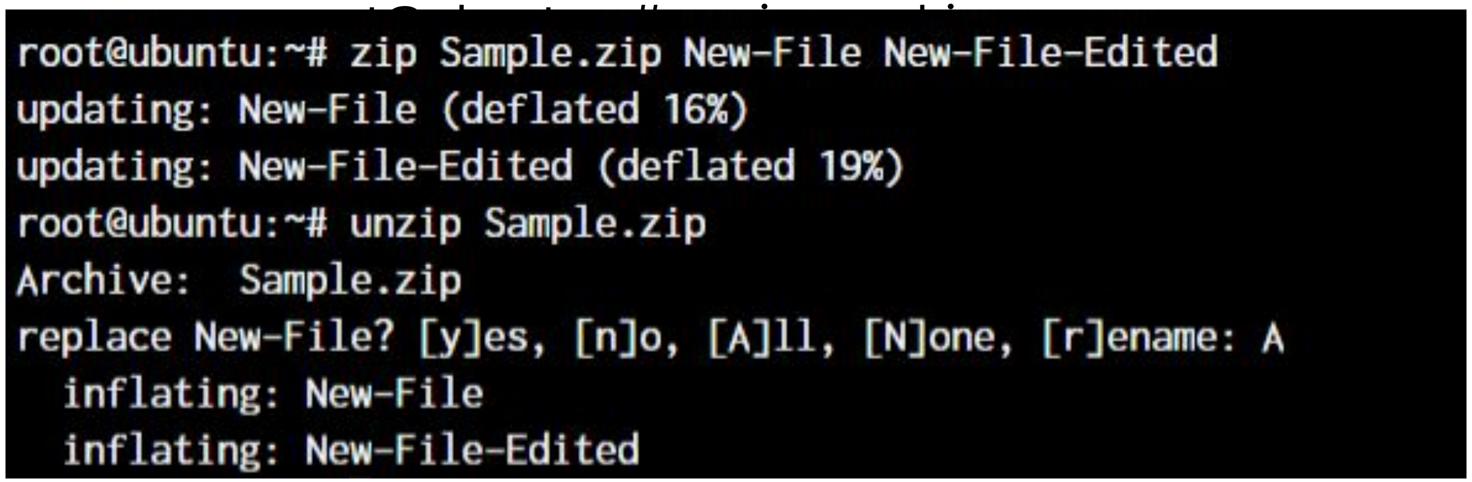
```
root@ubuntu:~# tar -cvf Compress.tar New-File New-File-Link
New-File
New-File-Link
root@ubuntu:~# tar -xvf Compress.tar
New-File
New-File-Link
root@ubuntu:~# ls
```

Linux Basic Commands…

tar, zip, and unzip

In the first line, we created an archive named **Compress.tar** with the New-File and New-File-Link. In the next command, we have extracted those files from the archive.
Now coming to the zip and unzip commands. Both these commands are very straight forward. You can use them without any parameters and they'll work as intended. Let's see an example below.

root@ubuntu:~# zip <archive name> <file names separated by space>



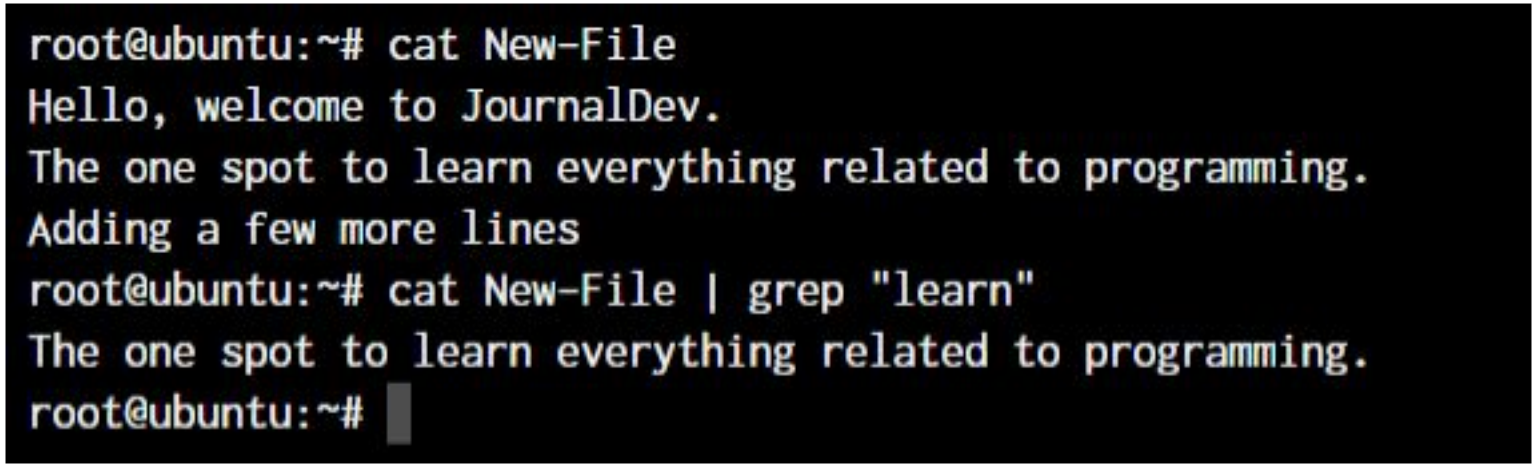
```
root@ubuntu:~# zip Sample.zip New-File New-File-Edited
updating: New-File (deflated 16%)
updating: New-File-Edited (deflated 19%)
root@ubuntu:~# unzip Sample.zip
Archive:  Sample.zip
replace New-File? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: New-File
  inflating: New-File-Edited
```

Linux Basic Commands…

grep

If you wish to search for a specific string within an output, the grep command comes into the picture. We can pipe (|) the output to the grep command and extract the required string.

```
root@ubuntu:~# <Any command with output> | grep "<string to find>"
```

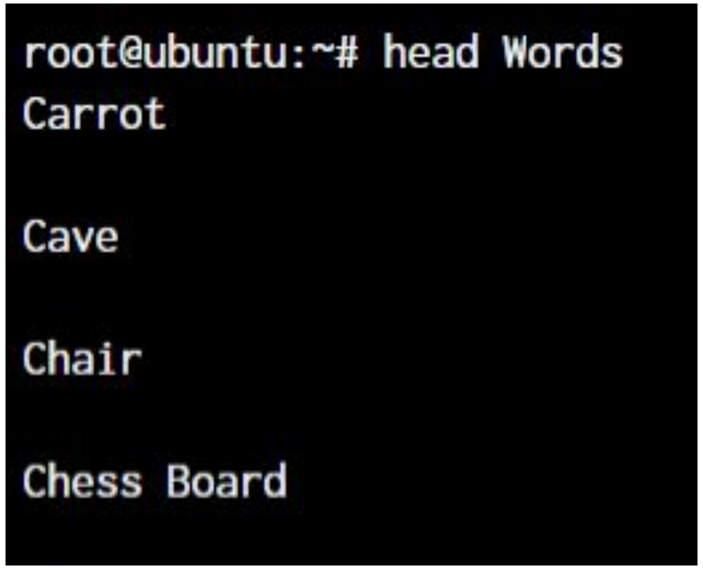


```
root@ubuntu:~# cat New-File
Hello, welcome to JournalDev.
The one spot to learn everything related to programming.
Adding a few more lines
root@ubuntu:~# cat New-File | grep "learn"
The one spot to learn everything related to programming.
root@ubuntu:~#
```

Linux Basic Commands…

head and tail

When outputting large files, the head and the tail commands come in handy. I've created a file named "Words" with a lot of words arranged alphabetically in it. The head command will output the first 10 lines from the file, while the tail command will output the last 10. This also includes any blank lines and not just lines with text.



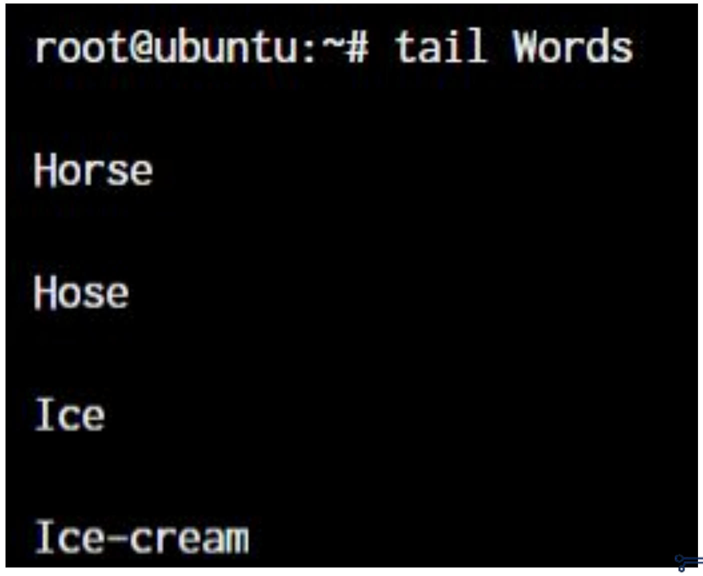
```
root@ubuntu:~# head Words
Carrot

Cave

Chair

Chess Board
```

root@ubuntu:~# head <file name>

root@ubuntu:~# tail <file name>

The head command showed 10 lines from the top of the file.

The tail command outputted the bottom 10 lines from the file.



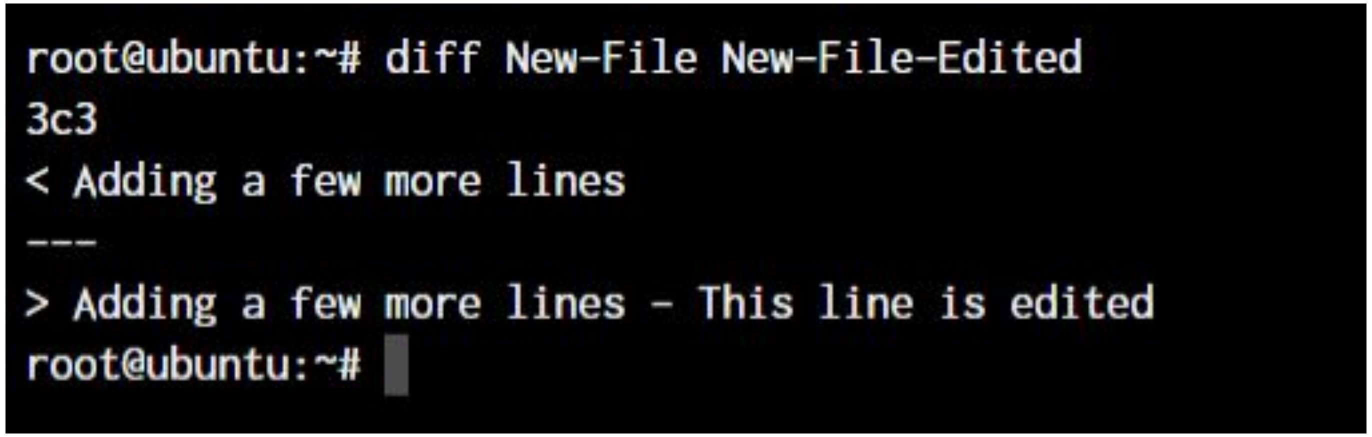
```
root@ubuntu:~# tail Words

Horse

Hose

Ice

Ice-cream
```

Linux Basic Commands…

diff and cmp

Linux offers multiple commands to compare files. The diff, comm, and cmp commands compare differences and are some of the most useful Linux commands you must know. Let's see the default outputs for all the three commands.

```
root@ubuntu:~# diff <file 1> <file 2>
```



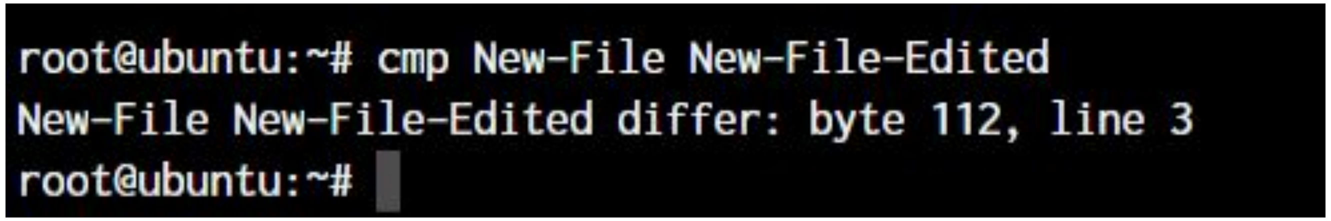
```
root@ubuntu:~# diff New-File New-File-Edited
3c3
< Adding a few more lines
---
> Adding a few more lines - This line is edited
root@ubuntu:~#
```

As you can see above, I've added a small piece of text saying "This line is edited" to the New-File-Edited file.

Linux Basic Commands…

diff and cmp

root@ubuntu:~# cmp <file 1> <file 2>



```
root@ubuntu:~# cmp New-File New-File-Edited
New-File New-File-Edited differ: byte 112, line 3
root@ubuntu:~#
```

The cmp command only tells use the line number which is different. Not the actual text. Let’s see what the comm command does.

Linux Basic Commands…

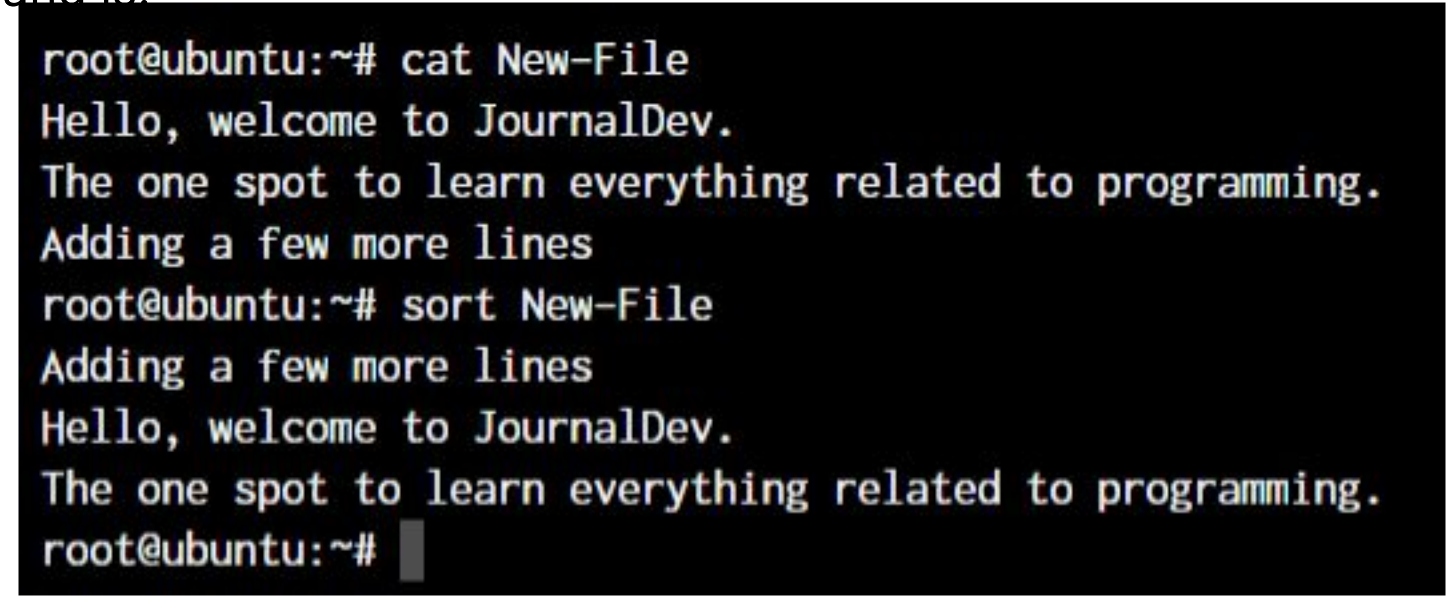
sort

The sort command will provide a sorted output of the contents of a file. Let's use the sort command without any parameters and see the output.

The basic syntax of the sort command is:

root@ubuntu:~# sort <filename>

- -r or --reverse
- -n or --numeric-sort
- -u or --unique



```
root@ubuntu:~# cat New-File
Hello, welcome to JournalDev.
The one spot to learn everything related to programming.
Adding a few more lines
root@ubuntu:~# sort New-File
Adding a few more lines
Hello, welcome to JournalDev.
The one spot to learn everything related to programming.
root@ubuntu:~#
```

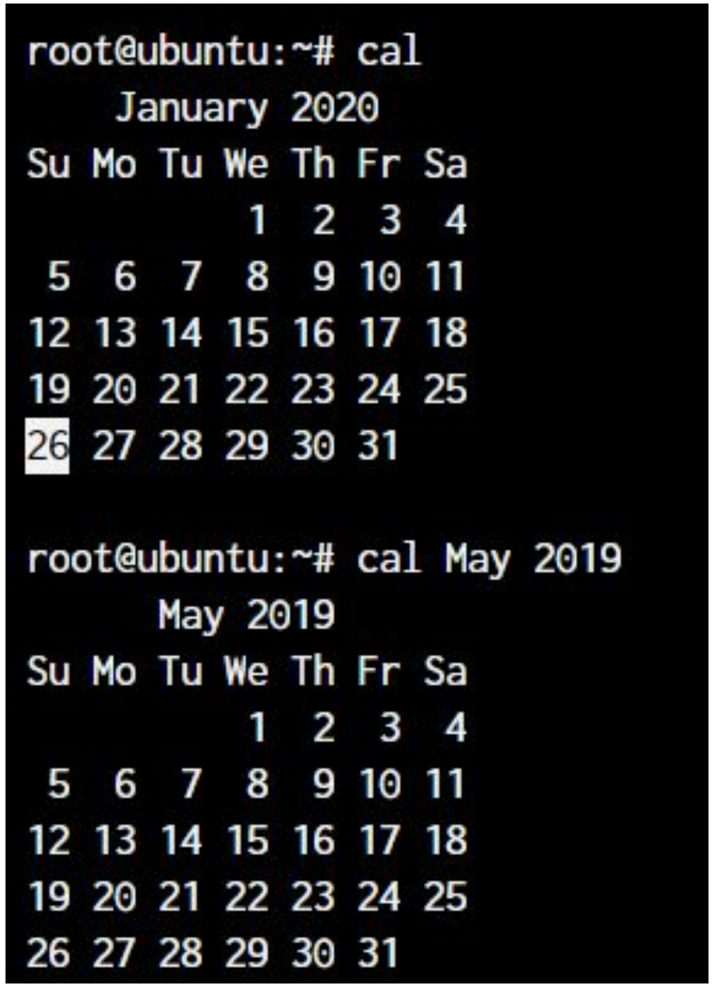
Linux Basic Commands…

cal

Ever wanted to view the calendar in the terminal? Me neither! But there apparently are people who wanted it to happen and well here it is.

The **cal** command displays a well-presented calendar on the terminal. Just enter the word cal on your terminal prompt.

root@ubuntu:~# cal
root@ubuntu:~# cal May 2019



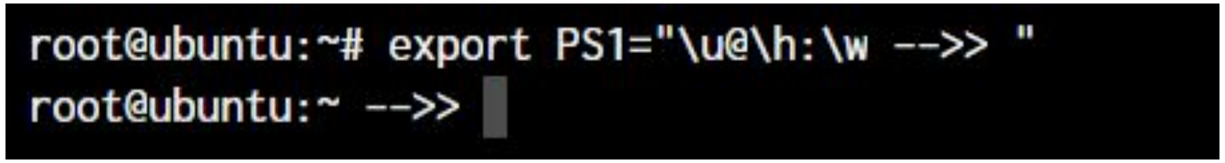
```
root@ubuntu:~# cal
    January 2020
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

root@ubuntu:~# cal May 2019
      May 2019
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

Linux Basic Commands…

Export / unset

The export command is specially used when exporting environment variables in runtime. For example, if I wanted to update the bash prompt, I'll update the PS1 environment variable. The bash prompt will be updated with immediate effect.

```
root@ubuntu:~# export <variable name>=<value>
root@ubuntu:~# unset <variable name>
```



```
root@ubuntu:~# export PS1="\u@\h:\w -->> "
root@ubuntu:~ -->>
```

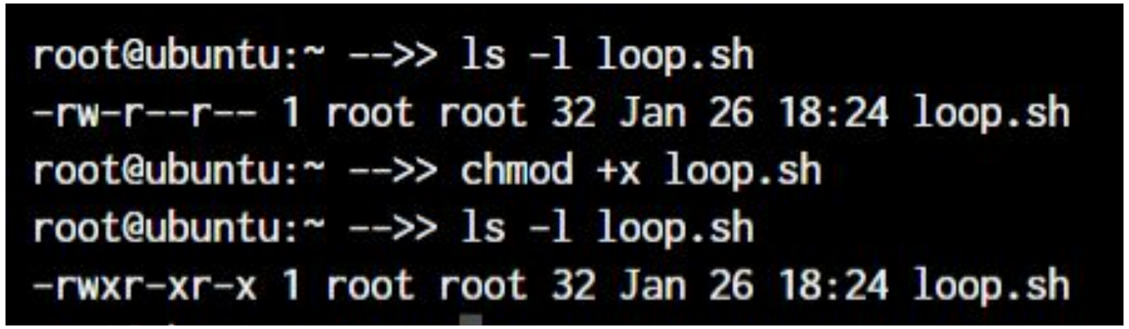
Linux Basic Commands…

chmod and chown

The chmod and chown commands give us the functionality to change the file permissions and file ownership are the most important Linux commands you should know.
The main difference between the functions of the two commands is that the chmod command allows changing file permissions, while chown allows us to change the file owners.
The default syntax for both the commands is **chmod <parameter> filename** and **chown user:group filename**

```
root@ubuntu:~ —>> chmod u+x loop.sh
root@ubuntu:~ —>> chown root:root loop.sh
```

In this example, we're adding executable permissions to the loop.sh file with the **chmod command**. Apart from that, with the **chown command**, we've made it accessible only by root user and users within the root group.



```
root@ubuntu:~ -->> ls -l loop.sh
-rw-r--r-- 1 root root 32 Jan 26 18:24 loop.sh
root@ubuntu:~ -->> chmod +x loop.sh
root@ubuntu:~ -->> ls -l loop.sh
-rwxr-xr-x 1 root root 32 Jan 26 18:24 loop.sh
```

Linux Basic Commands…

ssh

The ssh command allows us to connect to an external machine on the network with the use of the ssh protocol. The basic syntax of the ssh command is:

```
root@ubuntu:~ -->> ssh username@hostname
```

ps, kill, and killall

While we’re on the topic of processes, let’s see how we can find active processes and kill them. To find the running processes, we can simply type **ps** in the terminal prompt and get the list of running processes.

```
root@ubuntu:~ —>> ps
root@ubuntu:~ —>> kill <process ID>
root@ubuntu:~ —>> killall <process name>
```

Linux Basic Commands…

Package Managers

Different distros of Linux make use of different package managers. Since we're working on an Ubuntu server, we have the **apt package manager**. But for someone working on a Fedora, Red Hat, Arch, or Centos machine, the package manager will be different.

- **Debian and Debian-based distros** - apt install <package name>
- **Arch and Arch-based distros** - pacman -S <package name>
- **Red Hat and Red Hat-based distros** - yum install <package name>
- **Fedora and CentOS** - yum install <package>

Getting yourself well versed with the package manager of your distribution will make things much easier for you in the long run. So even if you have a GUI based package management tool installed, try to make use of the CLI based tool before you move on to the GUI utility. Add these to your list of Linux commands you must know.

Linux Basic Commands…

wget

If you want to download a file from within the terminal, the wget command is one of the handiest command-line utilities available. This will be one of the important Linux commands you should know when working with source files.

When you specify the link for download, it has to directly be a link to the file. If the file cannot be accessed by the wget command, it will simply download the webpage in HTML format instead of the actual file that you wanted.

The basic syntax of the wget command is :

```
root@ubuntu:~ →> wget <link to file>
OR
root@ubuntu:~ →> wget -c <link to file> Copy
```

The **-c** argument allows us to resume an interrupted download.

Linux Basic Commands…

ifconfig and traceroute

Moving on to the networking section in Linux, we come across the ifconfig and traceroute commands which will be frequently used if you manage a network.

The ifconfig command will give you the list of all the network interfaces along with the IP addresses, MAC addresses and other information about the interface.

```
root@ubuntu:~ -->> ifconfig
```

When working with traceroute, you can simply specify the IP address, the hostname or the domain name of the endpoint.

```
root@ubuntu:~ -->> traceroute <destination address>
```

Linux Basic Commands…

ufw and iptables

UFW and IPTables are firewall interfaces for the Linux Kernel's netfilter firewall. IPTables directly passes firewall rules to netfilter while UFW configures the rules in IPTables which then sends those rules to netfilter.
Why do we need UFW when we have IPTables? Because IPTables is pretty difficult for a newbie. UFW makes things extremely easy. See the below example where we are trying to allow the port 80 for our web server.

```
root@ubuntu:~# iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
root@ubuntu:~# ufw allow 80
```

I'm sure you now know why UFW was created! Look at how easy the syntax becomes. Both these firewalls are very comprehensive and can allow you to create any kind of configuration required for your network. Learn at least the basics of UFW or IPTables firewall as these are the Linux commands you must know.

Linux Basic Commands…

Sudo (superuser do)

“With great power, comes great responsibility”

This is the quote that’s displayed when a sudo enabled user(sudoer) first makes use of the sudo command to escalate privileges. This command is equivalent to having logged in as root (based on what permissions you have as a sudoer).

```
non-root-user@ubuntu:~# sudo <command you want to run>
Password: Copy
```

Just add the word **sudo** before any command that you need to run with escalated privileges and that’s it. It’s very simple to use but can also be an added security risk if a malicious user gains access to a **sudoer**.

sudo stands for "superuser do."

Linux Basic Commands…

alias/unalias

Do you have some commands that you run very frequently while using the terminal? It could be **rm -r** or **ls -l**, or it could be something longer like **tar -xvzf**. This is one of the productivity-boosting Linux commands you must know.
If you know a command that you run very often, it’s time to create an alias. What’s an alias? In simple terms, it’s another name for a command that you’ve defined.

```
root@ubuntu:~# alias lsl="ls -l"
OR
root@ubuntu:~# alias rmd="rm -r”
```

Now every time you enter **lsl** or **rmd** in the terminal, you’ll receive the output that you’d have received if you had used the full commands.
The examples here are for really small commands that you can still type by hand every time. But in some situations where a command has too many arguments that you need to type, it’s best to create a shorthand version of the same.

Linux Basic Commands…

whereis and whatis

The names of the commands make it very clear as to their functionality. But let's demonstrate their functionality to make things more clear.

The **whereis** command will output the exact location of any command that you type in after the **whereis** command.

```
root@ubuntu:~# whereis sudo sudo: /usr/bin/sudo /usr/lib/sudo /usr/share/man/man8/sudo.8.gz
```

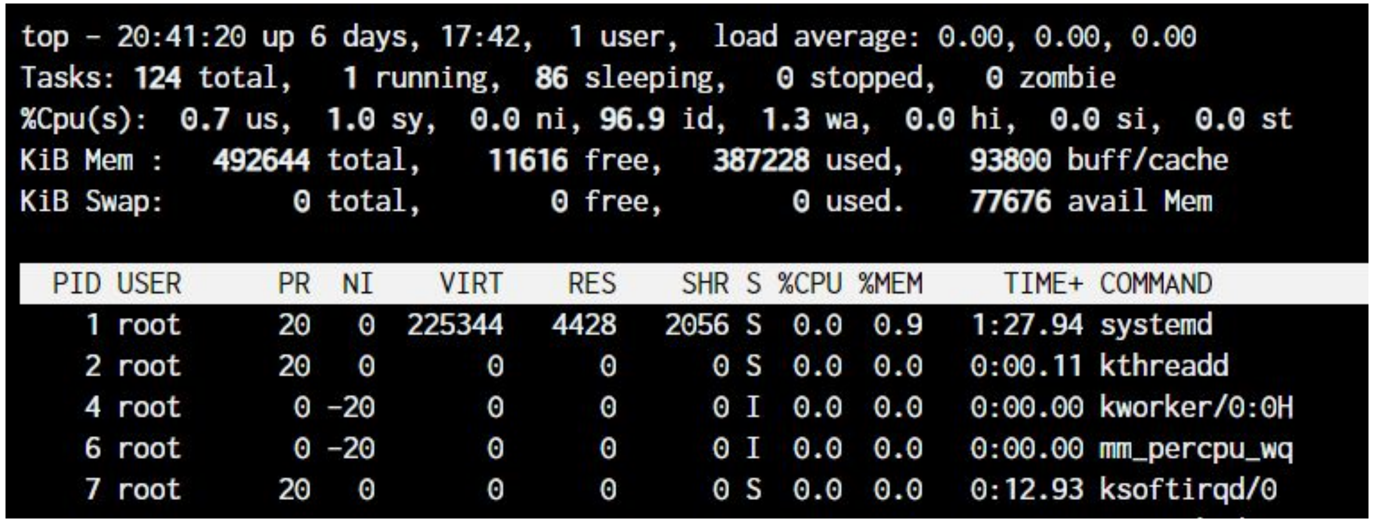
The **whatis** command gives us an explanation of what a command actually is. Similar to the whereis command, you'll receive the information for any command that you type after the **whatis** command.

```
root@ubuntu:~# whatis sudo sudo (8) - execute a command as another user
```

Linux Basic Commands…

top

A few sections earlier, we talked about the ps command. You observed that the ps command will output the active processes and end itself.
The top command is like a CLI version of the task manager in Windows. You get a live view of the processes and all the information accompanying those processes like memory usage, CPU usage, etc.
To get the top command, all you need to do is type the word **top** in your terminal.



```
top - 20:41:20 up 6 days, 17:42,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 124 total,   1 running,  86 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  1.0 sy,  0.0 ni, 96.9 id,  1.3 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   492644 total,    11616 free,   387228 used,    93800 buff/cache
KiB Swap:        0 total,        0 free,        0 used.    77676 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
    1 root      20   0  225344   4428   2056 S  0.0  0.9   1:27.94 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.11 kthreadd
    4 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 kworker/0:0H
    6 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0       0      0      0 S  0.0  0.0   0:12.93 ksoftirqd/0
```

Linux Basic Commands…

useradd

The **useradd** or **adduser** commands are the exact same commands where adduser is just a symbolic link to the useradd command. This command allows us to create a new user in Linux.

```
root@ubuntu:~# useradd JournalDev -d /home/JD Copy
```

The above command will create a new user named JournalDev with the home directory as **/home/JD**.

Linux Basic Commands…

useradd

Here are some common options used with the useradd command:

- -m: Create the user's home directory if it doesn't exist.
- -s <shell>: Set the user's login shell (e.g., -s /bin/bash sets the bash shell).
- -u <UID>: Set the numerical user ID for the new user.
- -g <group>: Set the initial login group for the new user.
- -G <group1>,<group2>: Set additional groups for the new user.
- -d <home_directory>: Set the user's home directory.
- -c <comment>: Set a comment for the user (usually the user's full name).
- -e <expiry_date>: Set an expiry date for the user account.

Linux Basic Commands…

usermod

The **usermod** command, on the other hand, is used to modify existing users. You can modify any value of the user including the groups, the permissions, etc.

For example, if you want to add more groups to the user, you can type in:

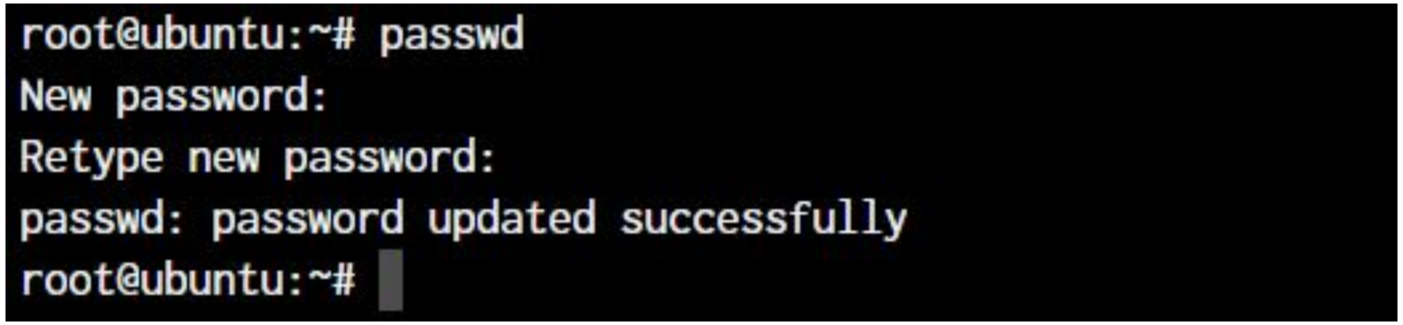
```
root@ubuntu:~# usermod JournalDev -a -G sudo, audio, mysql
```

Linux Basic Commands…

passwd

Now that you know how to create new users, let's also set the password for them.
The **passwd** command lets you set the password for your own account, or if you have the permissions, set the password for other accounts.

```
root@ubuntu:~# passwd
New password:
```



```
root@ubuntu:~# passwd
New password:
Retype new password:
passwd: password updated successfully
root@ubuntu:~#
```

If you add the username after **passwd**, you can set passwords for other users. Enter the new password twice and you're done. That's it! You will have a new password set for the user!

Directory Structure in **Linux…**

- In Linux/Unix operating system everything is a file even directories are files, files are files, and devices like mouse, keyboard, printer, etc are also files.

Types of files in the Linux system.

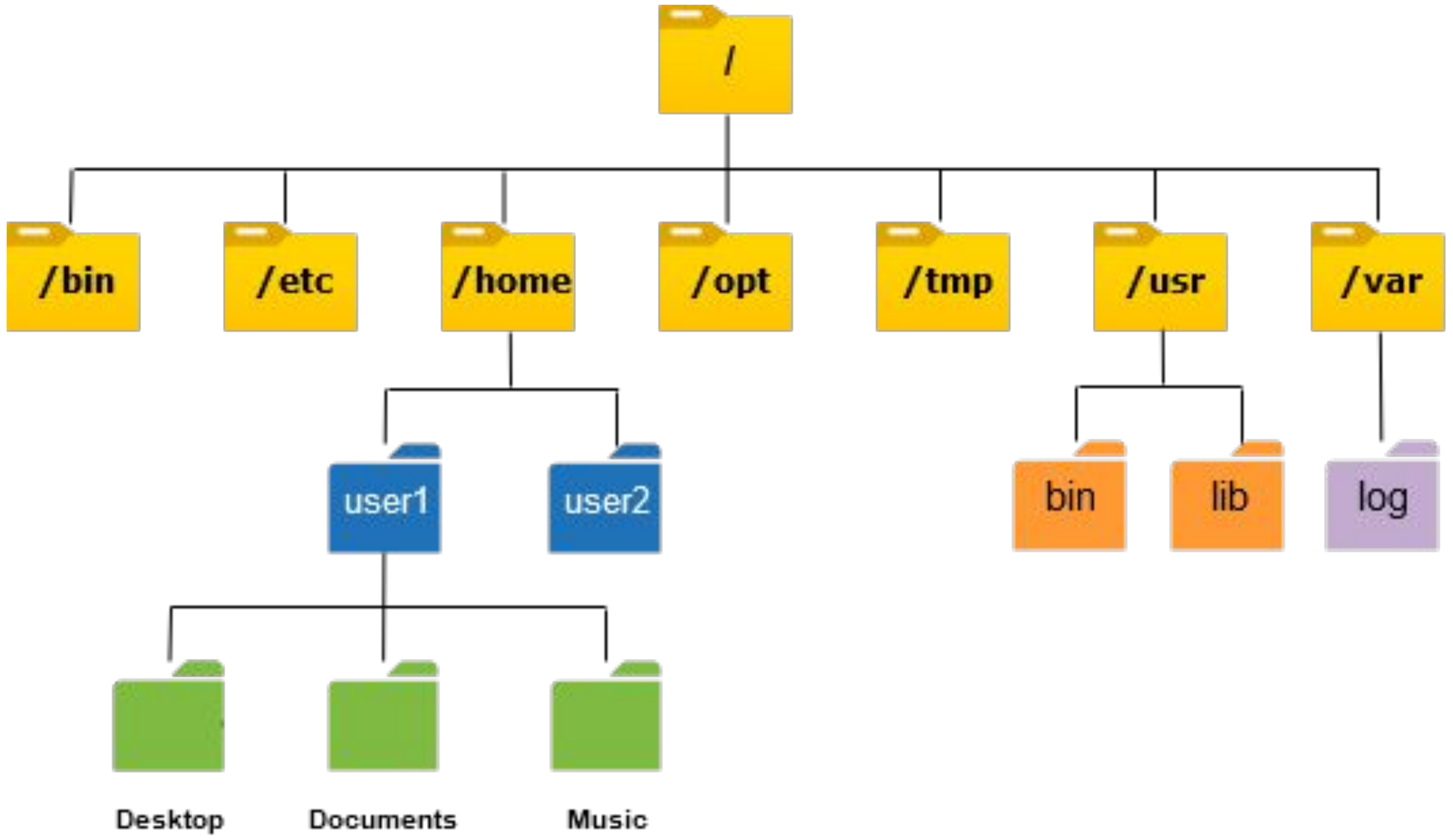
- **General Files:** It is also called ordinary files. It may be an image, video, program, or simple text file. These types of files can be in ASCII or Binary format. It is the most commonly used file in the Linux system.

- **Directory Files:** These types of files are a warehouse for other file types. It may be a directory file within a directory (subdirectory).

- **Device Files:** In a Windows-like operating system, devices like CD-ROM, and hard drives are represented as drive letters like F: G: H whereas in the Linux system devices are represented as files. As for example, /dev/sda1, /dev/sda2, and so on.

Directory Structure in **Linux…**

In Linux/Unix operating system everything is a file even directories are files, files are files, and devices like mouse, keyboard, printer, etc are also files.



Directory Structure in **Linux…**

/ – Root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user’s home directory, which is not same as /.

/bin – User Binaries

- Contains binary executables.
- Common Linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.

Directory Structure in **Linux…**

/sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.
- But, the Linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig

/etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/hosts, /etc/passwd

Directory Structure in **Linux…**

/dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/sda1, /dev/sdb1, /dev/usbmon0

/proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

Directory Structure in **Linux…**

/var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

/tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

Directory Structure in **Linux…**

/usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can’t find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can’t find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

/home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

Directory Structure in **Linux…**

/boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

/lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*
- For example: ld-2.11.1.so, libncurses.so.5.7

Directory Structure in Linux…

/opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

/mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

Directory Structure in **Linux…**

/media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

/srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data.

The End

BAITUSSALAM
—TECH PARK—