

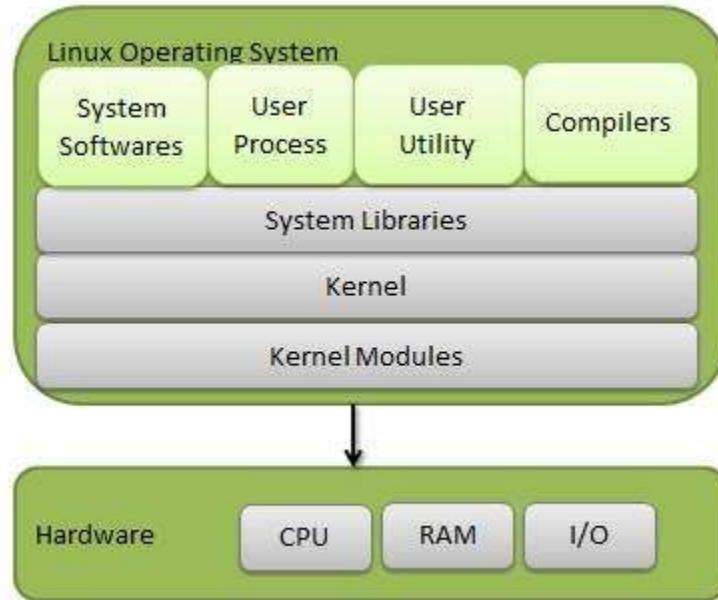
EMBEDDED LINUX

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System

Linux Operating System has primarily three components

- **Kernel** – Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Library** – System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.
- **System Utility** – System Utility programs are responsible to do specialized, individual level tasks.



Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and does not require any context switch and hence is very efficient and fast. Kernel

runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

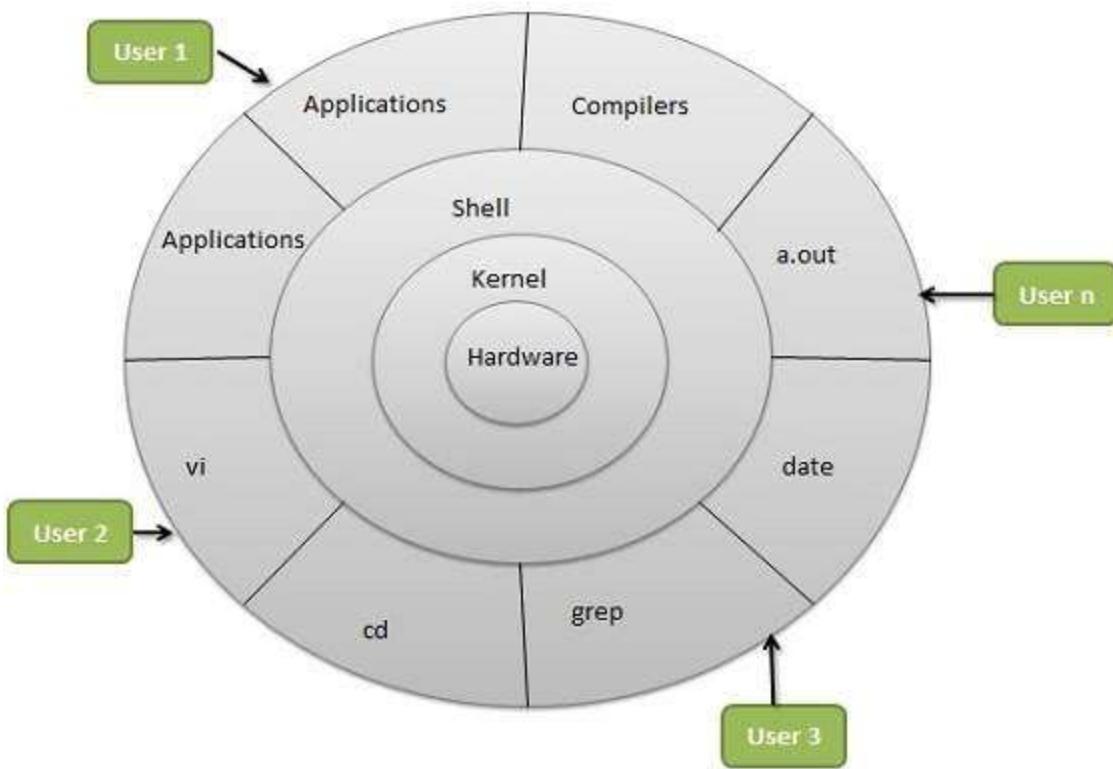
Basic Features

Following are some of the important features of Linux Operating System.

- **Portable** – Portability means software can work on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- **Open Source** – Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** – Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** – Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** – Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** – Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
- **Security** – Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Architecture

The following illustration shows the architecture of a Linux system –



The architecture of a Linux System consists of the following layers –

- **Hardware layer** – Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** – It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** – An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities** – Utility programs that provide the user most of the functionalities of an operating systems.

Look at Linux, the operating system and universal platform

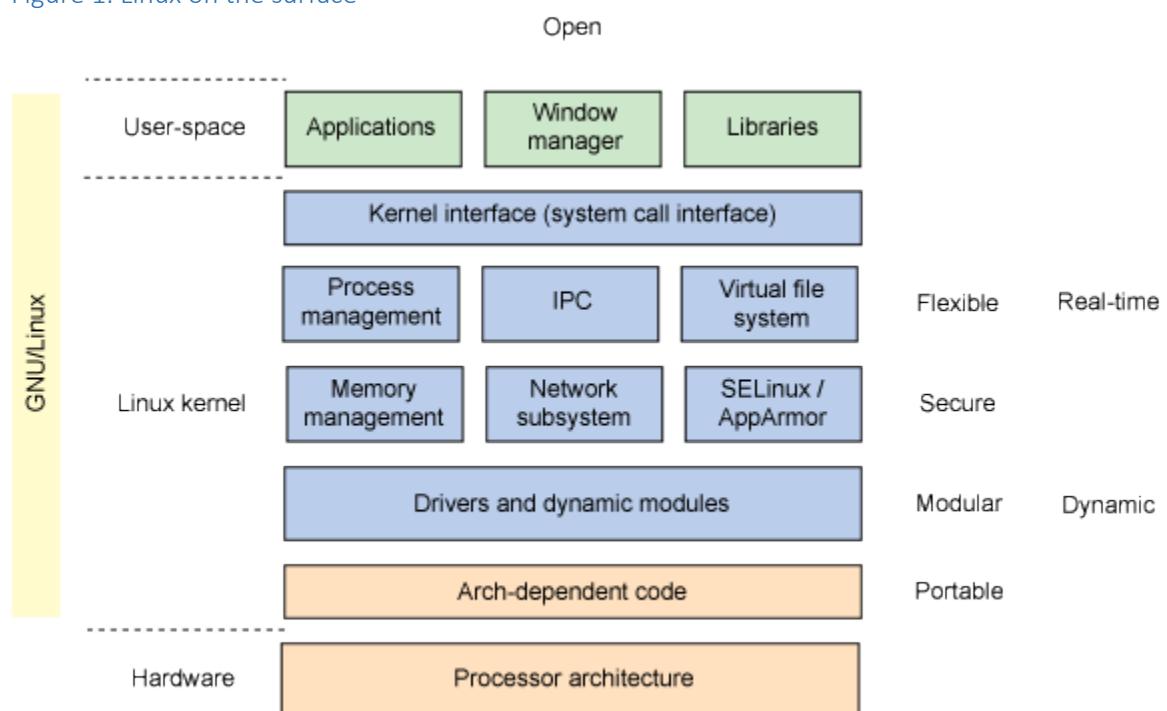
Linux® has come of age. In 2012, Linux will be 21 years old, a mature operating system with support for a spectrum of usage models. But it's hard to think of Linux as just an operating system—it's more like a chameleon. Its flexible and modular kernel addresses so many usage models (from the biggest supercomputer to the smallest embedded device) that it's difficult to classify it as anything other than an enabling technology. In fact, Linux is a platform. It's a key technology that enables the creation of new products, some of which were unknown just a short time ago.

Let's begin with a quick exploration of Linux, its basic architecture, and some of its important key principles. Then, look at how Linux applies these principles to a variety of usage models and why it's a platform, not just an operating system.

What is Linux?

On the surface, Linux is an operating system. As shown in [Figure 1](#), Linux consists of a kernel (the core code that manages hardware and software resources) and a collection of user applications (such as libraries, window managers, and applications).

Figure 1. Linux on the surface



This simple diagram shows key principles that are easily overlooked. At the bottom of the Linux stack is a set of architecture-dependent code that enables Linux on a vast array of hardware platforms (ARM, PowerPC, Tilera TILE, and more). This functionality, of course, is enabled by the GNU toolchain, which enables Linux *portability*.

Linux is in a class of its own in the arena of *portability*. The driver subsystem (which is vast in its capability) supports dynamically loaded modules with no performance impact, enabling *modularity* (in addition to a more *dynamic* platform). Linux also includes kernel-level security (in a number of schemes) enabling a *secure* platform. In the domain of external file systems, Linux enables the largest array of file system support of any operating system, enabling, as one example, *flexibility* through design modularity. Linux implements not only standard scheduling features but also *real-time* scheduling (including guarantees on interrupt latency).

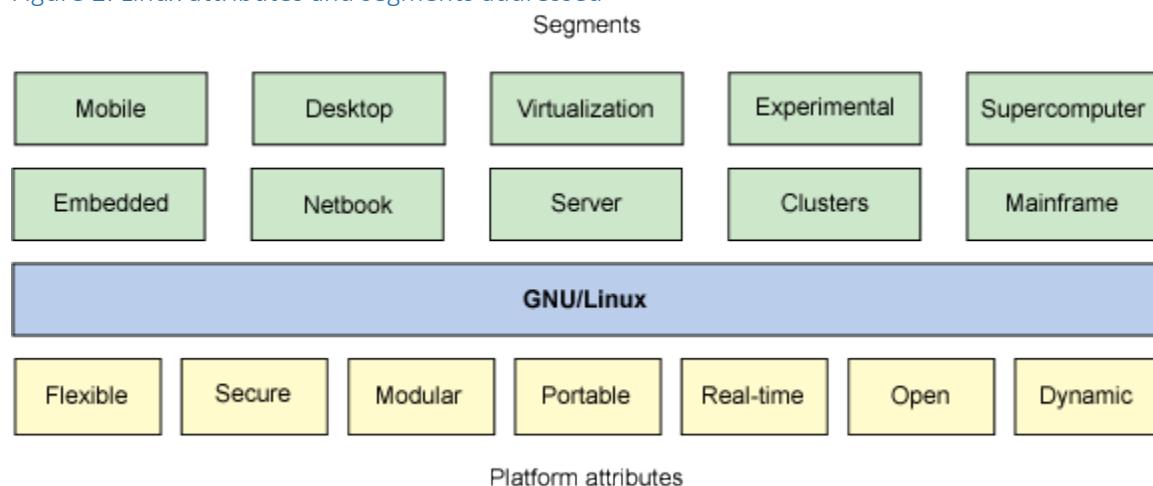
Finally, Linux is *open*, meaning that its source can be viewed and improved upon by practically anyone. This openness also minimizes the opportunities for exploits, creating a more *secure* platform. Many companies contribute to Linux, ensuring that it will continue to address a variety of usage models while maintaining its core properties.

These seven key principles are by no means the only attributes that Linux provides, but they enable Linux as a universal platform across a wide variety of usage models. Further, Linux is the same across these usage models—not just the design principles but the code itself. This cannot be said of other operating systems (such as Windows®—desktop, server, or embedded—or Mac OS X or Apple iOS), which fragment their offerings to support other usage models.

Where is Linux?

Where Linux is might be harder to answer than where it isn't. Linux, with its ability to morph and scale, can be found in all computing segments (and even some that are not yet fully defined). This section explores some of the major computing segments, including desktop/netbook, server, cluster, mainframe, supercomputer, handheld/tablet, embedded, virtualization, and experimental (see [Figure 2](#)).

Figure 2. Linux attributes and segments addressed



Desktop and netbook

Desktops and netbooks, where many people use Linux, is the area in which Linux struggles the most. Recent market share data indicates that Linux captures around 1.5% of the desktop market

but around 32% of the netbook market. These numbers might appear low, but as a developer, I tend to see Linux more than any other operating system.

Linux began as a simple experimental operating system, and with the introduction of XFree86 in 1994, a window manager showed the promise of a fledgling desktop operating system. Today, several window managers are available for Linux (both a blessing and a curse), allowing users to tailor its personality to their needs. Further, Linux scales automatically with processor capabilities (such as multicore and symmetric multithreading), efficiently scheduling processes across with performance in mind.

[Linux and the job market](#)

A recent survey from Dice.com and the Linux Foundation found that 81% of more than 2,000 respondents indicated that Linux hiring was a priority in 2012. Not only is Linux driving industries, it's driving careers, as well.

Server

In the server market (consisting of web servers, mail servers, Domain Name System servers, and other back-end devices), Linux rules. Recent surveys found that more than 60% of all servers run a form of Linux. Outside of traditional web services, Linux powers many of the biggest Internet properties (Facebook, eBay, Twitter, and Amazon, to name a few), with the varying usage models and requirements. Beyond traditional options (such as web or mail), Linux offers the largest array of web services (and varying options for those services).

Cluster and distributed computing

Linux is not only a staple in clusters and distributed computing models, it is a driving force and at the core of many new usage models. Two key models that are quickly growing today are cloud computing and big data.

Cloud computing is about delivery of IT as a service and relies on a cluster of shared resources that scale to the need of a given application. Clouds also rely on virtualization to support the automated management of nodes within a massive infrastructure. Within cloud environments, 66% rely on Linux as their primary platform.

Linux is also driving itself as the platform for data science. The Internet scales the volume of data that can be collected, and new problems arise in the processing of this data to identify its valuable patterns. What is now called *Big Data* was developed on Linux as a scalable way to manipulate data that exceeded prior traditional methods. Hadoop and its ecosystem are a result of the openness of Linux, along with an army of developers who are proficient with the platform.

Mainframe

In 1991, a well-known editor predicted that the last mainframe would be unplugged in early 1996. More than 20 years later, mainframes continue to be built and sold, and many run Linux. IBM

began supporting Linux on mainframes in 2000 (such as the popular IBM® System z®) and provides a common user experience across environments. A recent article from Michael Vizard documented that about 25% of new mainframe workloads rely on Linux. (See [Related topics](#).)

Supercomputer

Supercomputers are a constant arms race to hold the title of fastest, from the Oak Ridge National Laboratory's Jaguar supercomputer (2009) to the Chinese Tianhe-I (2010) to the current leader, Japan's RIKEN Kei computer (2011). In 2012, IBM's Sequoia supercomputer will be released and is expected to exceed the performance of RIKEN by a factor of two. What each of these supercomputers has in common is that they all run Linux. Linux is not only efficient, it's also adaptable to the various hardware platforms that push its performance. This shouldn't be surprising at all, given that more than 90% of supercomputers run Linux. (See [Related topics](#).)

Mobile devices and tablets

At the more constrained spectrum of consumer devices, mobile devices and tablets are demonstrating significant growth. These devices represent a Linux kernel coupled with a custom graphical user interface (GUI). A key example of this area is the Google Android operating system, which is used both in smart phones and in tablet computers. Today, more than 25% of smart phones run a form of Linux (primarily Android), with almost 40% of tablet computers running Android.

These devices rely on ARM-based processors (systems on a chip) for high performance and low power consumption. Regardless of the underlying platform, these are Linux devices, not forks of the kernel and applications.

Microsoft® recently confirmed that for their Windows on ARM (WOA) tablet, the only applications that will be supported are those being developed for the platform (in other words, you can't run old applications on the tablet). Compare this to Linux, which supports highly portable applications instead of a restricted and closed application ecosystem. (See [Related topics](#).)

Embedded

At the bottom of the spectrum are embedded devices, with varying degrees of constraints (processor performance, resources such as memory, and so on). Linux is ideal in most of these cases because of its ability to scale down and use any of the available embedded processors on the market. This flexibility makes Linux a highly used platform in televisions, in-car entertainment, navigation systems, and many other types of devices.

Linux is highly customizable and has a focus on low power consumption. To ensure the power focus, the Less Watts initiative tracks power consumption of Linux kernel releases. This project focuses mainly on Intel platforms but can also be useful for other processors.

Linux is a fairly standard offering for embedded devices and can determine the success or failure of the device (to support quick bring-up and development). One interesting recent device called the Raspberry Pi, an ARM-based credit card-sized computer, runs Linux and is intended as a

learning device to teach programming. The device is anticipated to be priced at US\$35 but is not yet available for purchase. (See [Related topics](#).)

Virtualization platform

One of the most interesting areas in which Linux drives innovation is in the virtualization domain. Linux is the operating system home to every kind of virtualization solution available, whether platform or para-virtualization, operating system virtualization, or more obscure ideas such as cooperative virtualization. Linux as an operating system is able to transform itself into a hypervisor (such as the Kernel Virtual Machine [KVM]) as well as hosting a number of research hypervisors. To bring additional efficiency to virtualization, Linux implements Kernel SamePage Merging to efficiently de-duplicate memory pages.

Linux is also driving the state of the art in a new advancement in virtualization called *nested virtualization*. *Nesting*, as the name implies, allows a hypervisor to host a guest hypervisor, which in turn hosts a set of guest virtual machines. Although at first glance an odd use case, nested virtualization will change cloud computing and extend the types of applications that can be hosted there. Today, Linux KVM supports nested virtualization.

Experimental platform

Last but not least is the foundation of Linux itself—an experimental platform through which many new ideas are being explored. In 1991, Linux was introduced as a toy operating system, 20 years after the first release of UNIX®. Today, Linux serves as a platform for experimentation in file system research, cluster computing, clouds, virtualization advancement, and stretches the limits by which a single operating system kernel can be applied to so many usage models. Linux as a platform enables accelerated experimentation through the use of both Linux and the massive array of open source components. The result is an array of interesting technologies built from Linux, including HP webOS, Google Chrome OS, and Android.

One interesting change introduced by Linux is the increasing irrelevance of the underlying hardware platform. Linux presents the same user experience regardless of the underlying hardware architecture. So whether a cloud is filled with AMD x86 servers or low-power ARM-based offerings, the applications running on Linux are abstracted from the physical architecture. This abstraction allows consumers to make decisions on platform based upon their requirements rather than being tied to common but archaic and inefficient architectures. Linux equals choice.

Linux is also a self-contained integrated development environment (IDE). In addition to hosting a world-class compiler toolchain (the GNC Compiler Collection), it hosts spectrum of tools ranging from debuggers, editors, version control systems, file tools, and shells and interpreters to help automate development tasks. Linux in this capacity makes it an ideal environment for software development and software research. (See [Related topics](#).)

Linux versatility

Supporting the various usage models defined here is simply a packaging option for Linux. Linux distributions address the desktop and server markets, where specialized distributions focus on embedded (such as uClinux, if your embedded device lacks a memory management unit). Anyone can take a Linux kernel and package a set of user applications for a specific usage model, taking advantage of the various benefits of Linux (the array of networking protocols and file systems, configurable and dynamic kernel, standard application programming interfaces). This is one of the reasons the fastest-growing smart phone platform runs Linux (with a customized UI for its personality).

Going further

If you compared Linux to a bridge, it would be a modern engineering marvel. Its distributed development model challenged the status quo, and the result is one of the most flexible software products ever created, spanning a variety of usage models from tiny embedded devices to massive supercomputers. Linux has shaped industries and led the way in cutting-edge research in cluster computing, file systems, clouds, and virtualization. Whatever computing environment is on the way, Linux will be there.

Inside the Linux boot process

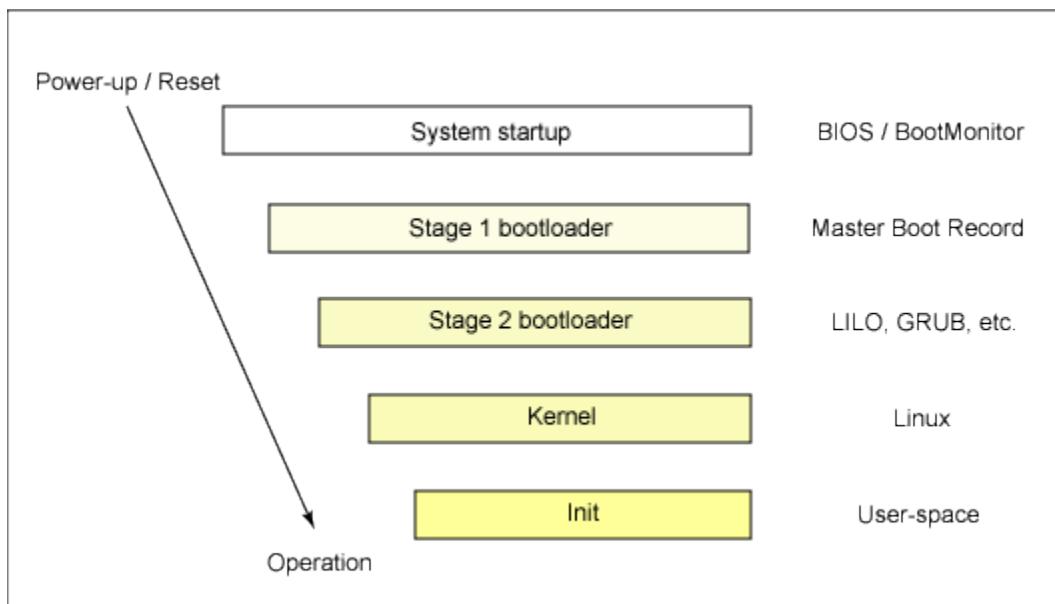
In the early days, bootstrapping a computer meant feeding a paper tape containing a boot program or manually loading a boot program using the front panel address/data/control switches. Today's computers are equipped with facilities to simplify the boot process, but that doesn't necessarily make it simple.

Let's start with a high-level view of Linux boot so you can see the entire landscape. Then we'll review what's going on at each of the individual steps. Source references along the way will help you navigate the kernel tree and dig in further.

Overview

Figure 1 gives you the 20,000-foot view.

Figure 1. The 20,000-foot view of the Linux boot process



When a system is first booted, or is reset, the processor executes code at a well-known location. In a personal computer (PC), this location is in the basic input/output system (BIOS), which is stored in flash memory on the motherboard. The central processing unit (CPU) in an embedded system invokes the reset vector to start a program at a known address in flash/ROM. In either case, the result is the same. Because PCs offer so much flexibility, the BIOS must determine which devices are candidates for boot. We'll look at this in more detail later.

When a boot device is found, the first-stage boot loader is loaded into RAM and executed. This boot loader is less than 512 bytes in length (a single sector), and its job is to load the second-stage boot loader.

When the second-stage boot loader is in RAM and executing, a splash screen is commonly displayed, and Linux and an optional initial RAM disk (temporary root file system) are loaded into memory. When the images are loaded, the second-stage boot loader passes control to the kernel image and the kernel is decompressed and initialized. At this stage, the second-stage boot loader checks the system hardware, enumerates the attached hardware devices, mounts the root device, and then loads the necessary kernel modules. When complete, the first user-space program (`init`) starts, and high-level system initialization is performed.

That's Linux boot in a nutshell. Now let's dig in a little further and explore some of the details of the Linux boot process.

System startup

The system startup stage depends on the hardware that Linux is being booted on. On an embedded platform, a bootstrap environment is used when the system is powered on, or reset. Examples include U-Boot, RedBoot, and MicroMonitor from Lucent. Embedded platforms are commonly shipped with a boot monitor. These programs reside in special region of flash memory on the target hardware and provide the means to download a Linux kernel image into flash memory and subsequently execute it. In addition to having the ability to store and boot a Linux image, these boot monitors perform some level of system test and hardware initialization. In an embedded target, these boot monitors commonly cover both the first- and second-stage boot loaders.

Extracting the MBR

To see the contents of your MBR, use this command:

```
# dd if=/dev/hda of=mbr.bin bs=512 count=1
# od -xa mbr.bin
```

The `dd` command, which needs to be run from root, reads the first 512 bytes from `/dev/hda` (the first Integrated Drive Electronics, or IDE drive) and writes them to the `mbr.bin` file. The `od` command prints the binary file in hex and ASCII formats.

In a PC, booting Linux begins in the BIOS at address 0xFFFF0. The first step of the BIOS is the power-on self test (POST). The job of the POST is to perform a check of the hardware. The second step of the BIOS is local device enumeration and initialization.

Given the different uses of BIOS functions, the BIOS is made up of two parts: the POST code and runtime services. After the POST is complete, it is flushed from memory, but the BIOS runtime services remain and are available to the target operating system.

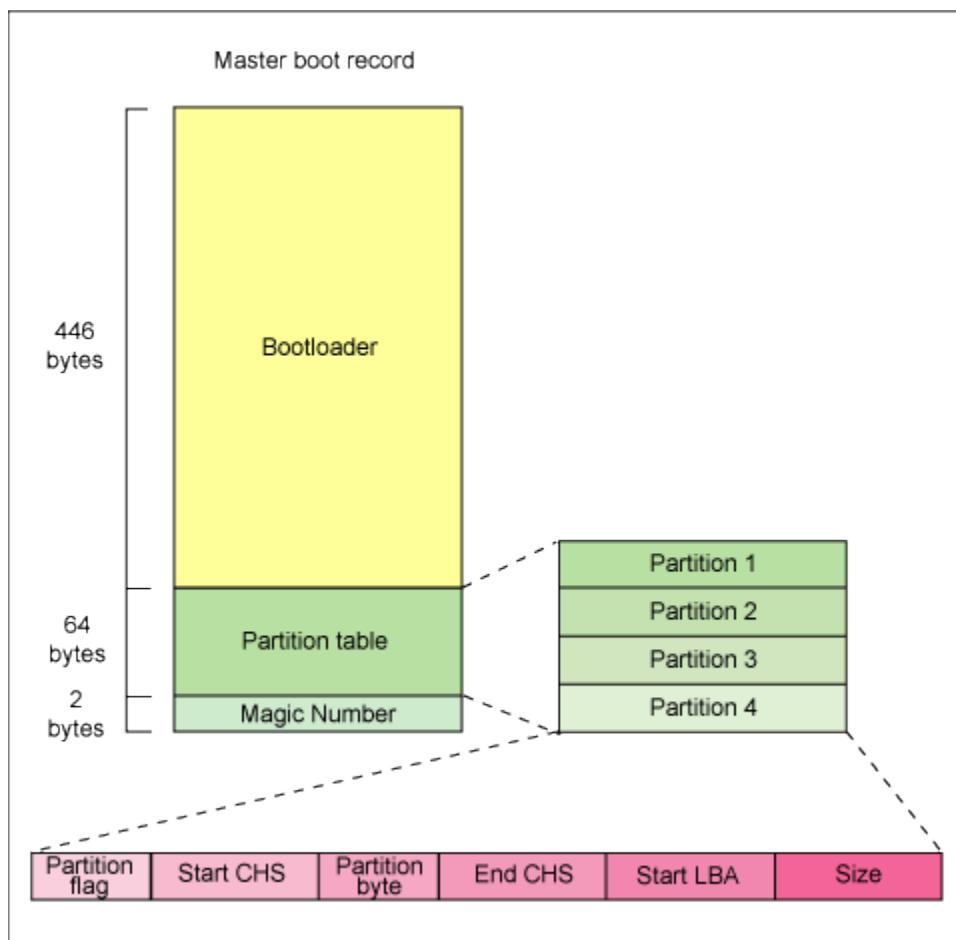
To boot an operating system, the BIOS runtime searches for devices that are both active and bootable in the order of preference defined by the complementary metal oxide semiconductor (CMOS) settings. A boot device can be a floppy disk, a CD-ROM, a partition on a hard disk, a device on the network, or even a USB flash memory stick.

Commonly, Linux is booted from a hard disk, where the Master Boot Record (MBR) contains the primary boot loader. The MBR is a 512-byte sector, located in the first sector on the disk (sector 1 of cylinder 0, head 0). After the MBR is loaded into RAM, the BIOS yields control to it.

Stage 1 boot loader

The primary boot loader that resides in the MBR is a 512-byte image containing both program code and a small partition table (see Figure 2). The first 446 bytes are the primary boot loader, which contains both executable code and error message text. The next sixty-four bytes are the partition table, which contains a record for each of four partitions (sixteen bytes each). The MBR ends with two bytes that are defined as the magic number (0xAA55). The magic number serves as a validation check of the MBR.

Figure 2. Anatomy of the MBR



The job of the primary boot loader is to find and load the secondary boot loader (stage 2). It does this by looking through the partition table for an active partition. When it finds an active partition, it scans the remaining partitions in the table to ensure that they're all inactive. When this is verified, the active partition's boot record is read from the device into RAM and executed.

Stage 2 boot loader

The secondary, or second-stage, boot loader could be more aptly called the kernel loader. The task at this stage is to load the Linux kernel and optional initial RAM disk.

GRUB stage boot loaders

The `/boot/grub` directory contains the `stage1`, `stage1.5`, and `stage2` boot loaders, as well as a number of alternate loaders (for example, CR-ROMs use the `iso9660_stage_1_5`).

The first- and second-stage boot loaders combined are called Linux Loader (LILO) or GRand Unified Bootloader (GRUB) in the x86 PC environment. Because LILO has some disadvantages that were corrected in GRUB, let's look into GRUB. (See many additional resources on GRUB, LILO, and related topics in the [Resources](#) section later in this article.)

The great thing about GRUB is that it includes knowledge of Linux file systems. Instead of using raw sectors on the disk, as LILO does, GRUB can load a Linux kernel from an ext2 or ext3 file system. It does this by making the two-stage boot loader into a three-stage boot loader. Stage 1 (MBR) boots a stage 1.5 boot loader that understands the particular file system containing the Linux kernel image. Examples include `reiserfs_stage1_5` (to load from a Reiser journaling file system) or `e2fs_stage1_5` (to load from an ext2 or ext3 file system). When the stage 1.5 boot loader is loaded and running, the stage 2 boot loader can be loaded.

With stage 2 loaded, GRUB can, upon request, display a list of available kernels (defined in `/etc/grub.conf`, with soft links from `/etc/grub/menu.lst` and `/etc/grub.conf`). You can select a kernel and even amend it with additional kernel parameters. Optionally, you can use a command-line shell for greater manual control over the boot process.

With the second-stage boot loader in memory, the file system is consulted, and the default kernel image and `initrd` image are loaded into memory. With the images ready, the stage 2 boot loader invokes the kernel image.

Kernel

Manual boot in GRUB

From the GRUB command-line, you can boot a specific kernel with a named `initrd` image as follows:

```
grub> kernel /bzImage-2.6.14.2
[Linux-bzImage, setup=0x1400, size=0x29672e]

grub> initrd /initrd-2.6.14.2.img
[Linux-initrd @ 0x5f13000, 0xcc199 bytes]

grub> boot
```

Uncompressing Linux... Ok, booting the kernel.

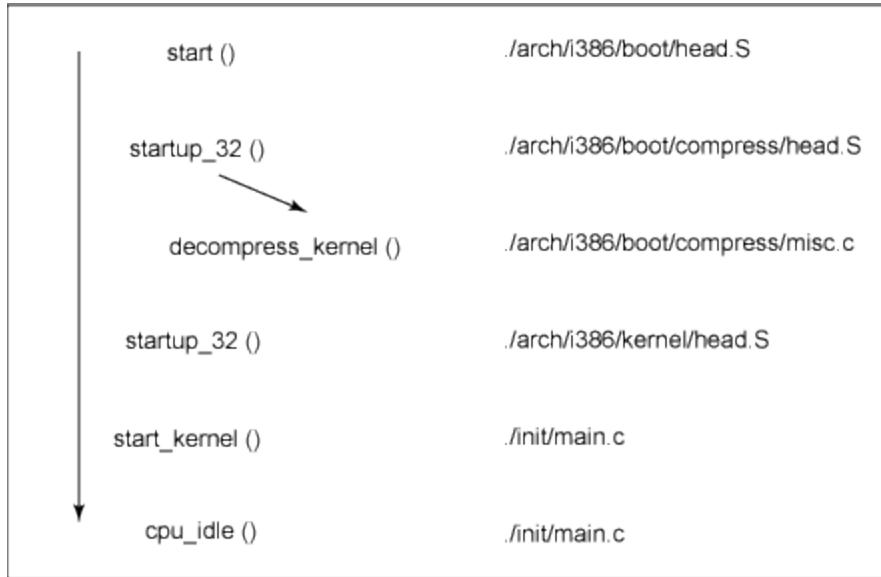
If you don't know the name of the kernel to boot, just type a forward slash (/) and press the Tab key. GRUB will display the list of kernels and initrd images.

With the kernel image in memory and control given from the stage 2 boot loader, the kernel stage begins. The kernel image isn't so much an executable kernel, but a compressed kernel image. Typically this is a zImage (compressed image, less than 512KB) or a bzImage (big compressed image, greater than 512KB), that has been previously compressed with zlib. At the head of this kernel image is a routine that does some minimal amount of hardware setup and then decompresses the kernel contained within the kernel image and places it into high memory. If an initial RAM disk image is present, this routine moves it into memory and notes it for later use. The routine then calls the kernel and the kernel boot begins.

When the bzImage (for an i386 image) is invoked, you begin at `./arch/i386/boot/head.S` in the `start` assembly routine (see Figure 3 for the major flow). This routine does some basic hardware setup and invokes the `startup_32` routine in `./arch/i386/boot/compressed/head.S`. This routine sets up a basic environment (stack, etc.) and clears the Block Started by Symbol (BSS). The kernel is then decompressed through a call to a C function called `decompress_kernel` (located in `./arch/i386/boot/compressed/misc.c`). When the kernel is decompressed into memory, it is called. This is yet another `startup_32` function, but this function is in `./arch/i386/kernel/head.S`.

In the new `startup_32` function (also called the swapper or process 0), the page tables are initialized and memory paging is enabled. The type of CPU is detected along with any optional floating-point unit (FPU) and stored away for later use. The `start_kernel` function is then invoked (`init/main.c`), which takes you to the non-architecture specific Linux kernel. This is, in essence, the `main` function for the Linux kernel.

Figure 3. Major functions flow for the Linux kernel i386 boot



With the call to `start_kernel`, a long list of initialization functions are called to set up interrupts, perform further memory configuration, and load the initial RAM disk. In the end, a call is made to `kernel_thread` (in `arch/i386/kernel/process.c`) to start the `init` function, which is the first user-space process. Finally, the idle task is started and the scheduler can now take control (after the call to `cpu_idle`). With interrupts enabled, the pre-emptive scheduler periodically takes control to provide multitasking.

During the boot of the kernel, the initial-RAM disk (`initrd`) that was loaded into memory by the stage 2 boot loader is copied into RAM and mounted. This `initrd` serves as a temporary root file system in RAM and allows the kernel to fully boot without having to mount any physical disks. Since the necessary modules needed to interface with peripherals can be part of the `initrd`, the kernel can be very small, but still support a large number of possible hardware configurations. After the kernel is booted, the root file system is pivoted (via `pivot_root`) where the `initrd` root file system is unmounted and the real root file system is mounted.

decompress_kernel output

The `decompress_kernel` function is where you see the usual decompression messages emitted to the display:

Uncompressing Linux... Ok, booting the kernel.

The `initrd` function allows you to create a small Linux kernel with drivers compiled as loadable modules. These loadable modules give the kernel the means to access disks and the file systems on those disks, as well as drivers for other hardware assets. Because the root file system is a *file system* on a disk, the `initrd` function provides a means of bootstrapping to gain access to the disk and mount the real root file system. In an embedded target without a hard disk, the `initrd` can be

the final root file system, or the final root file system can be mounted via the Network File System (NFS).

Init

After the kernel is booted and initialized, the kernel starts the first user-space application. This is the first program invoked that is compiled with the standard C library. Prior to this point in the process, no standard C applications have been executed.

In a desktop Linux system, the first application started is commonly `/sbin/init`. But it need not be. Rarely do embedded systems require the extensive initialization provided by `init` (as configured through `/etc/inittab`). In many cases, you can invoke a simple shell script that starts the necessary embedded applications.

Summary

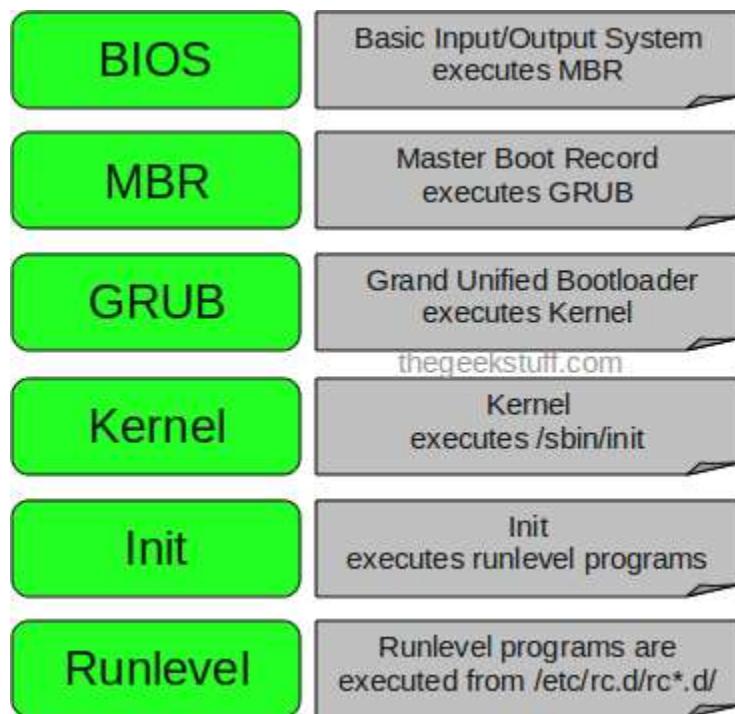
Much like Linux itself, the Linux boot process is highly flexible, supporting a huge number of processors and hardware platforms. In the beginning, the `loadlin` boot loader provided a simple way to boot Linux without any frills. The `LIFO` boot loader expanded the boot capabilities, but lacked any file system awareness. The latest generation of boot loaders, such as GRUB, permits Linux to boot from a range of file systems (from Minix to Reiser).

6 Stages of Linux Boot Process (Startup Sequence)

Press the power button on your system, and after few moments you see the Linux login prompt.

Have you ever wondered what happens behind the scenes from the time you press the power button until the Linux login prompt appears?

The following are the 6 high level stages of a typical Linux boot process.



1. BIOS

- BIOS stands for Basic Input/Output System
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 or F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS loads and executes the MBR boot loader.

2. MBR

- MBR stands for Master Boot Record.

- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5PAE)
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
    initrd /boot/initrd-2.6.18-194.el5PAE.img
```

- As you notice from the above info, it contains kernel and initrd image.
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. Kernel

- Mounts the root file system as specified in the “root=” in grub.conf
- Kernel executes the /sbin/init program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels
 - 0 – halt

- 1 – Single user mode
 - 2 – Multiuser, without NFS
 - 3 – Full multiuser mode
 - 4 – unused
 - 5 – X11
 - 6 – reboot
- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
- Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

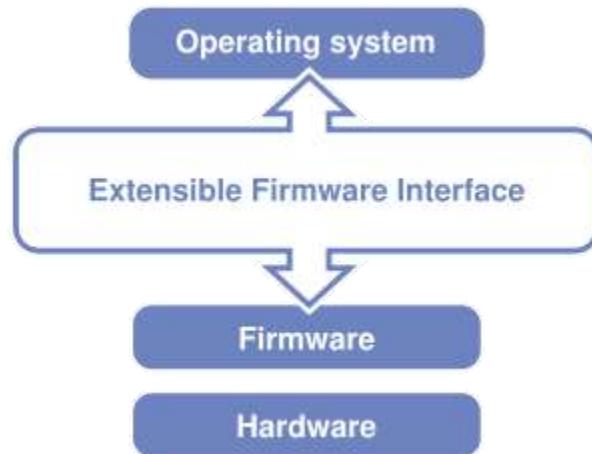
- When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail OK”. Those are the runlevel programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the following directories.
 - Run level 0 – /etc/rc.d/rc0.d/
 - Run level 1 – /etc/rc.d/rc1.d/
 - Run level 2 – /etc/rc.d/rc2.d/
 - Run level 3 – /etc/rc.d/rc3.d/
 - Run level 4 – /etc/rc.d/rc4.d/
 - Run level 5 – /etc/rc.d/rc5.d/
 - Run level 6 – /etc/rc.d/rc6.d/
- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.
- There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
- For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

BIOS vs UEFI

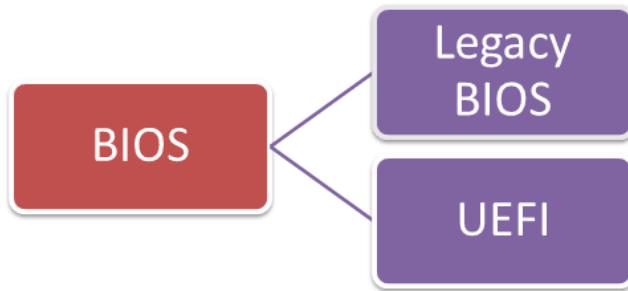
Basic Input-Output System Vs. Unified Extensible Firmware Interface

Introduction:

Basic Input-Output System (BIOS) and Unified Extensible Firmware Interface (UEFI) are two firmware interfaces for computers which work as an interpreter between the operating system and the computer firmware. Both of these interfaces are used at the startup of the computer to initialize the hardware components and start the operating system which is stored on the hard drive.



Role wise both are same. The older one is now called as Legacy BIOS and the newer version is called as UEFI.



BIOS:

BIOS works by reading the first sector of the hard drive (MBR) which has the next device's address to initialize or code to execute. BIOS also selects the boot device that needs to be initialized for starting the operating system. Since BIOS has been in use since the very

beginning (since 1975), it still works in 16-bit mode, limiting the amount of code that can be read and executed from the firmware ROM.

UEFI:

UEFI does the same task a little differently. It stores all the information about initialization and startup in an **.efi** file instead of the firmware. This file is stored on the hard drive inside a special partition called EFI System Partition (ESP). The ESP partition will also contain the boot loader programs for the Operating System installed on the computer. UEFI is meant to completely replace BIOS in the future and bring in many new features and enhancements that can't be implemented through BIOS.

Some of those features are discussed below:

(1) Breaking out of size limitations:

BIOS uses the Master Boot Record (MBR) to save information about the hard drive data while UEFI uses the GUID partition table (GPT). The major difference between the two is that MBR uses 32-bit entries in its table which limits the total physical partitions to only 4. Each partition can only be a maximum of 2TB in size, while GPT uses 64-bit entries in its table which dramatically extends the support for size possibilities of the hard drive up to Zeta Byte and maximum 128 partitions. (Read more on difference between MBR and GPT in my next blog article).

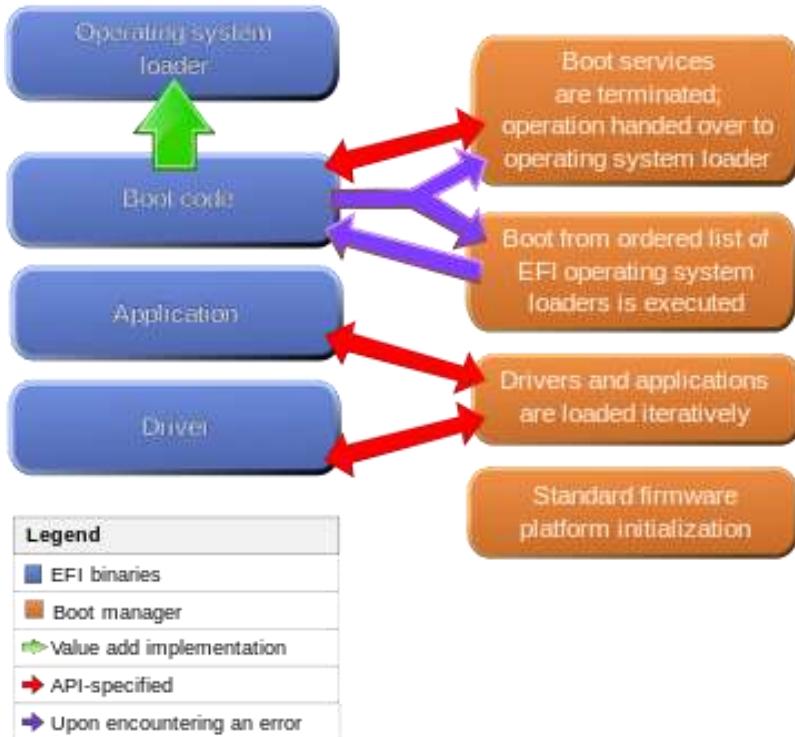
(2) Speed and performance:

Since UEFI is platform independent, it may be able to enhance the boot time and speed of the computer. This is especially the case when you have large hard drives installed in your computer. This enhancement depends upon how UEFI is configured to run. UEFI can perform better while initializing the hardware devices. Normally this speed enhancement is a fraction of the total boot time, so you will not see a huge difference in overall boot time. Developers can make use of UEFI shell environment which can execute command from other UEFI apps optimizing the performance of the system further.

(3) Security:

'Secure boot' is a feature of UEFI that has been implemented in Windows 8 recently. The biggest benefit of UEFI is its security over BIOS. UEFI can allow only authentic drivers and services to load at boot time, making sure that no malware can be loaded at computer startup. Microsoft implemented this feature to counter piracy issues in Windows, while Mac has been using UEFI for quite some time now. Secure Boot works by requiring a digital signature of boot loaders which should require digital signature by the Kernel. This process continues until the operating system is completely started. This secure boot

feature is also one of the reason why it is more difficult to install another operating system on a Windows machine.

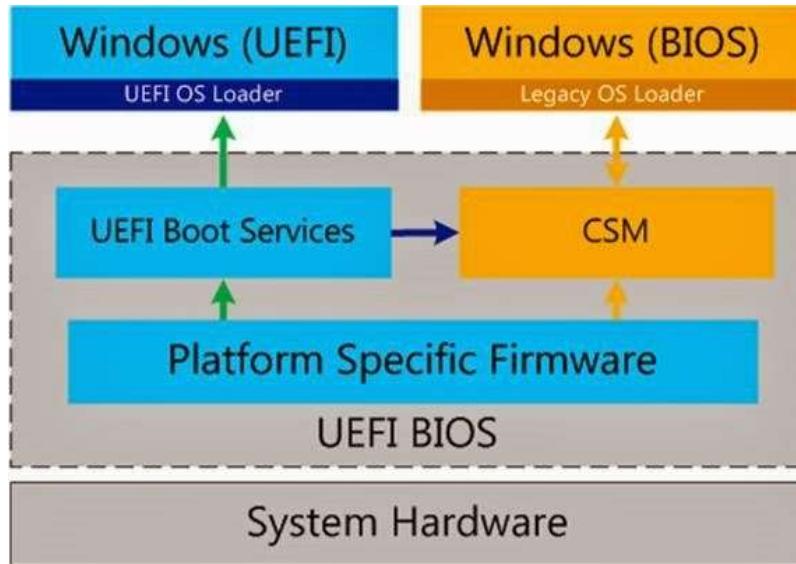


(4) Backward Compatibility using Compatibility Support Module (CSM):

For backwards compatibility, most of the UEFI implementations on PC-class machines also support booting in legacy BIOS mode from MBR-partitioned disks, through the *Compatibility Support Module (CSM)* which provides legacy BIOS compatibility. In that scenario, booting is performed in the same way as on legacy BIOS-based systems, by ignoring the partition table and relying on the content of a boot sector.

BIOS booting from MBR-partitioned disks is commonly called *B/OS-MBR*, regardless of it being performed on UEFI or legacy BIOS-based systems. As a side note, booting legacy BIOS-based systems from GPT disks is also possible, and it is commonly called *B/OS-GPT*.

Despite the fact MBR partition tables are required to be fully supported within the UEFI specification, some UEFI firmware immediately switch to the BIOS-based CSM booting depending on the type of boot disk's partition table, thus preventing UEFI booting to be performed from EFI System partitions on MBR-partitioned disks. Such a scheme is commonly called *UEFI-MBR*.



(5) Network boot support:

UEFI specification includes support for booting over network through the Preboot eXecution Environment (PXE). Underlying network protocols include Internet Protocol (IPv4 and IPv6), User Datagram Protocol (UDP), Dynamic Host Configuration Protocol (DHCP) and Trivial File Transfer Protocol (TFTP).

Also included support for boot images remotely stored on storage area networks (SANs), with Internet Small Computer System Interface (iSCSI) and Fibre Channel over Ethernet (FCoE) as supported protocols for accessing the SANs.

(6) Boot Manager:

The UEFI specification defines a "boot manager", a firmware policy engine that is in charge of loading the operating system loader and all necessary drivers. The boot configuration is controlled by a set of global NVRAM variables, including boot variables that indicate the paths to the loaders.

Operating system loaders are a class of the UEFI applications. As such, they are stored as files on a file system that can be accessed by the firmware, called EFI System partition (ESP). UEFI defines a specific version of FAT, which encompasses FAT32 file systems on ESPs, and FAT16 and FAT12 on removable media. Supported partition table schemes include MBR and GPT, as well as El Torito volumes on optical disks. UEFI does not rely on a boot sector, although ESP provides space for it as part of the backwards compatibility. UEFI booting from GPT disks is commonly called *UEFI-GPT*.

Boot loaders can also be automatically detected by the UEFI firmware, to enable booting from removable devices. Auto-detection relies on a standardized file path to the operating system loader, depending on the actual architecture to boot. Format of the file path is defined as /BOOT/BOOT.EFI, e.g. /efi/BOOT/BOOTX64.EFI

Technical differences between the two:

Legacy BIOS V.S. UEFI

	Legacy BIOS	UEFI
Language	Assembly	C (99%)
Resource	Interrupt Hardcode Memory Access Hardcode I/O Access	Driver \ Protocols
Processor	x86 16-bits	CPU Protected Mode (Flat Mode)
Target	Binary Code	Removable Binary Drivers
Expand	Hook Interrupt	Load Driver
OS Bridge	ACPI (SMI, INT Service)	RunTime Driver Service AED (ACPI EFI Protocol Driver) (ACPI, SMI)
3rd Party ISV & IHV	Bad for Support	Easy for Support and for Multi Platforms

The origin of UEFI:

The original **EFI (Extensible Firmware Interface)** specification was developed by Intel. Some of its practices and data formats mirror ones from Windows. In 2005, UEFI deprecated EFI 1.10 (final release of EFI). The UEFI specification is managed by the Unified EFI Forum.

The original motivation for EFI came during early development of the first Intel–HP Itanium systems in the mid-1990s. BIOS limitations (such as 16-bit processor mode, 1 MB addressable space and PC AT hardware) were unacceptable for the larger server platforms Itanium was targeting. The effort to address these concerns was initially called *Intel Boot Initiative*, which began in 1998 and was later renamed EFI.

In July 2005, Intel ceased development of the EFI specification at version 1.10, and contributed it to the Unified EFI Forum, which has evolved the specification as the Unified Extensible Firmware Interface (UEFI). The original EFI specification remains owned by Intel, which exclusively provides licenses for EFI-based products, but the UEFI specification is owned by the Forum.

Version 2.1 of the UEFI (*Unified Extensible Firmware Interface*) specification was released on 7 January 2007. It added cryptography, network authentication and the User

Interface Architecture (Human Interface Infrastructure in UEFI). The current UEFI specification, version 2.4, was approved in July 2013.

Criticism on the development:

Numerous digital rights activists have protested against UEFI. Ronald G. Minnich, a co-author of Coreboot, and Cory Doctorow, a digital rights activist, have criticized EFI as an attempt to remove the ability of the user to truly control the computer. It does not solve any of the BIOS's long-standing problems of requiring two different drivers—one for the firmware and one for the operating system—for most hardware.

Open source project TianoCore also provides the UEFI interfaces. Tiano Core lacks the specialized drivers that initialize chipset functions, which are instead provided by Coreboot, of which TianoCore is one of many payload options. The development of Coreboot requires cooperation from chipset manufacturers to provide the specifications needed to develop initialization drivers.

Conclusion:

There are several benefits as well as drawbacks of UEFI. It is not very common and is not supported by every computer or device. Its built-in boot manager means that there is no need for separate boot loaders. The biggest benefit of UEFI is that it can work alongside BIOS. It can sit on top of BIOS and work independently. BIOS can still be used in devices that do not require large storage or security.

(Contents: Wikipedia.com and MakeTechEasier.com)

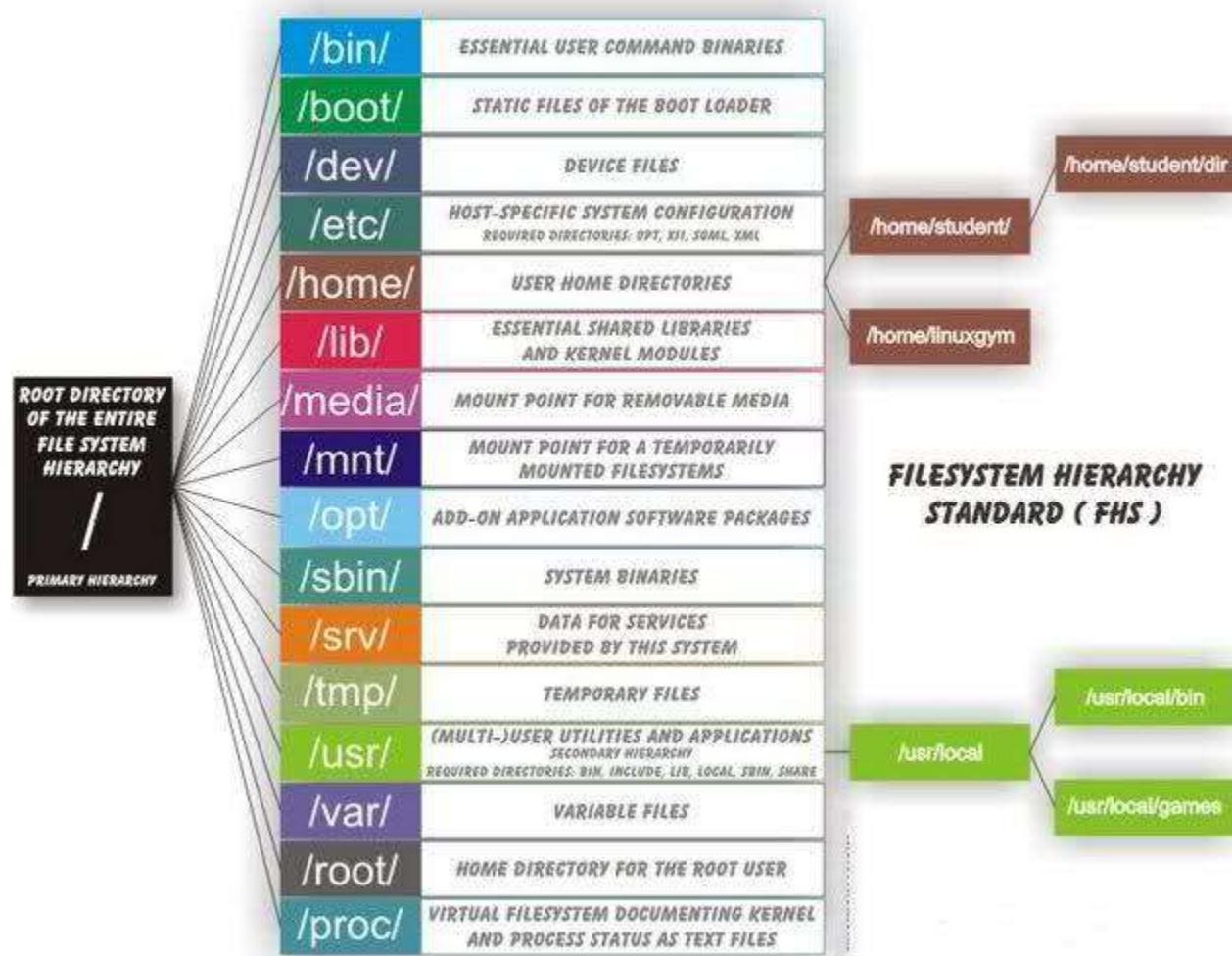
Linux Directory Structure and Important Files Paths Explained

For any person, who does not have a sound knowledge of **Linux Operating System** and **Linux File System**, dealing with the files and their location, their use may be horrible, and a newbie may really mess up.

This article is aimed to provide the information about **Linux File System**, some of the important files, their **usability** and **location**.

Linux Directory Structure Diagram

A standard **Linux** distribution follows the directory structure as provided below with Diagram and explanation.



Linux Directory Structure

Each of the above directory (which is a file, at the first place) contains important information, required for booting to device drivers, configuration files, etc. Describing briefly the purpose of each directory, we are starting hierarchically.

1. **/bin** : All the executable binary programs (file) required during booting, repairing, files required to run into single-user-mode, and other important, basic commands viz., [cat](#), [du](#), [df](#), [tar](#), [rpm](#), [wc](#), [history](#), etc.
2. **/boot** : Holds important files during [boot-up process](#), including **Linux Kernel**.
3. **/dev** : Contains device files for all the hardware devices on the machine e.g., **cdrom**, **cpu**, etc
4. **/etc** : Contains Application's configuration files, **startup**, **shutdown**, **start**, **stop** script for every individual program.
5. **/home** : Home directory of the users. Every time a new user is created, a directory in the name of user is created within home directory which contains other directories like **Desktop**, **Downloads**, **Documents**, etc.
6. **/lib** : The Lib directory contains **kernel modules** and **shared library** images required to boot the system and run commands in root file system.
7. **/lost+found** : This Directory is installed during installation of **Linux**, useful for recovering files which may be broken due to unexpected **shut-down**.
8. **/media** : Temporary mount directory is created for removable devices viz., **media/cdrom**.
9. **/mnt** : Temporary mount directory for [mounting file system](#).
10. **/opt** : Optional is abbreviated as opt. Contains third party application software. Viz., [Java](#), etc.
11. **/proc** : A virtual and pseudo file-system which contains information about **running process** with a particular **Process-id** aka **pid**.
12. **/root** : This is the home directory of root user and should never be confused with '/'
13. **/run** : This directory is the only clean solution for **early-runtime-dir** problem.
14. **/sbin** : Contains binary executable programs, required by **System Administrator**, for **Maintenance**. Viz., [iptables](#), [fdisk](#), [ifconfig](#), swapon, reboot, etc.
15. **/srv** : Service is abbreviated as 'srv'. This directory contains server specific and service related files.
16. **/sys** : Modern Linux distributions include a **/sys** directory as a **virtual filesystem**, which stores and allows modification of the devices connected to the system.
17. **/tmp** : System's Temporary Directory, Accessible by users and root. Stores temporary files for **user** and **system**, till next boot.
18. **/usr** : Contains executable **binaries**, **documentation**, **source code**, **libraries** for second level program.
19. **/var** : Stands for variable. The contents of this file is expected to grow. This directory contains **log**, **lock**, **spool**, **mail** and **temp** files.

Exploring Important file, their location and their Usability

Linux is a complex system which requires a more complex and efficient way to **start, stop, maintain** and **reboot** a system unlike **Windows**. There is a well defined configuration **files, binaries, man pages, info files**, etc. for every **process** in **Linux**.

1. **/boot/vmlinuz** : The **Linux Kernel** file.
2. **/dev/hda** : Device file for the first **IDE HDD (Hard Disk Drive)**
3. **/dev/hdc** : Device file for the **IDE Cdrom**, commonly
4. **/dev/null** : A pseudo device, that don't exist. Sometime garbage output is redirected to **/dev/null**, so that it gets lost, forever.
5. **/etc/bashrc** : Contains system **defaults** and **aliases** used by bash shell.
6. **/etc/crontab** : A **shell script** to run specified commands on a predefined time Interval.
7. **/etc/exports** : Information of the file system available on **network**.
8. **/etc/fstab** : Information of **Disk Drive** and their mount point.
9. **/etc/group** : Information of **Security Group**.
10. **/etc/grub.conf** : grub **bootloader** configuration file.
11. **/etc/init.d** : Service **startup** Script.
12. **/etc/lilo.conf** : lilo **bootloader** configuration file.
13. **/etc/hosts** : Information of **Ip addresses** and corresponding **host names**.
14. **/etc/hosts.allow** : List of **hosts allowed** to access services on the local machine.
15. **/etc/host.deny** : List of **hosts denied** to access services on the local machine.
16. **/etc/inittab** : INIT process and their interaction at various **run level**.
17. **/etc/issue** : Allows to edit the **pre-login** message.
18. **/etc/modules.conf** : Configuration files for **system modules**.
19. **/etc/motd** : **motd** stands for **Message Of The Day**, The Message users gets upon login.
20. **/etc/mtab** : Currently mounted **blocks** information.
21. **/etc/passwd** : Contains **password** of system **users** in a shadow file, a security implementation.
22. **/etc/printcap** : **Printer** Information
23. **/etc/profile** : Bash shell **defaults**
24. **/etc/profile.d** : Application script, executed after **login**.
25. **/etc/rc.d** : Information about **run level** specific script.
26. **/etc/rc.d/init.d** : Run Level **Initialisation** Script.
27. **/etc/resolv.conf** : Domain Name Servers (**DNS**) being used by System.
28. **/etc/securetty** : Terminal List, where **root** login is possible.
29. **/etc/skel** : Script that populates new user **home** directory.
30. **/etc/termcap** : An **ASCII** file that defines the behaviour of **Terminal, console** and **printers**.
31. **/etc/X11** : Configuration files of **X-window** System.
32. **/usr/bin** : Normal user **executable** commands.
33. **/usr/bin/X11** : Binaries of **X windows** System.
34. **/usr/include** : Contains include files used by 'c' program.
35. **/usr/share** : Shared directories of **man files, info files**, etc.
36. **/usr/lib** : Library files which are required during program **compilation**.
37. **/usr/sbin** : Commands for **Super User**, for System Administration.

38. **/proc/cpuinfo** : CPU Information
39. **/proc/filesystems** : File-system **Information** being used currently.
40. **/proc/interrupts** : Information about the current **interrupts** being utilised currently.
41. **/proc/iports** : Contains all the **Input/Output** addresses used by devices on the server.
42. **/proc/meminfo** : **Memory Usages** Information.
43. **/proc/modules** : Currently using **kernel** module.
44. **/proc/mount** : Mounted **File-system** Information.
45. **/proc/stat** : Detailed **Statistics** of the current System.
46. **/proc/swaps** : **Swap** File Information.
47. **/version** : Linux **Version** Information.
48. **/var/log/lastlog** : log of last **boot** process.
49. **/var/log/messages** : log of messages produced by **syslog** daemon at boot.
50. **/var/log/wtmp** : list login **time** and **duration** of each user on the system currently.

That's all for now. Keep connected to **Tecmint** for any **News** and post related to **Linux** and **Foss** world. Stay healthy and Don't forget to give your value-able comments in comment section.

How to Find Out What Version of Linux You Are Running

There are several ways of knowing the version of Linux you are running on your machine as well as your distribution name and kernel version plus some extra information that you may probably want to have in mind or at your fingertips.

Therefore, in this simple yet important guide for new Linux users, I will show you how to do just that. Doing this may seem to be relatively easy task, however, having a good knowledge of your system is always a recommended practice for a good number of reasons including installing and running the appropriate packages for your Linux version, for easy reporting of bugs coupled with many more.

With that said, let us proceed to how you can figure out [information about your Linux distribution](#).

Find Out Linux Kernel Version

We will use **uname** command, which is used to print your Linux system information such as kernel version and release name, network hostname, machine hardware name, processor architecture, hardware platform and the operating system.

To find out which version of Linux kernel you are running, type:

```
$ uname -or
```

The screenshot shows two terminal windows side-by-side. Both windows display the command `uname -or` and its output. A red arrow points from the text "My Linux Version on Fedora 24" to the top window, and another red arrow points from the text "My Linux Version on Linux Mint 17" to the bottom window.

```
[root@TecMint ~]# uname -or
4.5.5-300.fc24.x86_64 GNU/Linux
[root@TecMint ~]#
```

My Linux Version on Fedora 24

```
tecmint@TecMint ~ $ uname -or
4.4.0-21-generic GNU/Linux
tecmint@TecMint ~ $
```

My Linux Version on Linux Mint 17

Shows Current Linux Kernel Version Running on System

In the preceding command, the option `-o` prints operating system name and `-r` prints the kernel release version.

You can also use `-a` option with **uname** command to print all system information as shown:

```
$ uname -a
```

[root@TecMint ~]# uname -a
Linux TecMint.com 4.5.5-300.fc24.x86_64 #1 SMP Tue May 19 13:05:32 UTC 2016 x86_64 x86_64 x86_64
GNU/Linux

Linux System Information on My Fedora 24

tecmint@TecMint ~ \$ uname -a
Linux TecMint 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 x86_64
GNU/Linux

Linux System Information on Linux Mint 17

Shows Linux System Information

Next, we will use [/proc file system](#), that stores information about processes and other system information, it's mapped to `/proc` and mounted at boot time.

Simply type the command below to display some of your system information including the Linux kernel version:

```
$ cat /proc/version
```

```
[root@TecMint ~]# cat /proc/version
Linux version 4.5.5-300.fc24.x86_64 (mockbuild@b
kernel101.phx2.fedoraproject.org) (gcc version 6.
1.1 20160510 (Red Hat 6.1.1-2) (GCC) ) #1 SMP Th
u May 19 13:05:32 UTC 2016
```

Shows My Fedora 24 Linux System Information

```
tecmint@TecMint ~ $ cat /proc/version
Linux version 4.4.0-21-generic (buildd@lgw01-21)
(gcc version 5.3.1 20160413 (Ubuntu 5.3.1-14ubu
ntu2) ) #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2
016
```

Shows My Linux Mint 17 System Information

Shows Linux System Information

From the image above, you have the following information:

1. Version of the Linux (kernel) you are running: **Linux version 4.5.5-300.fc24.x86_64**
2. Name of the user who compiled your kernel: **mockbuild@bkernel101.phx2.fedoraproject.org**
3. Version of the GCC compiler used for building the kernel: **gcc version 6.1.1 20160510**
4. Type of the kernel: **#1 SMP** (Symmetric MultiProcessing kernel) it supports systems with multiple CPUs or multiple CPU cores.
5. Date and time when the kernel was built: **Thu May 19 13:05:32 UTC 2016**

Find Out Linux Distribution Name and Release Version

The best way to determine a Linux distribution name and release version information is using `cat /etc/os-release` command, which works on almost all Linux system.

```
----- On Red Hat Linux -----
$ cat /etc/redhat-release
----- On CentOS Linux -----
$ cat /etc/centos-release
----- On Fedora Linux -----
$ cat /etc/fedora-release
----- On Debian Linux -----
$ cat /etc/debian_version
----- On Ubuntu and Linux Mint -----
$ cat /etc/lsb-release
----- On Gentoo Linux -----
$ cat /etc/gentoo-release
----- On SuSE Linux -----
$ cat /etc/SuSE-release
```

```
[tecmint@server ~]$ cat /etc/centos-release  
CentOS Linux release 7.2.1511 (Core)
```

```
[root@TecMint ~]# cat /etc/fedora-release  
Fedora release 24 (Twenty Four)
```

```
tecmint@TecMint:~$ cat /etc/lsb-release  
DISTRIB_ID=Ubuntu  
DISTRIB_RELEASE=16.04  
DISTRIB_CODENAME=xenial  
DISTRIB_DESCRIPTION="Ubuntu 16.04.1 LTS"  
tecmint@TecMint:~$
```

```
tecmint@TecMint ~ $ cat /etc/lsb-release  
DISTRIB_ID=LinuxMint  
DISTRIB_RELEASE=18  
DISTRIB_CODENAME=sarah  
DISTRIB_DESCRIPTION="Linux Mint 18 Sarah"
```

Find Linux Distribution Name and Release Version

In this article, we walked through a brief and simple guide intended to help new Linux user find out the Linux version they are running and also get to know their Linux distribution name and version from the shell prompt.

Perhaps it can also be useful to advanced users on one or two occasions. Lastly, to reach us for any assistance or suggestions you wish to offer, make use of the feedback form below.

1. sudo

This SuperUserDo is the most important command Linux newbies will use. Every single command that needs root's permission, need this sudo command. You can use sudo before each command that requires root permissions -

```
$ sudo su
```

2. ls (list)

Just like the other, you often want to see anything in your directory. With list command, the terminal will show you all the files and folders of the directory that you're working in. Let's say I'm in the /home folder and I want to see the directories & files in /home.

```
/home$ ls
```

ls in /home returns the following -
imad lost+found

3. cd

Changing directory (cd) is the main command that always be in use in terminal. It's one of the most Linux basic commands. Using this is easy. Just type the name of the folder you want to go in from your current directory. If you want to go up just do it by giving double dots (..) as the parameter.

Let's say I'm in /home directory and I want to move in usr directory which is always in the /home. Here is how I can use **cd** commands -

```
/home $ cd usr
```

```
/home/usr $
```

4. mkdir

Just changing directory is still incomplete. Sometimes you want to create a new folder or subfolder. You can use mkdir command to do that. Just give your folder name after mkdir command in your terminal.

```
~$ mkdir folderName
```

5. cp

copy-and-paste is the important task we need to do to organize our files. Using **cp** will help you to copy-and-paste the file from terminal. First, you determine the file you want to copy and type the destination location to paste the file.

```
$ cp src des
```

Note: If you're copying files into the directory that requires root permission for any new file, then you'll need to use **sudo** command.

6. rm

rm is a command to remove your file or even your directory. You can use -f if the file need root permission to be removed. And also you can use -r to do recursive removal to remove your folder.

```
$ rm myfile.txt
```

7. apt-get

This command differs distro-by-distro. In Debian based Linux distributions, to install, remove and upgrade any package we've *Advanced Packaging Tool* (APT) package manager. The apt-get command will help you installing the software you need to run in your Linux. It is a powerful command-line tool which can perform installation, upgrade, and even removing your software.

In other distributions, such as Fedora, Centos there are different package managers. Fedora used to have **yum** but now it has **dnf**.

```
$ sudo apt-get update
```

```
$ sudo dnf update
```

8. grep

You need to find a file but you don't remember its exact location or the path. grep will help you to solve this problem. You can use the grep command to help finding the file based on given keywords.

```
$ grep user /etc/passwd
```

9. cat

As a user, you often need to view some of text or code from your script. Again, one of the Linux basic commands is cat command. It will show you the text inside your file.

```
$ cat CMakeLists.txt
```

10. poweroff

And the last one is **poweroff**. Sometimes you need to **poweroff** directly from your terminal. This command will do the task. Don't forget to add sudo at the beginning of the command since it needs root permission to execute poweroff.

```
$ sudo poweroff
```

Through this article we are sharing our day-to-day Linux find command experience and its usage in the form of examples. In this article we will show you the most used **35 Find Commands** examples in Linux. We have divided the section into **Five** parts from basic to advance usage of find command.

1. **Part I:** Basic Find Commands for Finding Files with Names
2. **Part II:** Find Files Based on their Permissions
3. **Part III:** Search Files Based On Owners and Groups
4. **Part IV:** Find Files and Directories Based on Date and Time
5. **Part V:** Find Files and Directories Based on Size
6. **Part VI:** [Find Multiple Filenames in Linux](#)

Part I – Basic Find Commands for Finding Files with Names

1. Find Files Using Name in Current Directory

Find all the files whose name is **tecmint.txt** in a current working directory.

```
# find . -name tecmint.txt  
./tecmint.txt
```

2. Find Files Under Home Directory

Find all the files under **/home** directory with name **tecmint.txt**.

```
# find /home -name tecmint.txt  
/home/tecmint.txt
```

3. Find Files Using Name and Ignoring Case

Find all the files whose name is **tecmint.txt** and contains both capital and small letters in **/home** directory.

```
# find /home -iname tecmint.txt  
./tecmint.txt  
./Tecmint.txt
```

4. Find Directories Using Name

Find all directories whose name is **Tecmint** in **/** directory.

```
# find / -type d -name Tecmint  
/Tecmint
```

5. Find PHP Files Using Name

Find all **php** files whose name is **tecmint.php** in a current working directory.

```
# find . -type f -name tecmint.php  
./tecmint.php
```

6. Find all PHP Files in Directory

Find all **php** files in a directory.

```
# find . -type f -name "*.php"  
./tecmint.php  
./login.php  
./index.php
```

Part II – Find Files Based on their Permissions

7. Find Files With 777 Permissions

Find all the files whose permissions are **777**.

```
# find . -type f -perm 0777 -print
```

8. Find Files Without 777 Permissions

Find all the files without permission **777**.

```
# find / -type f ! -perm 777
```

9. Find SGID Files with 644 Permissions

Find all the **SGID bit** files whose permissions set to **644**.

```
# find / -perm 2644
```

10. Find Sticky Bit Files with 551 Permissions

Find all the **Sticky Bit** set files whose permission are **551**.

```
# find / -perm 1551
```

11. Find SUID Files

Find all **SUID** set files.

```
# find / -perm /u=s
```

12. Find SGID Files

Find all **SGID** set files.

```
# find / -perm /g=s
```

13. Find Read Only Files

Find all **Read Only** files.

```
# find / -perm /u=r
```

14. Find Executable Files

Find all **Executable** files.

```
# find / -perm /a=x
```

15. Find Files with 777 Permissions and Chmod to 644

Find all **777** permission files and use **chmod** command to set permissions to **644**.

```
# find / -type f -perm 0777 -print -exec chmod 644 {} \;
```

16. Find Directories with 777 Permissions and Chmod to 755

Find all **777** permission directories and use **chmod** command to set permissions to **755**.

```
# find / -type d -perm 777 -print -exec chmod 755 {} \;
```

17. Find and remove single File

To find a single file called **tecmint.txt** and remove it.

```
# find . -type f -name "tecmint.txt" -exec rm -f {} \;
```

18. Find and remove Multiple File

To find and remove multiple files such as **.mp3** or **.txt**, then use.

```
# find . -type f -name "*.txt" -exec rm -f {} \;
OR
# find . -type f -name "*.mp3" -exec rm -f {} \;
```

19. Find all Empty Files

To find all empty files under certain path.

```
# find /tmp -type f -empty
```

20. Find all Empty Directories

To file all empty directories under certain path.

```
# find /tmp -type d -empty
```

21. File all Hidden Files

To find all hidden files, use below command.

```
# find /tmp -type f -name ".*"
```

Part III – Search Files Based On Owners and Groups

22. Find Single File Based on User

To find all or single file called **tecmint.txt** under / root directory of owner root.

```
# find / -user root -name tecmint.txt
```

23. Find all Files Based on User

To find all files that belongs to user **Tecmint** under **/home** directory.

```
# find /home -user tecmint
```

24. Find all Files Based on Group

To find all files that belongs to group **Developer** under **/home** directory.

```
# find /home -group developer
```

25. Find Particular Files of User

To find all **.txt** files of user **Tecmint** under **/home** directory.

```
# find /home -user tecmint -iname "*.*txt"
```

Part IV – Find Files and Directories Based on Date and Time

26. Find Last 50 Days Modified Files

To find all the files which are modified **50** days back.

```
# find / -mtime 50
```

27. Find Last 50 Days Accessed Files

To find all the files which are accessed **50** days back.

```
# find / -atime 50
```

28. Find Last 50-100 Days Modified Files

To find all the files which are modified more than **50** days back and less than **100** days.

```
# find / -mtime +50 -mtime -100
```

29. Find Changed Files in Last 1 Hour

To find all the files which are changed in last **1 hour**.

```
# find / -cmin -60
```

30. Find Modified Files in Last 1 Hour

To find all the files which are modified in last **1 hour**.

```
# find / -mmin -60
```

31. Find Accessed Files in Last 1 Hour

To find all the files which are accessed in last **1 hour**.

```
# find / -amin -60
```

Part V – Find Files and Directories Based on Size

32. Find 50MB Files

To find all **50MB** files, use.

```
# find / -size 50M
```

33. Find Size between 50MB – 100MB

To find all the files which are greater than **50MB** and less than **100MB**.

```
# find / -size +50M -size -100M
```

34. Find and Delete 100MB Files

To find all **100MB** files and delete them using one single command.

```
# find / -size +100M -exec rm -rf {} \;
```

35. Find Specific Files and Delete

Find all **.mp3** files with more than **10MB** and delete them using one single command.

```
# find / -type f -name *.mp3 -size +10M -exec rm {} \;
```

That's it, We are ending this post here, In our next article we will discuss more about other Linux commands in depth with practical examples. Let us know your opinions on this article using our comment section.

How to Use 'find' Command to Search for Multiple Filenames (Extensions) in Linux

Many times, we are locked in a situation where we have to search for multiple files with different extensions, this has probably happened to several Linux users especially from within the terminal.

There are several Linux utilities that we can use to locate or find files on the file system, but finding multiple filenames or files with different extensions can sometimes prove tricky and requires specific commands.

```
tecmint@tecmint ~/testing $ find . -type f \( -name "*.txt" -o -name "*.sh" -o -name "*.c" \)
./emails.txt
./script-1.sh
./header.c
./examples.txt
./script.sh
./expenses.txt
```

Find Multiple Filenames (File Extensions) Using 'find' Command in Linux

Find Multiple File Names in Linux

One of the many utilities for locating files on a Linux file system is the **find** utility and in this how-to guide, we shall walk through a few examples of using **find** to help us locate multiple filenames at once.

Before we dive into the actual commands, let us look at a brief introduction to the Linux `find` utility.

The simplest and general syntax of the `find` utility is as follows:

```
# find directory options [ expression ]
```

Let us proceed to look at some examples of **find** command in Linux.

1. Assuming that you want to find all files in the current directory with `.sh` and `.txt` file extensions, you can do this by running the command below:

```
# find . -type f \(-name "*.sh" -o -name "*.txt" \)
aaronkilik@tecMint ~/bin $ find . -type f \(-name "*.sh" -o -name "*.txt" \)
./examples.txt
./script.sh
./test.sh
./list.txt
aaronkilik@tecMint ~/bin $
```

Find `.sh` and `.txt` Extension Files in Linux

Interpretation of the command above:

1. `.` means the current directory
2. `-type` option is used to specify file type and here, we are searching for regular files as represented by `f`
3. `-name` option is used to specify a search pattern in this case, the file extensions
4. `-o` means “OR”

It is recommended that you enclose the file extensions in a bracket, and also use the `\` (**back slash**) escape character as in the command.

2. To find three filenames with `.sh`, `.txt` and `.c` extensions, issues the command below:

```
# find . -type f \(-name "*.sh" -o -name "*.txt" -o -name "*.c" \)
aaronkilik@tecMint ~/bin $ find . -type f \(-name "*.sh" -o -name "*.txt" -o -name "*.c" \)
./examples.txt
./script.sh
./test.sh
./list.txt
./file.c
./header.c
./lost.c
aaronkilik@tecMint ~/bin $
```

Find Multiple File Extensions in Linux

3. Here is another example where we search for files with `.png`, `.jpg`, `.deb` and `.pdf` extensions:

```
# find /home/aaronkilik/Documents/ -type f \(\ -name "*.png" -o -name "*.jpg"
-o -name "*.deb" -o -name ".pdf" \)
```

```
aaronkilik@tecMint ~ $ find /home/aaronkilik/Documents/ -type f \(\ -name "*.png" -o -name "*.jpg"
-o -name "*.deb" -o -name ".pdf" \
/home/aaronkilik/Documents/sudo.png
/home/aaronkilik/Documents/festival3.jpg
/home/aaronkilik/Documents/true.jpg
/home/aaronkilik/Documents/keyboard.jpg
/home/aaronkilik/Documents/programmers.jpg
/home/aaronkilik/Documents/own_business.jpg
/home/aaronkilik/Documents/HIM.jpg
/home/aaronkilik/Documents/life.jpg
/home/aaronkilik/Documents/cabling.png
/home/aaronkilik/Documents/Tecmint-News/ServerMania-Cloud-Hosting.jpg
/home/aaronkilik/Documents/f8M93fM.jpg
/home/aaronkilik/Documents/linux.jpg
/home/aaronkilik/Documents/children.jpg
/home/aaronkilik/Documents/dp.jpg
/home/aaronkilik/Documents/festival2.jpg
/home/aaronkilik/Documents/keyboard-shortcuts.jpg
/home/aaronkilik/Documents/festival.jpg
/home/aaronkilik/Documents/Tecmint.com/How to Upgrade To Linux Mint 18 /check-mintupgrade1.png
/home/aaronkilik/Documents/Tecmint.com/How to Upgrade To Linux Mint 18 /download-complete.png
/home/aaronkilik/Documents/Tecmint.com/How to Upgrade To Linux Mint 18 /upgrade3.png
/home/aaronkilik/Documents/Tecmint.com/How to Upgrade To Linux Mint 18 /download-mintupgrade-pac
ges.png
/home/aaronkilik/Documents/Tecmint.com/How to Upgrade To Linux Mint 18 /check-unlimited-scrolling.
png
/home/aaronkilik/Documents/Tecmint.com/How to Upgrade To Linux Mint 18 /update-manager.png
/home/aaronkilik/Documents/Tecmint.com/How to Upgrade To Linux Mint 18 /package-configuration.png
```

Find More than 3 File Extensions in Linux

When you critically observe all the commands above, the little trick is using the `-o` option in the **find** command, it enables you to add more filenames to the search array, and also knowing the filenames or file extensions you are searching for.

Conclusion

In this guide, we covered a simple yet helpful **find utility** trick to enable us find multiple filenames by issuing a single command. To understand and use find for many other vital command line operations, you can read our article below.

25 simple examples of Linux find command

Linux find command

The Linux **find command** is a very useful and handy command to search for files from the command line. It can be used to find files based on various search criterias like permissions, user ownership, modification date/time, size etc. In this post we shall learn to use the find command along with various options that it supports.

The find command is available on most linux distros by default so you do not have to install any package. The find command is an essential one to learn, if you want to get super productive with the command line on linux.

The basic syntax of the find command looks like this

```
$ find location comparison-criteria search-term
```

Basic examples

1. List all files in current and sub directories

This command lists out all the files in the current directory as well as the subdirectories in the current directory.

```
$ find  
. ./abc.txt  
. ./subdir  
. ./subdir/how.php  
. ./cool.php
```

The command is same as the following

```
$ find .  
$ find . -print
```

2. Search specific directory or path

The following command will look for files in the test directory in the current directory. Lists out all files by default.

```
$ find ./test  
. ./test  
. ./test/abc.txt  
. ./test/subdir  
. ./test/subdir/how.php  
. ./test/cool.php
```

The following command searches for files by their name.

```
$ find ./test -name "abc.txt"  
./test/abc.txt
```

We can also use wildcards

```
$ find ./test -name "*.php"  
./test/subdir/how.php  
./test/cool.php
```

Note that all sub directories are searched recursively. So this is a very powerful way to find all files of a given extension.

Trying to search the "/" directory which is the root, would search the entire file system including mounted devices and network storage devices. So be careful. Of course you can press Ctrl + c anytime to stop the command.

When specifying the directory ("./test" in this example), its fine to omit the trailing slash. However, if the directory is actually a symlink to some other location then you MUST specify the trailing slash for it to work properly (find ./test/ ...)

Ignore the case

It is often useful to ignore the case when searching for file names. To ignore the case, just use the "iname" option instead of the "name" option.

```
$ find ./test -iname "*.Php"  
./test/subdir/how.php  
./test/cool.php
```

Its always better to wrap the search term (name parameter) in double or single quotes. Not doing so will seem to work sometimes and give strange results at other times.

3. Limit depth of directory traversal

The find command by default travels down the entire directory tree recursively, which is time and resource consuming. However the depth of directory traversal can be specified. For example we don't want to go more than 2 or 3 levels down in the sub directories. This is done using the maxdepth option.

```
$ find ./test -maxdepth 2 -name "*.php"  
./test/subdir/how.php  
./test/cool.php
```



```
$ find ./test -maxdepth 1 -name *.php  
./test/cool.php
```

The second example uses maxdepth of 1, which means it will not go lower than 1 level deep, either only in the current directory.

This is very useful when we want to do a limited search only in the current directory or max 1 level deep sub directories and not the entire directory tree which would take more time.

Just like maxdepth there is an option called mindepth which does what the name suggests, that is, it will go atleast N level deep before searching for the files.

4. Invert match

It is also possible to search for files that do no match a given name or pattern. This is helpful when we know which files to exclude from the search.

```
$ find ./test -not -name "*.php"  
./test  
./test/abc.txt  
./test/subdir
```

So in the above example we found all files that do not have the extension of php, either non-php files. The find command also supports the exclamation mark inplace of not.

```
find ./test ! -name "*.php"
```

5. Combine multiple search criterias

It is possible to use multiple criterias when specifying name and inverting. For example

```
$ find ./test -name 'abc*' ! -name '*.php'  
./test/abc.txt  
./test/abc
```

The above find command looks for files that begin with abc in their names and do not have a php extension. This is an example of how powerful search expressions can be build with the find command.

OR operator

When using multiple name criterias, the find command would combine them with AND operator, which means that only those files which satisfy all criterias will be matched. However if we need to perform an OR based matching then the find command has the "o" switch.

```
$ find -name '*.php' -o -name '*.txt'  
./abc.txt  
./subdir/how.php  
./abc.php  
./cool.php
```

The above command search for files ending in either the php extension or the txt extension.

6. Search only files or only directories

Sometimes we want to find only files or only directories with a given name. Find can do this easily as well.

```
$ find ./test -name abc*
./test/abc.txt
./test/abc

Only files

$ find ./test -type f -name "abc*"
./test/abc.txt

Only directories

$ find ./test -type d -name "abc*"
./test/abc
```

Quite useful and handy!

7. Search multiple directories together

So lets say you want to search inside 2 separate directories. Again, the command is very simple

```
$ find ./test ./dir2 -type f -name "abc*"
./test/abc.txt
./dir2/abcdefg.txt
```

Check, that it listed files from 2 separate directories.

8. Find hidden files

Hidden files on linux begin with a period. So its easy to mention that in the name criteria and list all hidden files.

```
$ find ~ -type f -name ".*"
```

Find files based on permissions

9. Find files with certain permissions

The find command can be used to find files with a specific permission using the "perm" option. The following command searches for files with the permission 0664

```
$ find . -type f -perm 0664
./abc.txt
./subdir/how.php
./abc.php
./cool.php
```

This can be useful to find files with wrong permissions which can lead to security issues. Inversion can also be applied to permission checking.

```
$ find . -type f ! -perm 0777  
./abc.txt  
./subdir/how.php  
./abc.php  
./cool.php
```

10. Find files with sgid/suid bits set

The "perm" option of find command accepts the same mode string like chmod. The following command finds all files with permission 644 and sgid bit set.

```
# find / -perm 2644
```

Similarly use 1664 for sticky bit. The perm option also supports using an alternative syntax instead of octal numbers.

```
$ find / -maxdepth 2 -perm /u=s 2>/dev/null  
/bin/mount  
/bin/su  
/bin/ping6  
/bin/fusermount  
/bin/ping  
/bin/umount  
/sbin/mount_ecryptfs_private
```

Note that the "2>/dev/null" removes those entries that have an error of "Permission Denied"

11. Find readonly files

Find all Read Only files.

```
$ find /etc -maxdepth 1 -perm /u=r  
/etc  
/etc/thunderbird  
/etc/brltty  
/etc/dkms  
/etc/phpmyadmin  
... output truncated ...
```

12. Find executable files

The following command will find executable files

```
$ find /bin -maxdepth 2 -perm /a=x  
/bin  
/bin/preseed_command  
/bin/mount  
/bin/zfgrep
```

```
/bin/tempfile  
... output truncated ...
```

Search Files Based On Owners and Groups

13. Find files belonging to particular user

To find all or single file called tecmint.txt under /root directory of owner root.

```
$ find . -user bob  
.  
.abc.txt  
.abc  
.subdir  
.subdir/how.php  
.abc.php
```

We could also specify the name of the file or any name related criteria along with user criteria

```
$ find . -user bob -name '*.php'
```

Its very easy to see, how we can build up criteria after criteria to narrow down our search for matching files.

14. Search files belonging to group

Find all files that belong to a particular group.

```
# find /var/www -group developer
```

Did you know you could search your home directory by using the ~ symbol ?

```
$ find ~ -name "hidden.php"
```

Easy!!

Search file and directories based on modification date and time

Another great search criteria that the find command supports is modification and accessed date/times. This is very handy when we want to find out which files were modified as a certain time or date range. Lets take a few examples

15. Find files modified N days back

To find all the files which are modified 50 days back.

```
# find / -mtime 50
```

16. Find files accessed in last N days

Find all files that were accessed in the last 50 days.

```
# find / -atime 50
```

17. Find files modified in a range of days

Find all files that were modified between 50 to 100 days ago.

```
# find / -mtime +50 -mtime -100
```

18. Find files changed in last N minutes.

Find files modified within the last 1 hour.

```
$ find /home/bob -cmin -60
```

19. Files modified in last hour

To find all the files which are modified in last 1 hour.

```
# find / -mmin -60
```

20. Find Accessed Files in Last 1 Hour

To find all the files which are accessed in last 1 hour.

```
# find / -amin -60
```

Search files and directories based on size

21. Find files of given size

To find all 50MB files, use.

```
# find / -size 50M
```

22. Find files in a size range

To find all the files which are greater than 50MB and less than 100MB.

```
$ find / -size +50M -size -100M
```

23. Find largest and smallest files

The find command when used in combination with the ls and sort command can be used to list out the largest files.

The following command will display the 5 largest file in the current directory and its subdirectory. This may take a while to execute depending on the total number of files the command has to process.

```
$ find . -type f -exec ls -s {} \; | sort -n -r | head -5
```

Similary when sorted in ascending order, it would show the smallest files first

```
$ find . -type f -exec ls -s {} \; | sort -n | head -5
```

24. Find empty files and directories

The following command uses the "empty" option of the find command, which finds all files that are empty.

```
# find /tmp -type f -empty
```

To file all empty directories use the type "d".

```
$ find ~/ -type d -empty
```

Really very simple and easy

Some advanced operations

The find command not only finds files based on a certain criteria, it can also act upon those files using any linux command. For example, we might want to delete some files.

Here are some quick examples

25. List out the found files

Lets say we found files using find command, and now want to list them out as the ls command would have done. This is very easy.

```
$ find . -exec ls -ld {} \;
drwxrwxr-x 4 enlightened enlightened 4096 Aug 11 19:01 .
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:25 ./abc.txt
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11 16:48 ./abc
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11 16:26 ./subdir
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:26 ./subdir/how.php
-rw-rw-r-- 1 enlightened enlightened 29 Aug 11 19:13 ./abc.php
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:25 ./cool.php
```

26. Delete all matching files or directories

The following command will remove all text files in the tmp directory.

```
$ find /tmp -type f -name "*.txt" -exec rm -f {} \;
```

The same operating can be carried out with directories, just put type d, instead of type f.

Lets take another example where we want to delete files larger than 100MB

```
$ find /home/bob/dir -type f -name *.log -size +10M -exec rm -f {} \;
```

Summary

So that was a quick tutorial on the **linux find command**. The find command is one of the most essential commands on the linux terminal, that enables searching of files very easy. Its a must of all system administrators. So learn it up. Have any questions ? Leave a comment below.

16 commands to check hardware information on Linux

Hardware information

Like for every thing, there are plenty of commands to check information about the hardware of your linux system. Some commands report only specific hardware components like cpu or memory while the rest cover multiple hardware units.

This post takes a quick look at some of the most commonly used commands to check information and configuration details about various hardware peripherals and devices. The list includes lscpu, hwinfo, lshw, dmidecode, lspci etc.

[1. lscpu](#)

The lscpu command reports information about the cpu and processing units. It does not have any further options or functionality.

```
$ lscpu
Architecture:           x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):              1
NUMA node(s):           1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  23
Stepping:                10
CPU MHz:                1998.000
BogoMIPS:               5302.48
Virtualization:         VT-x
L1d cache:              32K
L1i cache:              32K
L2 cache:                2048K
NUMA node0 CPU(s):      0-3
```

[2. lshw - List Hardware](#)

A general purpose utility, that reports detailed and brief information about multiple different hardware units such as cpu, memory, disk, usb controllers, network adapters etc. Lshw extracts the information from different /proc files.

```
$ sudo lshw -short
```

H/W path	Device	Class	Description
/0		system	()
/0/0		bus	DG35EC
@ 2.66GHz		processor	Intel(R) Core(TM)2 Quad CPU Q8400
/0/0/1		memory	2MiB L2 cache
/0/0/3		memory	32KiB L1 cache
/0/2		memory	32KiB L1 cache
/0/4		memory	64KiB BIOS
/0/14		memory	8GiB System Memory
/0/14/0		memory	2GiB DIMM DDR2 Synchronous 667 MHz (1.5 ns)
/0/14/1		memory	2GiB DIMM DDR2 Synchronous 667 MHz (1.5 ns)
/0/14/2		memory	2GiB DIMM DDR2 Synchronous 667 MHz (1.5 ns)
/0/14/3		memory	2GiB DIMM DDR2 Synchronous 667 MHz (1.5 ns)
/0/100		bridge	82G35 Express DRAM Controller
/0/100/2		display	82G35 Express Integrated Graphics
Controller		display	82G35 Express Integrated Graphics
/0/100/2.1			
Controller			
/0/100/19	eth0	network	82566DC Gigabit Network Connection
/0/100/1a		bus	82801H (ICH8 Family) USB UHCI
Controller #4		bus	82801H (ICH8 Family) USB UHCI
/0/100/1a.1		bus	82801H (ICH8 Family) USB UHCI
Controller #5		bus	82801H (ICH8 Family) USB2 EHCI
/0/100/1a.7		bus	82801H (ICH8 Family) USB2 EHCI
Controller #2		multimedia	82801H (ICH8 Family) HD Audio
/0/100/1b		bridge	82801H (ICH8 Family) PCI Express Port
Controller		bridge	82801H (ICH8 Family) PCI Express Port
/0/100/1c		bridge	82801H (ICH8 Family) PCI Express Port
1		bridge	82801H (ICH8 Family) PCI Express Port
/0/100/1c.1		bridge	82801H (ICH8 Family) PCI Express Port
2		bridge	82801H (ICH8 Family) PCI Express Port
/0/100/1c.2		bridge	82801H (ICH8 Family) PCI Express Port
3		bridge	82801H (ICH8 Family) PCI Express Port
/0/100/1c.2/0		storage	JMB368 IDE controller
/0/100/1d		bus	82801H (ICH8 Family) USB UHCI
Controller #1		bus	82801H (ICH8 Family) USB UHCI
/0/100/1d.1		bus	82801H (ICH8 Family) USB UHCI
Controller #2		bus	82801H (ICH8 Family) USB UHCI
/0/100/1d.2		bus	82801H (ICH8 Family) USB UHCI
Controller #3		bus	82801H (ICH8 Family) USB2 EHCI
/0/100/1d.7		bus	82801H (ICH8 Family) USB2 EHCI
Controller #1		bridge	82801 PCI Bridge
/0/100/1e		bus	FW322/323 [TrueFire] 1394a Controller
/0/100/1e/5		bridge	82801HB/HR (ICH8/R) LPC Interface
/0/100/1f		bridge	
Controller		storage	82801H (ICH8 Family) 4 port SATA
/0/100/1f.2		bus	82801H (ICH8 Family) SMBus Controller
Controller [IDE mode]		storage	82801HR/HO/HH (ICH8R/DO/DH) 2 port
/0/100/1f.3			
/0/100/1f.5			
SATA Controller [IDE m			

```

/0/1           scsi3      storage
/0/1/0.0.0    /dev/sda    disk      500GB ST3500418AS
/0/1/0.0.0/1  /dev/sda1   volume   70GiB Windows NTFS volume
/0/1/0.0.0/2  /dev/sda2   volume   395GiB Extended partition
/0/1/0.0.0/2/5 /dev/sda5   volume   97GiB HPFS/NTFS partition
/0/1/0.0.0/2/6 /dev/sda6   volume   97GiB Linux filesystem partition
/0/1/0.0.0/2/7 /dev/sda7   volume   1952MiB Linux swap / Solaris
partition
/0/1/0.0.0/2/8 /dev/sda8   volume   198GiB Linux filesystem partition
/0/3           scsi4      storage
/0/3/0.0.0    /dev/cdrom  disk      DVD RW DRU-190A

```

Check out the following post to learn more about lshw

[Get hardware information on Linux with lshw command](#)

3. hwinfo - Hardware Information

Hwinfo is another general purpose hardware probing utility that can report detailed and brief information about multiple different hardware components, and more than what lshw can report.

```

$ hwinfo --short
cpu:
MHz                         Intel (R) Core(TM)2 Quad CPU        Q8400 @ 2.66GHz, 2000
MHz                         Intel (R) Core(TM)2 Quad CPU        Q8400 @ 2.66GHz, 2000
MHz                         Intel (R) Core(TM)2 Quad CPU        Q8400 @ 2.66GHz, 2666
MHz                         Intel (R) Core(TM)2 Quad CPU        Q8400 @ 2.66GHz, 2666
MHz
keyboard:
/dev/input/event2          AT Translated Set 2 keyboard
mouse:
/dev/input/mice             Microsoft Basic Optical Mouse v2.0
graphics card:
Intel 965G-1
Intel 82G35 Express Integrated Graphics Controller
sound:
Intel 82801H (ICH8 Family) HD Audio Controller
storage:
Intel 82801H (ICH8 Family) 4 port SATA IDE Controller
Intel 82801H (ICH8 Family) 2 port SATA IDE Controller
JMicron JMB368 IDE controller
network:
eth0                        Intel 82566DC Gigabit Network Connection
network interface:
eth0                        Ethernet network interface
lo                           Loopback network interface
disk:
/dev/sda                     ST3500418AS
partition:
/dev/sda1                    Partition
/dev/sda2                    Partition
/dev/sda5                    Partition

```

```

/dev/sda6          Partition
/dev/sda7          Partition
/dev/sda8          Partition
cdrom:
/dev/sr0           SONY DVD RW DRU-190A
usb controller:
Intel 82801H (ICH8 Family) USB UHCI Controller #4
Intel 82801H (ICH8 Family) USB UHCI Controller #5
Intel 82801H (ICH8 Family) USB2 EHCI Controller #2
Intel 82801H (ICH8 Family) USB UHCI Controller #1
Intel 82801H (ICH8 Family) USB UHCI Controller #2
Intel 82801H (ICH8 Family) USB UHCI Controller #3
Intel 82801H (ICH8 Family) USB2 EHCI Controller #1
bios:
BIOS
... TRUNCATED ...

```

Check out our previous post on hwinfo

[Check hardware information on Linux with hwinfo command](#)

4. *lspci - List PCI*

The lspci command lists out all the pci buses and details about the devices connected to them. The vga adapter, graphics card, network adapter, usb ports, sata controllers, etc all fall under this category.

```
$ lspci
00:00.0 Host bridge: Intel Corporation 82G35 Express DRAM Controller (rev 03)
00:02.0 VGA compatible controller: Intel Corporation 82G35 Express Integrated
Graphics Controller (rev 03)
00:02.1 Display controller: Intel Corporation 82G35 Express Integrated
Graphics Controller (rev 03)
00:19.0 Ethernet controller: Intel Corporation 82566DC Gigabit Network
Connection (rev 02)
00:1a.0 USB controller: Intel Corporation 82801H (ICH8 Family) USB UHCI
Controller #4 (rev 02)
00:1a.1 USB controller: Intel Corporation 82801H (ICH8 Family) USB UHCI
Controller #5 (rev 02)
00:1a.7 USB controller: Intel Corporation 82801H (ICH8 Family) USB2 EHCI
Controller #2 (rev 02)
00:1b.0 Audio device: Intel Corporation 82801H (ICH8 Family) HD Audio
Controller (rev 02)
00:1c.0 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 1
(rev 02)
00:1c.1 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 2
(rev 02)
00:1c.2 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 3
(rev 02)
00:1d.0 USB controller: Intel Corporation 82801H (ICH8 Family) USB UHCI
Controller #1 (rev 02)
00:1d.1 USB controller: Intel Corporation 82801H (ICH8 Family) USB UHCI
Controller #2 (rev 02)
```

```
00:1d.2 USB controller: Intel Corporation 82801H (ICH8 Family) USB UHCI  
Controller #3 (rev 02)  
00:1d.7 USB controller: Intel Corporation 82801H (ICH8 Family) USB2 EHCI  
Controller #1 (rev 02)  
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev f2)  
00:1f.0 ISA bridge: Intel Corporation 82801HB/HR (ICH8/R) LPC Interface  
Controller (rev 02)  
00:1f.2 IDE interface: Intel Corporation 82801H (ICH8 Family) 4 port SATA  
Controller [IDE mode] (rev 02)  
00:1f.3 SMBus: Intel Corporation 82801H (ICH8 Family) SMBus Controller (rev  
02)  
00:1f.5 IDE interface: Intel Corporation 82801HR/HO/HH (ICH8R/DO/DH) 2 port  
SATA Controller [IDE mode] (rev 02)  
03:00.0 IDE interface: JMicron Technology Corp. JMB368 IDE controller  
04:05.0 FireWire (IEEE 1394): LSI Corporation FW322/323 [TrueFire] 1394a  
Controller (rev 70)
```

Filter out specific device information with grep.

```
$ lspci -v | grep "VGA" -A 12
```

[5. lsscsi - List scsi devices](#)

Lists out the scsi/sata devices like hard drives and optical drives.

```
$ lsscsi  
[3:0:0:0]      disk      ATA        ST3500418AS      CC38    /dev/sda  
[4:0:0:0]      cd/dvd   SONY       DVD RW DRU-190A  1.63    /dev/sr0
```

[6. lsusb - List usb buses and device details](#)

This command shows the USB controllers and details about devices connected to them. By default brief information is printed. Use the verbose option "-v" to print detailed information about each usb port

```
$ lsusb  
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
Bus 005 Device 002: ID 045e:00cb Microsoft Corp. Basic Optical Mouse v2.0  
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

On the above system, 1 usb port is being used by the mouse.

[7. Inxi](#)

Inxi is a 10K line mega bash script that fetches hardware details from multiple different sources and commands on the system, and generates a beautiful looking report that non technical users can read easily.

```
$ inxi -Fx
```

The screenshot shows a terminal window titled "enlightened : bash - Konsole". The window contains the output of the "inxi -Fx" command, which provides detailed hardware information. The output includes:

- System:** Host: enlightened Kernel: 3.11.0-12-generic x86_64 (64 bit, gcc: 4.8.1)
- Machine:** Desktop: KDE 4.11.5 (Qt 4.8.4) Distro: Ubuntu 13.10
- CPU:** Model: Intel Core2 Quad CPU Q8400 (-MCP-) cache: 2048 KB flags: (lm nx sse sse2 sse3 sse4_1 ssse3 vmx) bmips: 21212.2 Clock Speeds: 1: 1998.00 MHz 2: 2664.00 MHz 3: 1998.00 MHz 4: 2664.00 MHz
- Graphics:** Card: Intel 82G35 Express Integrated Graphics Controller bus-ID: 00:02.0 X.Org: 1.14.5 drivers: intel (unloaded: fbdev,vesa) Resolution: 1360x768@60.0hz GLX Renderer: Mesa DRI Intel 965G GLX Version: 2.1 Mesa 9.2.1 Direct Rendering: Yes
- Audio:** Card: Intel 82801H (ICH8 Family) HD Audio Controller driver: snd_hda_intel bus-ID: 00:1b.0 Sound: Advanced Linux Sound Architecture ver: k3.11.0-12-generic
- Network:** Card: Intel 82566DC Gigabit Network Connection driver: e1000e ver: 2.3.2-k port: 20c0 bus-ID: 00:19.0 IF: eth0 state: up speed: 100 Mbps duplex: full mac: 00:1c:c0:f8:79:ee
- Drives:** HDD Total Size: 500.1GB (41.1% used) 1: id: /dev/sda model: ST3500418AS size: 500.1GB
- Partition:** ID: / size: 97G used: 25G (28%) fs: ext4 ID: swap-1 size: 2.05GB

[8. lsblk - List block devices](#)

List out information all block devices, which are the hard drive partitions and other storage devices like optical drives and flash drives

```
$ lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	465.8G	0	disk	
└─sda1	8:1	0	70G	0	part	
└─sda2	8:2	0	1K	0	part	
└─sda5	8:5	0	97.7G	0	part	/media/4668484A68483B47
└─sda6	8:6	0	97.7G	0	part	/
└─sda7	8:7	0	1.9G	0	part	[SWAP]
└─sda8	8:8	0	198.5G	0	part	/media/13f35f59-f023-4d98-b06f-9dfaebefc6c1
sr0	11:0	1	1024M	0	rom	

[9. df - disk space of file systems](#)

Reports various partitions, their mount points and the used and available space on each.

```
$ df -H
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda6        104G   26G   73G  26% /
none            4.1k     0  4.1k  0% /sys/fs/cgroup
udev            4.2G   4.1k   4.2G  1% /dev
tmpfs           837M  1.6M  835M  1% /run
none            5.3M     0  5.3M  0% /run/lock
none            4.2G   13M   4.2G  1% /run/shm
none            105M   21k   105M  1% /run/user
/dev/sda8        210G  149G   51G  75% /media/13f35f59-f023-4d98-b06f-
9dfaebefdf6c1
/dev/sda5        105G   31G   75G  30% /media/4668484A68483B47
```

10. Pydf - Python df

An improved df version written in python, that displays colored output that looks better than df

```
$ pydf
Filesystem Size Used Avail Use% Mounted on
/dev/sda6 96G 23G 68G 24.4 [#. ....] /
/dev/sda8 195G 138G 47G 70.6 [## ## ..] /media/13f35f59-f023-4d98-b06f-
9dfaebefc6c1
/dev/sda5 98G 28G 69G 29.2 [## ....] /media/4668484A68483B47
```

11. *fdisk*

Fdisk is a utility to modify partitions on hard drives, and can be used to list out the partition information as well.

```
$ sudo fdisk -l
```

```
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x30093008
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	63	146801969	73400953+	7	HPFS/NTFS/exFAT
/dev/sda2		146802031	976771071	414984520+	f	W95 Ext'd (LBA)
/dev/sda5		146802033	351614654	102406311	7	HPFS/NTFS/exFAT
/dev/sda6		351614718	556427339	102406311	83	Linux
/dev/sda7		556429312	560427007	1998848	82	Linux swap / Solaris
/dev/sda8		560429056	976771071	208171008	83	Linux

12. mount

The mount is used to mount/unmount and view mounted file systems.

```

sysfs      on /sys                           type sysfs
(rw,noexec,nosuid,nodev)
none       on /sys/fs/cgroup                  type tmpfs
(rw)
none       on /sys/fs/fuse/connections        type fusectl
(rw)
none       on /sys/kernel/debug               type debugfs
(rw)
none       on /sys/kernel/security            type
securityfs (rw)
udev       on /dev                           type devtmpfs
(rw,mode=0755)
devpts     on /dev/pts                      type devpts
(rw,noexec,nosuid,gid=5,mode=0620)
tmpfs      on /run                          type tmpfs
(rw,noexec,nosuid,size=10%,mode=0755)
none       on /run/lock                     type tmpfs
(rw,noexec,nosuid,nodev,size=5242880)
none       on /run/shm                      type tmpfs
(rw,nosuid,nodev)
none       on /run/user                     type tmpfs
(rw,noexec,nosuid,nodev,size=104857600,mode=0755)
none       on /sys/fs/pstore                type pstore
(rw)
/dev/sda8   on /media/13f35f59-f023-4d98-b06f-9dfaebefc1 type ext4
(rw,nosuid,nodev,errorss=remount-ro)
/dev/sda5   on /media/4668484A68483B47      type fuseblk
(rw,nosuid,nodev,allow_other,blksize=4096)
binfmt_misc on /proc/sys/fs/binfmt_misc    type
binfmt_misc (rw,noexec,nosuid,nodev)
systemd    on /sys/fs/cgroup/systemd       type cgroup
(rw,noexec,nosuid,nodev,none,name=systemd)
gvfsd-fuse on /run/user/1000/gvfs          type
fuse.gvfsd-fuse (rw,nosuid,nodev,user=enlightened)

```

Again, use grep to filter out only those file systems that you want to see

```
$ mount | column -t | grep ext
```

[13. free - Check RAM](#)

Check the amount of used, free and total amount of RAM on system with the free command.

```
$ free -m
              total        used         free        shared       buffers       cached
Mem:       7975        5865        2110           0          24        622
-/+ buffers/cache:  5218        2757
Swap:      1951         921        1030
```

[14. dmidecode](#)

The dmidecode command is different from all other commands. It extracts hardware information by reading data from the [SMBOIS data structures](#) (also called DMI tables).

```
# display information about the processor/cpu
$ sudo dmidecode -t processor
```

```
# memory/ram information  
$ sudo dmidecode -t memory  
  
# bios details  
$ sudo dmidecode -t bios
```

Check out the man page for more details.

[15. /proc files](#)

Many of the virtual files in the /proc directory contain information about hardware and configurations. Here are some of them

CPU/Memory information

```
# cpu information  
$ cat /proc/cpuinfo  
  
# memory information  
$ cat /proc/meminfo
```

Linux/kernel information

```
$ cat /proc/version  
Linux version 3.11.0-12-generic (buildd@allspice) (gcc version 4.8.1 (Ubuntu/  
Linaro 4.8.1-10ubuntu7) ) #19-Ubuntu SMP Wed Oct 9 16:20:46 UTC 2013
```

SCSI/Sata devices

```
$ cat /proc/scsi/scsi  
Attached devices:  
Host: scsi3 Channel: 00 Id: 00 Lun: 00  
      Vendor: ATA           Model: ST3500418AS          Rev: CC38  
      Type:  Direct-Access                         ANSI  SCSI revision: 05  
Host: scsi4 Channel: 00 Id: 00 Lun: 00  
      Vendor: SONY          Model: DVD RW DRU-190A        Rev: 1.63  
      Type:  CD-ROM                           ANSI  SCSI revision: 05
```

Partitions

```
$ cat /proc/partitions  
major minor  #blocks  name  
  
 8       0   488386584 sda  
 8       1    73400953 sda1  
 8       2         1 sda2  
 8       5  102406311 sda5  
 8       6  102406311 sda6  
 8       7   1998848 sda7  
 8       8  208171008 sda8  
11       0   1048575 sr0
```

16. hdparm

The hdparm command gets information about sata devices like hard disks.

```
$ sudo hdparm -i /dev/sda

/dev/sda:

Model=ST3500418AS, FwRev=CC38, SerialNo=9VMJXV1N
Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs RotSpdTol>.5% }
RawCHS=16383/16/63, TrkSize=0, SectSize=0, ECCbytes=4
BuffType=unknown, BuffSize=16384kB, MaxMultSect=16, MultSect=16
CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAssects=976773168
IORDY=on/off, tPIO={min:120,w/IORDY:120}, tDMA={min:120,rec:120}
PIO modes: pio0 pio1 pio2 pio3 pio4
DMA modes: mdma0 mdma1 mdma2
UDMA modes: udma0 udma1 udma2 udma3 udma4 udma5 *udma6
AdvancedPM=no WriteCache=enabled
Drive conforms to: unknown: ATA/ATAPI-4,5,6,7

* signifies the current active mode
```

Summary

Each of the command has a slightly different method of extracting information, and you may need to try more than one of them, while looking for specific hardware details. However they are available across most linux distros, and can be easily installed from the default repositories.

On the desktop there are gui tools, for those who do not want to memorise and type commands. Hardinfo, I-nex are some of the popular ones that provide detailed information about multiple different hardware components.

8 commands to check cpu information on Linux

CPU hardware information

The cpu information includes details about the processor, like the architecture, vendor name, model, number of cores, speed of each core etc. There are quite a few commands on linux to get those details about the cpu hardware, and here is a brief about some of the commands.

1. /proc/cpuinfo

The /proc/cpuinfo file contains details about individual cpu cores. Output its contents with less or cat.

```
$ less /proc/cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 23
model name    : Intel(R) Core(TM)2 Quad CPU      Q8400 @ 2.66GHz
stepping       : 10
microcode     : 0xa07
cpu MHz       : 1998.000
cache size    : 2048 KB
physical id   : 0
siblings       : 4
core id        : 0
cpu cores     : 4
apicid         : 0
initial apicid: 0
fpu            : yes
fpu_exception  : yes
cpuid level   : 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cm
ov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm co
nstant_tsc arch_perfmon pebs bts rep_good nopl aperf mperf pni dtes64 monitor
ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dtherm tpr_shado
w vnmi flexpriority
bogomips      : 5303.14
clflush size  : 64
cache_alignment: 64
address sizes  : 36 bits physical, 48 bits virtual
power management:
```

Every processor or core is listed separately the various details about speed, cache size and model name are included in the description.

To count the number of processing units use grep with wc

```
$ cat /proc/cpuinfo | grep processor | wc -l  
4
```

The number of processors shown by /proc/cpuinfo might not be the actual number of cores on the processor. For example a processor with 2 cores and hyperthreading would be reported as a processor with 4 cores.

To get the actual number of cores, check the core id for unique values

```
$ cat /proc/cpuinfo | grep 'core id'  
core id      : 0  
core id      : 2  
core id      : 1  
core id      : 3
```

So there are 4 different core ids. This indicates that there are 4 actual cores.

2. lscpu

lscpu is a small and quick command that does not need any options. It would simply print the cpu hardware details in a user-friendly format.

```
$ lscpu  
Architecture:          x86_64  
CPU op-mode(s):       32-bit, 64-bit  
Byte Order:            Little Endian  
CPU(s):                4  
On-line CPU(s) list:  0-3  
Thread(s) per core:   1  
Core(s) per socket:   4  
Socket(s):             1  
NUMA node(s):          1  
Vendor ID:             GenuineIntel  
CPU family:            6  
Model:                 23  
Stepping:               10  
CPU MHz:                1998.000  
BogoMIPS:               5303.14  
Virtualization:        VT-x  
L1d cache:              32K  
L1i cache:              32K  
L2 cache:                2048K  
NUMA node0 CPU(s):     0-3
```

3. hardinfo

Hardinfo is a gtk based gui tool that generates reports about various hardware components. But it can also run from the command line only if there is no gui display available.

```
$ hardinfo | less
```

It would produce a large report about many hardware parts, by reading files from the /proc directory. The cpu information is towards the beginning of the report. The report can also be written to a text file.

Hardinfo also performs a few benchmark tests taking a few minutes before the report is displayed.

4. lshw

The lshw command can display limited information about the cpu. lshw by default shows information about various hardware parts, and the '-class' option can be used to pickup information about a specific hardware part.

```
$ sudo lshw -class processor
*-cpu
      description: CPU
      product: Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz
      vendor: Intel Corp.
      physical id: 0
      bus info: cpu@0
      version: Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz
      slot: LGA 775
      size: 1998MHz
      capacity: 4GHz
      width: 64 bits
      clock: 333MHz
      capabilities: fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx x86-64 constant_tsc arch_perfmon pebs bts rep_good nopl aperfmp erf dni dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave l ahf_lm dtherm tpr_shadow vnmi flexpriority cpufreq
```

The vendor, model and speed of the processor are being shown correctly. However it is not possible to deduce the number of cores on the processor from the above output.

5. nproc

The nproc command just prints out the number of processing units available. Note that the number of processing units might not always be the same as number of cores.

```
$ nproc
4
```

6. dmidecode

The dmidecode command displays some information about the cpu, which includes the socket type, vendor name and various flags.

```
$ sudo dmidecode -t 4
# dmidecode 2.12
```

SMBIOS 2.4 present.

```
Handle 0x0000, DMI type 4, 35 bytes
Processor Information
    Socket Designation: LGA 775
    Type: Central Processor
    Family: Pentium D
    Manufacturer: Intel(R) Corporation
    ID: 7A 06 01 00 FF FB EB BF
    Signature: Type 0, Family 6, Model 23, Stepping 10
    Flags:
        FPU (Floating-point unit on-chip)
        VME (Virtual mode extension)
        DE (Debugging extension)
        PSE (Page size extension)
        TSC (Time stamp counter)
        MSR (Model specific registers)
        PAE (Physical address extension)
        MCE (Machine check exception)
        CX8 (CMPXCHG8 instruction supported)
        APIC (On-chip APIC hardware supported)
        SEP (Fast system call)
        MTRR (Memory type range registers)
        PGE (Page global enable)
        MCA (Machine check architecture)
        CMOV (Conditional move instruction supported)
        PAT (Page attribute table)
        PSE-36 (36-bit page size extension)
        CLFLSH (CLFLUSH instruction supported)
        DS (Debug store)
        ACPI (ACPI supported)
        MMX (MMX technology supported)
        FXSR (FXSAVE and FXSTOR instructions supported)
        SSE (Streaming SIMD extensions)
        SSE2 (Streaming SIMD extensions 2)
        SS (Self-snoop)
        HTT (Multi-threading)
        TM (Thermal monitor supported)
        PBE (Pending break enabled)
    Version: Intel(R) Core(TM)2 Quad CPU      Q8400 @ 2.66GHz
    Voltage: 1.6 V
    External Clock: 333 MHz
    Max Speed: 4000 MHz
    Current Speed: 2666 MHz
    Status: Populated, Enabled
    Upgrade: Socket LGA775
    L1 Cache Handle: 0x0003
    L2 Cache Handle: 0x0001
    L3 Cache Handle: Not Provided
    Serial Number: Not Specified
    Asset Tag: Not Specified
    Part Number: Not Specified
```

7. cpuid

The cpuid command fetches [CPUID](#) information about Intel and AMD x86 processors.

The program can be installed with apt on ubuntu

```
$ sudo apt-get install cpuid
```

And here is sample output

```
$ cpuid
.....
Vendor ID: "GenuineIntel"; CPUID level 13

Intel-specific functions:
Version 0001067a:
Type 0 - Original OEM
Family 6 - Pentium Pro
Model 7 - Pentium III/Pentium III Xeon - external L2 cache
Stepping 10
Reserved 4

Extended brand string: "Intel(R) Core(TM)2 Quad CPU      Q8400 @ 2.66GHz"
CLFLUSH instruction cache line size: 8
Initial APIC ID: 2
Hyper threading siblings: 4

Feature flags bfebfbff:
FPU   Floating Point Unit
VME   Virtual 8086 Mode Enhancements
DE    Debugging Extensions
PSE   Page Size Extensions
TSC   Time Stamp Counter
MSR   Model Specific Registers
PAE   Physical Address Extension
MCE   Machine Check Exception
CX8   COMPXCHG8B Instruction
APIC  On-chip Advanced Programmable Interrupt Controller present and enabled
SEP   Fast System Call
MTRR  Memory Type Range Registers
PGE   PTE Global Flag
MCA   Machine Check Architecture
CMOV  Conditional Move and Compare Instructions
FGPAT Page Attribute Table
PSE-36 36-bit Page Size Extension
CLFSH CFLUSH instruction
DS    Debug store
ACPI  Thermal Monitor and Clock Ctrl
MMX   MMX instruction set
FXSR  Fast FP/MMX Streaming SIMD Extensions save/restore
SSE   Streaming SIMD Extensions instruction set
SSE2  SSE2 extensions
SS    Self Snoop
HT    Hyper Threading
TM    Thermal monitor
31    reserved

.....
```

8. inxi

Inxi is a script that uses other programs to generate a well structured easy to read report about various hardware components on the system. Check out the [full tutorial on inxi](#).

```
$ sudo apt-get install inxi
```

Print out cpu/processor related information

```
$ inxi -C
CPU:       Quad core Intel Core2 Quad CPU Q8400 (-MCP-) cache: 2048 KB flags:
(lm nx sse sse2 sse3 sse4_1 ssse3 vmx)
          Clock Speeds: 1: 1998.00 MHz 2: 1998.00 MHz 3: 1998.00 MHz 4: 1998
.00 MHz
```

18 Tar Command Examples in Linux

The Linux “tar” stands for tape archive, which is used by large number of **Linux/Unix** system administrators to deal with tape drives backup. The tar command used to rip a collection of files and directories into highly compressed archive file commonly called **tarball** or **tar, gzip** and **bzip** in **Linux**. The tar is most widely used command to create compressed archive files and that can be moved easily from one disk to another disk or machine to machine.



Linux Tar Command Examples

In this article we will be going to review and discuss various **tar command examples** including how to create archive files using (**tar**, **tar.gz** and **tar.bz2**) compression, how to extract archive file, extract a single file, view content of file, verify a file, add files or directories to archive file, estimate the size of tar archive file, etc.

The main purpose of this guide is to provide various **tar command examples** that might be helpful for you to understand and become expert in tar archive manipulation.

1. Create tar Archive File

The below example command will create a **tar** archive file **tecmint-14-09-12.tar** for a directory **/home/tecmint** in current working directory. See the example command in action.

```
# tar -cvf tecmint-14-09-12.tar /home/tecmint/
/home/tecmint/
/home/tecmint/cleanfiles.sh
/home/tecmint/openvpn-2.1.4.tar.gz
/home/tecmint/tecmint-14-09-12.tar
/home/tecmint/phpmyadmin-2.11.11.3-1.el5.rf.noarch.rpm
/home/tecmint/rpmforge-release-0.5.2-2.el5.rf.i386.rpm
```

Let's discuss each option that we have used in the above command for creating a tar archive file.

1. **c** – Creates a new .tar archive file.

2. **v** – Verbosely show the .tar file progress.
3. **f** – File name type of the archive file.

2. Create tar.gz Archive File

To create a compressed **gzip** archive file we use the option as **z**. For example the below command will create a compressed **MyImages-14-09-12.tar.gz** file for the directory **/home/MyImages**. (Note : **tar.gz** and **tgz** both are similar).

```
# tar cvzf MyImages-14-09-12.tar.gz /home/MyImages
OR
# tar cvzf MyImages-14-09-12.tgz /home/MyImages
/home/MyImages/
/home/MyImages/Sara-Khan-and-model-Priyanka-Shah.jpg
/home/MyImages/RobertKristenviolent101201.jpg
/home/MyImages/Justintimerlake101125.jpg
/home/MyImages/Mileyphoto101203.jpg
/home/MyImages/JenniferRobert101130.jpg
/home/MyImages/katrinabarbiedoll231110.jpg
/home/MyImages/the-japanese-wife-press-conference.jpg
/home/MyImages/ReesewitherspoonCIA101202.jpg
/home/MyImages/yanaguptabaresf231110.jpg
```

3. Create tar.bz2 Archive File

The **bz2** feature compress and create archive file less than the size of the **gzip**. The **bz2** compression takes more time to compress and decompress files as compared to **gzip** which takes less time. To create highly compressed tar file we use option as **j**. The following example command will create a **Phpfiles-org.tar.bz2** file for a directory **/home/php**. (Note: **tar.bz2** and **tbz** is similar as **tb2**).

```
# tar cvfj Phpfiles-org.tar.bz2 /home/php
OR
# tar cvfj Phpfiles-org.tar.tbz /home/php
OR
# tar cvfj Phpfiles-org.tar.tb2 /home/php
/home/php/
/home/php/iframe_ew.php
/home/php/videos_all.php
/home/php/rss.php
/home/php/index.php
/home/php/vendor.php
/home/php/video_title.php
/home/php/report.php
/home/php/object.html
/home/php/video.php
```

4. Untar tar Archive File

To untar or extract a tar file, just issue following command using option **x** (**extract**). For example the below command will untar the file **public_html-14-09-12.tar** in present working directory. If you want to untar in a different directory then use option as **-C** (**specified directory**).

```
## Untar files in Current Directory ##
# tar -xvf public_html-14-09-12.tar
## Untar files in specified Directory ##
# tar -xvf public_html-14-09-12.tar -C /home/public_html/videos/
/home/public_html/videos/
/home/public_html/videos/views.php
/home/public_html/videos/index.php
/home/public_html/videos/logout.php
/home/public_html/videos/all_categories.php
/home/public_html/videos/feeds.xml
```

5. Uncompress tar.gz Archive File

To Uncompress **tar.gz** archive file, just run following command. If would like to untar in different directory just use option **-C** and the path of the directory, like we shown in the above example.

```
# tar -xvf thumbnails-14-09-12.tar.gz
/home/public_html/videos/thumbnails/
/home/public_html/videos/thumbnails/katdeepika231110.jpg
/home/public_html/videos/thumbnails/katrinabarbie1231110.jpg
/home/public_html/videos/thumbnails/onceuponatime101125.jpg
/home/public_html/videos/thumbnails/playbutton.png
/home/public_html/videos/thumbnails/ReesewitherspoonCIA101202.jpg
/home/public_html/videos/thumbnails/snagItNarration.jpg
/home/public_html/videos/thumbnails/Minissha-Lamba.jpg
/home/public_html/videos/thumbnails/Lindsaydance101201.jpg
/home/public_html/videos/thumbnails/Mileyphoto101203.jpg
```

6. Uncompress tar.bz2 Archive File

To Uncompress highly compressed **tar.bz2** file, just use the following command. The below example command will untar all the **.flv** files from the archive file.

```
# tar -xvf videos-14-09-12.tar.bz2
/home/public_html/videos/flv/katrinabarbie1231110.flv
/home/public_html/videos/flv/BrookmuellerCIA101125.flv
/home/public_html/videos/flv/dollybackinbb4101125.flv
/home/public_html/videos/flv/JenniferRobert101130.flv
/home/public_html/videos/flv/JustinAwardmovie101125.flv
/home/public_html/videos/flv/Lakme-Fashion-Week.flv
/home/public_html/videos/flv/Mileyphoto101203.flv
/home/public_html/videos/flv/Minissha-Lamba.flv
```

7. List Content of tar Archive File

To list the contents of tar archive file, just run the following command with option **t** (**list content**). The below command will list the content of **uploadprogress.tar** file.

```
# tar -tvf uploadprogress.tar
-rw-r--r-- chregu/staff    2276 2011-08-15 18:51:10 package2.xml
-rw-r--r-- chregu/staff    7877 2011-08-15 18:51:10
uploadprogress/examples/index.php
-rw-r--r-- chregu/staff    1685 2011-08-15 18:51:10
uploadprogress/examples/server.php
-rw-r--r-- chregu/staff    1697 2011-08-15 18:51:10
uploadprogress/examples/info.php
-rw-r--r-- chregu/staff    367 2011-08-15 18:51:10 uploadprogress/config.m4
-rw-r--r-- chregu/staff    303 2011-08-15 18:51:10 uploadprogress/config.w32
-rw-r--r-- chregu/staff    3563 2011-08-15 18:51:10
uploadprogress/php_uploadprogress.h
-rw-r--r-- chregu/staff   15433 2011-08-15 18:51:10
uploadprogress/uploadprogress.c
-rw-r--r-- chregu/staff    1433 2011-08-15 18:51:10 package.xml
```

8. List Content tar.gz Archive File

Use the following command to list the content of **tar.gz** file.

```
# tar -tvf staging.tecmint.com.tar.gz
-rw-r--r-- root/root      0 2012-08-30 04:03:57 staging.tecmint.com-
access_log
-rw-r--r-- root/root      587 2012-08-29 18:35:12 staging.tecmint.com-
access_log.1
-rw-r--r-- root/root      156 2012-01-21 07:17:56 staging.tecmint.com-
access_log.2
-rw-r--r-- root/root      156 2011-12-21 11:30:56 staging.tecmint.com-
access_log.3
-rw-r--r-- root/root      156 2011-11-20 17:28:24 staging.tecmint.com-
access_log.4
-rw-r--r-- root/root      0 2012-08-30 04:03:57 staging.tecmint.com-
error_log
-rw-r--r-- root/root     3981 2012-08-29 18:35:12 staging.tecmint.com-
error_log.1
-rw-r--r-- root/root      211 2012-01-21 07:17:56 staging.tecmint.com-
error_log.2
-rw-r--r-- root/root      211 2011-12-21 11:30:56 staging.tecmint.com-
error_log.3
-rw-r--r-- root/root      211 2011-11-20 17:28:24 staging.tecmint.com-
error_log.4
```

9. List Content tar.bz2 Archive File

To list the content of **tar.bz2** file, issue the following command.

```
# tar -tvf Phpfiles-org.tar.bz2
drwxr-xr-x root/root      0 2012-09-15 03:06:08 /home/php/
-rw-r--r-- root/root     1751 2012-09-15 03:06:08 /home/php/iframe_ew.php
-rw-r--r-- root/root    11220 2012-09-15 03:06:08 /home/php/videos_all.php
-rw-r--r-- root/root     2152 2012-09-15 03:06:08 /home/php/rss.php
-rw-r--r-- root/root    3021 2012-09-15 03:06:08 /home/php/index.php
-rw-r--r-- root/root    2554 2012-09-15 03:06:08 /home/php/vendor.php
-rw-r--r-- root/root     406 2012-09-15 03:06:08 /home/php/video_title.php
```

```
-rw-r--r-- root/root      4116 2012-09-15 03:06:08 /home/php/report.php  
-rw-r--r-- root/root     1273 2012-09-15 03:06:08 /home/php/object.html
```

10. Untar Single file from tar File

To extract a single file called **cleanfiles.sh** from **cleanfiles.sh.tar** use the following command.

```
# tar -xvf cleanfiles.sh.tar cleanfiles.sh  
OR  
# tar --extract --file=cleanfiles.sh.tar cleanfiles.sh  
cleanfiles.sh
```

11. Untar Single file from tar.gz File

To extract a single file **tecmintbackup.xml** from **tecmintbackup.tar.gz** archive file, use the command as follows.

```
# tar -zxvf tecmintbackup.tar.gz tecmintbackup.xml  
OR  
# tar --extract --file=tecmintbackup.tar.gz tecmintbackup.xml  
tecmintbackup.xml
```

12. Untar Single file from tar.bz2 File

To extract a single file called **index.php** from the file **Phpfiles-org.tar.bz2** use the following option.

```
# tar -jxvf Phpfiles-org.tar.bz2 home/php/index.php  
OR  
# tar --extract --file=Phpfiles-org.tar.bz2 /home/php/index.php  
/home/php/index.php
```

13. Untar Multiple files from tar, tar.gz and tar.bz2 File

To extract or untar multiple files from the **tar**, **tar.gz** and **tar.bz2** archive file. For example the below command will extract “**file 1**” “**file 2**” from the archive files.

```
# tar -xvf tecmint-14-09-12.tar "file 1" "file 2"  
# tar -zxvf MyImages-14-09-12.tar.gz "file 1" "file 2"  
# tar -jxvf Phpfiles-org.tar.bz2 "file 1" "file 2"
```

14. Extract Group of Files using Wildcard

To extract a group of files we use **wildcard** based extracting. For example, to extract a group of all files whose pattern begins with **.php** from a **tar**, **tar.gz** and **tar.bz2** archive file.

```
# tar -xvf Phpfiles-org.tar --wildcards '*.php'  
# tar -zxvf Phpfiles-org.tar.gz --wildcards '*.php'  
# tar -jxvf Phpfiles-org.tar.bz2 --wildcards '*.php'  
/home/php/iframe_ew.php
```

```
/home/php/videos_all.php  
/home/php/rss.php  
/home/php/index.php  
/home/php/vendor.php  
/home/php/video_title.php  
/home/php/report.php  
/home/php/video.php
```

15. Add Files or Directories to tar Archive File

To add files or directories to existing tar archived file we use the option **r (append)**. For example we add file **xyz.txt** and directory **php** to existing **tecmint-14-09-12.tar** archive file.

```
# tar -rvf tecmint-14-09-12.tar xyz.txt  
# tar -rvf tecmint-14-09-12.tar php  
drwxr-xr-x root/root 0 2012-09-15 02:24:21 home/tecmint/  
-rw-r--r-- root/root 15740615 2012-09-15 02:23:42 home/tecmint/cleanfiles.sh  
-rw-r--r-- root/root 863726 2012-09-15 02:23:41 home/tecmint/openvpn-  
2.1.4.tar.gz  
-rw-r--r-- root/root 21063680 2012-09-15 02:24:21 home/tecmint/tecmint-14-  
09-12.tar  
-rw-r--r-- root/root 4437600 2012-09-15 02:23:41 home/tecmint/phpmyadmin-  
2.11.11.3-1.el5.rf.noarch.rpm  
-rw-r--r-- root/root 12680 2012-09-15 02:23:41 home/tecmint/rpmforge-  
release-0.5.2-2.el5.rf.i386.rpm  
-rw-r--r-- root/root 0 2012-08-18 19:11:04 xyz.txt  
drwxr-xr-x root/root 0 2012-09-15 03:06:08 php/  
-rw-r--r-- root/root 1751 2012-09-15 03:06:08 php/iframe_ew.php  
-rw-r--r-- root/root 11220 2012-09-15 03:06:08 php/videos_all.php  
-rw-r--r-- root/root 2152 2012-09-15 03:06:08 php/rss.php  
-rw-r--r-- root/root 3021 2012-09-15 03:06:08 php/index.php  
-rw-r--r-- root/root 2554 2012-09-15 03:06:08 php/vendor.php  
-rw-r--r-- root/root 406 2012-09-15 03:06:08 php/video_title.php
```

16. Add Files or Directories to tar.gz and tar.bz2 files

The tar command don't have a option to add files or directories to an existing compressed **tar.gz** and **tar.bz2** archive file. If we do try will get the following error.

```
# tar -rvf MyImages-14-09-12.tar.gz xyz.txt  
# tar -rvf Phpfiles-org.tar.bz2 xyz.txt  
tar: This does not look like a tar archive  
tar: Skipping to next header  
xyz.txt  
tar: Error exit delayed from previous errors
```

17. How To Verify tar, tar.gz and tar.bz2 Archive File

To verify any tar or compressed archived file we use option as **W (verify)**. To do, just use the following examples of command. (**Note :** You cannot do verification on a compressed (*.tar.gz, *.tar.bz2) archive file).

```
# tar tvfW tecmint-14-09-12.tar
tar: This does not look like a tar archive
tar: Skipping to next header
tar: Archive contains obsolescent base-64 headers
tar: VERIFY FAILURE: 30740 invalid headers detected
Verify -rw-r--r-- root/root    863726 2012-09-15 02:23:41
/home/tecmint/openvpn-2.1.4.tar.gz
Verify -rw-r--r-- root/root   21063680 2012-09-15 02:24:21
/home/tecmint/tecmint-14-09-12.tar
tar: /home/tecmint/tecmint-14-09-12.tar: Warning: Cannot stat: No such file
or directory
Verify -rw-r--r-- root/root   4437600 2012-09-15 02:23:41
/home/tecmint/phpmyadmin-2.11.11.3-1.el5.rf.noarch.rpm
tar: /home/tecmint/phpmyadmin-2.11.11.3-1.el5.rf.noarch.rpm: Warning: Cannot
stat: No such file or directory
Verify -rw-r--r-- root/root    12680 2012-09-15 02:23:41
/home/tecmint/rpmforge-release-0.5.2-2.el5.rf.i386.rpm
tar: /home/tecmint/rpmforge-release-0.5.2-2.el5.rf.i386.rpm: Warning: Cannot
stat: No such file or directory
Verify -rw-r--r-- root/root        0 2012-08-18 19:11:04 xyz.txt
Verify drwxr-xr-x root/root        0 2012-09-15 03:06:08 php/
```

18. Check the Size of the tar, tar.gz and tar.bz2 Archive File

To check the size of any **tar**, **tar.gz** and **tar.bz2** archive file, use the following command. For example the below command will display the size of archive file in Kilobytes (**KB**).

```
# tar -czf - tecmint-14-09-12.tar | wc -c
12820480
# tar -czf - MyImages-14-09-12.tar.gz | wc -c
112640
# tar -czf - Phpfiles-org.tar.bz2 | wc -c
20480
```

Tar Usage and Options

1. **c** – create a archive file.
2. **x** – extract a archive file.
3. **v** – show the progress of archive file.
4. **f** – filename of archive file.
5. **t** – viewing content of archive file.
6. **j** – filter archive through bzip2.
7. **z** – filter archive through gzip.
8. **r** – append or update files or directories to existing archive file.
9. **W** – Verify a archive file.
10. **wildcards** – Specify patterns in unix tar command.

That's it for now, hope the above **tar command examples** are enough for you to learn and for more information please use **man tar** command.

If you are looking to split any large tar archive file into multiple parts or blocks, just go through this article:

Don't Miss: [Split Large 'tar' Archive into Multiple Files of Certain Size](#)

If we've missed any example please do share with us via comment box and please don't forget to share this article with your friends. This is the best way to say thanks.....

How to Extract Tar Files to Specific or Different Directory in Linux

by [Aaron Kili](#) | Published: January 2, 2016 | Last Updated: July 17, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

The **tar** utility is one of the utilities that you can use to create a backup on a Linux system. It includes many options that one can use to specify the task to achieve.



Extract Linux Tar Files Different or New Directory

One thing to understand is that you can extract tar files to a different or specific directory, not necessarily the current working directory. You can read more about **tar** backup utility with many different examples in the following article, before proceeding further with this article.

[Mastering tar Command with this 18 Examples in Linux](#)

In this guide, we shall take a look at how to extract tar files to a specific or different directory, where you want the files to reside.

The general syntax of **tar** utility for extracting files:

```
# tar -xf file_name.tar -C /target/directory
```

```
# tar -xf file_name.tar.gz --directory /target/directory
```

Note: In the above first syntax, the **-C** option is used to specify a different directory other than the current working directory.

Let us now look at some examples below.

Example 1: Extracting tar Files to a Specific Directory

In the first example, I will extract the files in **articles.tar** to a directory **/tmp/my_article**. Always make sure that the directory into which you want to extract tar file exists.

Let me start by creating the **/tmp/my_article** directory using the command below:

```
# mkdir /tmp/my_article
```

You can include the **-p** option to the above command so that the command does not complain.

To extract the files in **articles.tar** to **/tmp/my_article**, I will run the command bellow:

```
# tar -xvf articles.tar -C /tmp/my_article/
[root@KTech-lab
~/Documents]# mkdir /tmp/my_articles
[root@KTech-lab
~/Documents]# tar -xvf articles.tar -C /tmp/my_articles/
Articles/
Articles/xibo-article.zip
Articles/xibo-article/
Articles/xibo-article/web-install-2.png
Articles/xibo-article/web-install.png
Articles/xibo-article/xibo-article.docx
Articles/xibo-article/logged-in.png
Articles/xibo-article/install-4.png
Articles/xibo-article/download-xibo.png
Articles/xibo-article/install-5.png
Articles/xibo-article/install-3.png
Articles/xibo-article/install-6.png
Articles/article-and-images/
Articles/article-and-images/install-nginx.png
Articles/article-and-images/wp-config.png
Articles/article-and-images/install-hhvm2.png
Articles/article-and-images/install-mariadb.png
Articles/article-and-images/install-mariadb-setrootpass.png
```

Img 01: Extract Tar Files to Different Directory

In the above example I used the **-v** option to monitor the progress of the tar extraction.

Let me also use the `--directory` option instead of `-C` for the example above. It works just in the same way.

```
# tar -xvf articles.tar --directory /tmp/my_articles/
[root@KTech-lab
~/Documents]# tar -xvf articles.tar --directory /tmp/my_articles/
Articles/
Articles/xibo-article.zip
Articles/xibo-article/
Articles/xibo-article/web-install-2.png
Articles/xibo-article/web-install.png
Articles/xibo-article/xibo-article.docx
Articles/xibo-article/logged-in.png
Articles/xibo-article/install-4.png
Articles/xibo-article/download-xibo.png
Articles/xibo-article/install-5.png
Articles/xibo-article/install-3.png
Articles/xibo-article/install-6.png
Articles/article-and-images/
Articles/article-and-images/install-nginx.png
```

Img 02: Extract Tar Files to Specific Directory

Example 2: Extract .tar.gz or .tgz Files to Different Directory

First make sure that you create the specific directory that you want to extract into by using:

```
# mkdir -p /tmp/tgz
```

Now we will extract the contents of `documents.tgz` file to separate `/tmp/tgz/` directory.

```
# tar -zvxf documents.tgz -C /tmp/tgz/
```

```
[root@KTech-lab  
~/Documents]# mkdir -p /tmp/tgz  
[root@KTech-lab  
~/Documents]# tar -zvxf documents.tgz -C /tmp/tgz/  
Articles/  
Articles/xibo-article.zip  
Articles/xibo-article/  
Articles/xibo-article/web-install-2.png  
Articles/xibo-article/web-install.png  
Articles/xibo-article/xibo-article.docx  
Articles/xibo-article/logged-in.png  
Articles/xibo-article/install-4.png  
Articles/xibo-article/download-xibo.png  
Articles/xibo-article/install-5.png  
Articles/xibo-article/install-3.png  
Articles/xibo-article/install-6.png  
Articles/article-and-images/  
Articles/article-and-images/install-nginx.png  
Articles/article-and-images/wp-config.png  
Articles/article-and-images/install-hhvm2.png  
Articles/article-and-images/install-mariadb.png  
Articles/article-and-images/install-mariadb-setrootpass.png  
Articles/article-and-images/wpdb.png  
Articles/article-and-images/install-hhvm-start-service.png  
Articles/article-and-images/start-nginx.png  
Articles/article-and-images/install-mariadb-setrootpass2.png  
Articles/article-and-images/view-site1.png
```

Img 03: Extract tar.gz or .tgz Files to Different Directory

Example 3: Extract tar.bz2, .tar.bz, .tbz or .tbz2 Files to Different Directory

Again repeating that you must create a separate directory before unpacking files:

```
# mkdir -p /tmp/tar.bz2
```

Now we will be unpacking the `documents.tbz2` files to `/tmp/tar.bz2/` directory.

```
# tar -jvxf documents.tbz2 -C /tmp/tar.bz2/
```

```
[root@KTech-lab ~]# mkdir /tmp/tar.bz2
[root@KTech-lab ~]# tar -jvxf documents.tar.bz2 --directory /tmp/tar.bz2/
Articles/
Articles/xibo-article.zip
Articles/xibo-article/
Articles/xibo-article/web-install-2.png
Articles/xibo-article/web-install.png
Articles/xibo-article/xibo-article.docx
Articles/xibo-article/logged-in.png
Articles/xibo-article/install-4.png
Articles/xibo-article/download-xibo.png
Articles/xibo-article/install-5.png
Articles/xibo-article/install-3.png
Articles/xibo-article/install-6.png
Articles/article-and-images/
Articles/article-and-images/install-nginx.png
Articles/article-and-images/wp-config.png
Articles/article-and-images/install-hhvm2.png
Articles/article-and-images/install-mariadb.png
Articles/article-and-images/install-mariadb-setrootpass.png
Articles/article-and-images/wpdb.png
Articles/article-and-images/install-hhvm-start-service.png
Articles/article-and-images/start-nginx.png
Articles/article-and-images/install-mariadb-setrootpass2.png
Articles/article-and-images/view-site1.png
Articles/article-and-images/view-site.png
```

Img 04: Extract tar.bz2 Files to Different Directory

Example 4: Extract Only Specific or Selected Files from Tar Archive

The **tar** utility also allows you to define the files that you want to only extract from a **.tar** file. In the next example, I will extract specific files out of a tar file to a specific directory as follows:

```
# mkdir /backup/tar_extracts
# tar -xvf etc.tar etc/issue etc/fuse.conf etc/mysql/ -C
/backup/tar_extracts/
```

```
[root@KTech-lab ~]# mkdir -p /backup/tar_extracts
[root@KTech-lab ~]# tar -xvf etc.tar etc/issue etc/fuse.conf etc/mysql/ -C /backup/tar_extracts/
etc/mysql/
etc/mysql/debian-start
etc/mysql/debian.cnf
etc/mysql/conf.d/
etc/mysql/conf.d/.keepme
etc/mysql/conf.d/mysqld_safe_syslog.cnf
etc/mysql/conf.d/my5.6.cnf
etc/mysql/my.cnf
etc/issue
etc/fuse.conf
```

Img 05: Extract Specific Files From Tar Archive

Summary

That is it with extracting **tar** files to a specific directory and also extracting specific files from a tar file. If you find this guide helpful or have more information or additional ideas, you can give me a feedback by posting a comment.

How to Archive/Compress Files & Directories, Setting File Attributes and Finding Files in Linux – Part 3

Recently, the Linux Foundation started the **LFCS (Linux Foundation Certified Sysadmin)** certification, a brand new program whose purpose is allowing individuals from all corners of the globe to have access to an exam, which if approved, certifies that the person is knowledgeable in performing basic to intermediate system administration tasks on Linux systems. This includes supporting already running systems and services, along with first-level troubleshooting and analysis, plus the ability to decide when to escalate issues to engineering teams.

Preparation for the LFCS (Linux Foundation Certified Sysadmin)

Part - 3



Linux Foundation Certified Sysadmin – Part 3

Please watch the below video that gives the idea about The Linux Foundation Certification Program.

This post is Part 3 of a 10-tutorial series, here in this part, we will cover how to archive/compress files and directories, set file attributes, and find files on the filesystem, that are required for the LFCS certification exam.

Archiving and Compression Tools

A file archiving tool groups a set of files into a single standalone file that we can backup to several types of media, transfer across a network, or send via email. The most frequently used archiving utility in Linux is **tar**. When an archiving utility is used along with a compression tool, it allows to reduce the disk size that is needed to store the same files and information.

The tar utility

tar bundles a group of files together into a single archive (commonly called a tar file or tarball). The name originally stood for tape archiver, but we must note that we can use this tool to archive data to any kind of writeable media (not only to tapes). Tar is normally used with a compression tool such as **gzip**, **bzip2**, or **xz** to produce a compressed tarball.

Basic syntax:

```
# tar [options] [pathname ...]
```

Where ... represents the expression used to specify which files should be acted upon.

Most commonly used tar commands

Long option	Abbreviation	Description
-create	c	Creates a tar archive
-concatenate	A	Appends tar files to an archive
-append	r	Appends files to the end of an archive
-update	u	Appends files newer than copy in archive
-diff or -compare	d	Find differences between archive and file system
-file archive	f	Use archive file or device ARCHIVE
-list	t	Lists the contents of a tarball
-extract or -get	x	Extracts files from an archive

Normally used operation modifiers

Long option	Abbreviation	Description
-directory dir	C	Changes to directory dir before performing operations
-same-permissions	p	Preserves original permissions
-verbose	v	Lists all files read or extracted. When this flag is used along with -list, the file sizes, ownership, and time stamps are displayed.
-verify	w	Verifies the archive after writing it
-exclude file	-	Excludes file from the archive

- exclude=pattern	X	Exclude files, given as a PATTERN
-gzip or -gunzip	z	Processes an archive through gzip
-bzip2	j	Processes an archive through bzip2
-xz	J	Processes an archive through xz

Gzip is the oldest compression tool and provides the least compression, while **bzip2** provides improved compression. In addition, **xz** is the newest but (usually) provides the best compression. This advantages of best compression come at a price: the time it takes to complete the operation, and system resources used during the process.

Normally, **tar** files compressed with these utilities have **.gz**, **.bz2**, or **.xz** extensions, respectively. In the following examples we will be using these files: file1, file2, file3, file4, and file5.

Grouping and compressing with gzip, bzip2 and xz

Group all the files in the current working directory and compress the resulting bundle with **gzip**, **bzip2**, and **xz** (please note the use of a regular expression to specify which files should be included in the bundle – this is to prevent the archiving tool to group the tarballs created in previous steps).

```
# tar czf myfiles.tar.gz file[0-9]
# tar cjf myfiles.tar.bz2 file[0-9]
# tar cJf myfile.tar.xz file[0-9]
gacanepa@debian:~/LFCS/lab3$ ls -lh | grep tar
-rw-r--r-- 1 gacanepa gacanepa 79K Oct 13 09:44 myfiles.tar.bz2
-rw-r--r-- 1 gacanepa gacanepa 101K Oct 13 09:43 myfiles.tar.gz
-rw-r--r-- 1 gacanepa gacanepa 84K Oct 13 09:44 myfiles.tar.xz
gacanepa@debian:~/LFCS/lab3$
```

Compress Multiple Files

Listing the contents of a tarball and updating / appending files to the bundle

List the contents of a tarball and display the same information as a long directory listing. Note that **update** or **append** operations cannot be applied to compressed files directly (if you need to update or append a file to a compressed tarball, you need to uncompress the tar file and update / append to it, then compress again).

```
# tar tvf [tarball]
```

```
gacanepa@debian:~/LFCS/lab3$ tar tvf myfiles.tar.gz
-rw-r--r-- gacanepa/gacanepa 7986 2014-10-13 09:16 file1
-rw-r--r-- gacanepa/gacanepa 2232 2014-10-13 09:16 file2
-rw-r--r-- gacanepa/gacanepa 3214 2014-10-13 09:17 file3
-rw-r--r-- gacanepa/gacanepa 321072 2014-10-13 09:20 file4
-rw-r--r-- gacanepa/gacanepa 37271 2014-10-13 09:21 file5
gacanepa@debian:~/LFCS/lab3$ http://www.tecmint.com
```

List Archive Content

Run any of the following commands:

```
# gzip -d myfiles.tar.gz      [#1]
# bzip2 -d myfiles.tar.bz2    [#2]
# xz -d myfiles.tar.xz       [#3]
```

Then

```
# tar --delete --file myfiles.tar file4 (deletes the file inside the tarball)
# tar --update --file myfiles.tar file4 (adds the updated file)
```

and

```
# gzip myfiles.tar           [ if you choose #1 above ]
# bzip2 myfiles.tar          [ if you choose #2 above ]
# xz myfiles.tar             [ if you choose #3 above ]
```

Finally,

```
# tar tvf [tarball] #again
```

and compare the modification date and time of **file4** with the same information as shown earlier.

Excluding file types

Suppose you want to perform a backup of user's **home** directories. A good sysadmin practice would be (may also be specified by company policies) to exclude all video and audio files from backups.

Maybe your first approach would be to exclude from the backup all files with an **.mp3** or **.mp4** extension (or other extensions). What if you have a clever user who can change the extension to **.txt** or **.b kp**, your approach won't do you much good. In order to detect an audio or video file, you need to check its file type with file. The following shell script will do the job.

```
#!/bin/bash
# Pass the directory to backup as first argument.
DIR=$1
# Create the tarball and compress it. Exclude files with the MPEG string in
its file type.
```

```

# -If the file type contains the string mpeg, $? (the exit status of the most
recently executed command) expands to 0, and the filename is redirected to
the exclude option. Otherwise, it expands to 1.
# -If $? equals 0, add the file to the list of files to be backed up.
tar X <(for i in $DIR/*; do file $i | grep -i mpeg; if [ $? -eq 0 ]; then
echo $i; fi;done) -cjf backupfile.tar.bz2 $DIR/*
1 #!/bin/bash
2 # Tell the directory to backup in first argument.
3 DIR=$1
4 # Create the tarball and compress it, excluding files with the specified file extension.
5 # If the file type contains the string mpeg, the exit status of the most recently
6 # executed command expands to 0, and the filename is redirected to the exclude option.
7 # Otherwise, it expands to 1.
8 # Then it adds 0, and the file to the list of files to be backed up.
9 tar X <(for i in $DIR/*; do file $i | grep -i mpeg; if [ $? -eq 0 ]; then echo $i; fi;done) -cjf backupfile.tar.bz2 $DIR/*

```

<http://www.tecmint.com>

Exclude Files in tar

Restoring backups with tar preserving permissions

You can then restore the backup to the original user's home directory (user_restore in this example), preserving permissions, with the following command.

```

# tar xjf backupfile.tar.bz2 --directory user_restore --same-permissions
root@debian:/home/gacanepa/LFCS/lab3# tar xjf backupfile.tar.bz2 --directory user_restore --same-permissions
root@debian:/home/gacanepa/LFCS/lab3# ls -l user_restore/
user_restore/backupfile.bz2  user_restore/file1  user_restore/file3  user_restore/file5          user_restore/myfiles.tar.gz
user_restore/backup.sh      user_restore/file2  user_restore/file4  user_restore/myfiles.tar.bz2  user_restore/myfiles.tar.xz
user_restore/aux:
file1  file2  file3  file4  file5
root@debian:/home/gacanepa/LFCS/lab3#

```

<http://www.tecmint.com>

Restore Files from Archive

Read Also:

1. [18 tar Command Examples in Linux](#)
2. [Dtrx – An Intelligent Archive Tool for Linux](#)

Using find Command to Search for Files

The **find** command is used to search recursively through directory trees for files or directories that match certain characteristics, and can then either print the matching files or directories or perform other operations on the matches.

Normally, we will search by name, owner, group, type, permissions, date, and size.

Basic syntax:

```
# find [directory_to_search] [expression]
```

Finding files recursively according to Size

Find all files (**-f**) in the current directory (.) and 2 subdirectories below (**-maxdepth 3** includes the current working directory and 2 levels down) whose size (**-size**) is greater than **2 MB**.

```
# find . -maxdepth 3 -type f -size +2M
root@debian:/home/gacanepa# find . -maxdepth 3 -type f -size +2M
./LFCS/lab3/SAPPProductSuite.mp4
./LFCS/lab3/StayWithMeTonight.mp3
./Books/Wiley.Linux.Command.Line.and.Shell.Scripting.Bible.May.2008.pdf
./Books/Head.First.C.pdf
./Books/libre_m2_baja.pdf
root@debian:/home/gacanepa# http://www.tecmint.com
```

Find Files Based on Size

Finding and deleting files that match a certain criteria

Files with **777** permissions are sometimes considered an open door to external attackers. Either way, it is not safe to let anyone do anything with files. We will take a rather aggressive approach and delete them! ('{}' + is used to "collect" the results of the search).

```
# find /home/user -perm 777 -exec rm '{}' +
root@debian:/home/gacanepa# ls -l | grep sed
-rwxrwxrwx 1 gacanepa gacanepa 13761 Oct  8 09:04 sed.pdf
root@debian:/home/gacanepa# find . -perm 777 -exec rm '{}' +
root@debian:/home/gacanepa# ls -l | grep sed
root@debian:/home/gacanepa# http://www.tecmint.com
```

Find Files with 777Permission

Finding files per atime or mtime

Search for configuration files in **/etc** that have been accessed (**-atime**) or modified (**-mtime**) more (+**180**) or less (**-180**) than **6** months ago or exactly **6** months ago (**180**).

Modify the following command as per the example below:

```
# find /etc -iname "*.conf" -mtime -180 -print
```

```
root@debian:~# find /etc -iname "*.conf" -mtime -180 -print
/etc/samba/smb.conf
/etc/cups/cups-pdf.conf
/etc/cups/printers.conf
/etc/cups/cupsd.conf
root@debian:~# find /etc -iname "*.conf" -mtime 180 -print
root@debian:~# find /etc -iname "*.conf" -atime 180 -print
root@debian:~# find /etc -iname "*.conf" -atime +180 -print
/etc/dhcp/dhclient.conf
/etc/openal/alsof.conf
/etc/gconf/2/evoldap.conf
/etc/dbus-1/session.conf
/etc/logrotate.conf
http://www.tecmint.com
```

Find Modified Files

Read Also: [35 Practical Examples of Linux ‘find’ Command](#)

File Permissions and Basic Attributes

The first **10** characters in the output of **ls -l** are the file attributes. The first of these characters is used to indicate the file type:

1. **-** : a regular file
2. **-d** : a directory
3. **-l** : a symbolic link
4. **-c** : a character device (which treats data as a stream of bytes, i.e. a terminal)
5. **-b** : a block device (which handles data in blocks, i.e. storage devices)

The next nine characters of the file attributes are called the file mode and represent the read (**r**), write (**w**), and execute (**x**) permissions of the file’s owner, the file’s group owner, and the rest of the users (commonly referred to as “the world”).

Whereas the read permission on a file allows the same to be opened and read, the same permission on a directory allows its contents to be listed if the execute permission is also set. In addition, the execute permission in a file allows it to be handled as a program and run, while in a directory it allows the same to be cd’ed into it.

File permissions are changed with the **chmod** command, whose basic syntax is as follows:

```
# chmod [new_mode] file
```

Where **new_mode** is either an octal number or an expression that specifies the new permissions.

The octal number can be converted from its binary equivalent, which is calculated from the desired file permissions for the owner, the group, and the world, as follows:

The presence of a certain permission equals a power of 2 (**r=2²**, **w=2¹**, **x=2⁰**), while its absence equates to **0**. For example:

http://www.tecmint.com								
r	w	x	r	-	-	r	-	-
4	2	1	4	0	0	4	0	0
4+2+1=7			4+0+0=4			4+0+0=4		

File Permissions

To set the file's permissions as above in octal form, type:

```
# chmod 744 myfile
```

You can also set a file's mode using an expression that indicates the owner's rights with the letter **u**, the group owner's rights with the letter **g**, and the rest with **o**. All of these “**individuals**” can be represented at the same time with the letter **a**. Permissions are granted (or revoked) with the + or – signs, respectively.

Revoking execute permission for a shell script to all users

As we explained earlier, we can revoke a certain permission prepending it with the minus sign and indicating whether it needs to be revoked for the owner, the group owner, or all users. The one-liner below can be interpreted as follows: Change mode for all (**a**) users, revoke (–) execute permission (**x**).

```
# chmod a-x backup.sh
```

Granting read, write, and execute permissions for a file to the owner and group owner, and read permissions for the world.

When we use a 3-digit octal number to set permissions for a file, the first digit indicates the permissions for the owner, the second digit for the group owner and the third digit for everyone else:

1. **Owner:** ($r=2^2 + w=2^1 + x=2^0 = 7$)
2. **Group owner:** ($r=2^2 + w=2^1 + x=2^0 = 7$)
3. **World:** ($r=2^2 + w=0 + x=0 = 4$),

```
# chmod 774 myfile
```

In time, and with practice, you will be able to decide which method to change a file mode works best for you in each case. A long directory listing also shows the file's owner and its group owner (which serve as a rudimentary yet effective access control to files in a system):

```
root@debian:~# ls -l
total 1808
-rw-r--r-- 1 root root      48 Oct  6 13:51 email_body.txt
drwx----- 2 root root    4096 Oct  3 22:06 PDF
-rw-r--r-- 1 root root   1105 Oct 11 00:46 reporte.html
-rw----- 1 root root 386504 Oct  6 14:48 sent
-rw-r--r-- 1 root root       0 Jun  9 17:58 TestFile
-rw-r--r-- 1 root root 1298432 May  6 2013 upgrade-wheezystep.script
-rw-r--r-- 1 root root 138674 May  6 2013 upgrade-wheezystep.time
root@debian:~#
```

Linux File Listing

File ownership is changed with the **chown** command. The owner and the group owner can be changed at the same time or separately. Its basic syntax is as follows:

```
# chown user:group file
```

Where at least user or group need to be present.

Few Examples

Changing the owner of a file to a certain user.

```
# chown gacanepa sent
```

Changing the owner and group of a file to an specific user:group pair.

```
# chown gacanepa:gacanepa TestFile
```

Changing only the group owner of a file to a certain group. Note the colon before the group's name.

```
# chown :gacanepa email_body.txt
```

Conclusion

As a sysadmin, you need to know how to create and restore backups, how to find files in your system and change their attributes, along with a few tricks that can make your life easier and will prevent you from running into future issues.

I hope that the tips provided in the present article will help you to achieve that goal. Feel free to add your own tips and ideas in the comments section for the benefit of the community. Thanks in advance!

Understanding Shutdown, Poweroff, Halt and Reboot Commands in Linux

In this article, we will explain to you the difference between **shutdown**, **poweroff**, **halt** and **reboot** Linux commands. We will make clear what they actually do when you execute them with available options.

If you are hoping to dive into Linux server administration, then these are [some of the important Linux commands](#) you need to fully understand for effective and reliable server administration.

Normally, when you want to turn off or reboot your machine, you'll run one of the commands below:

Shutdown Command

shutdown schedules a time for the system to be powered down. It may be used to halt, power-off or reboot the machine.

You may specify a time string (which is usually “**now**” or “**hh:mm**” for hour/minutes) as the first argument. Additionally, you may set a wall message to be sent to all logged-in users before the system goes down.

Important: If the time argument is used, 5 minutes before the system goes down the **/run/nologin** file is created to ensure that further logins will not be allowed.

Examples of shutdown commands:

```
# shutdown
# shutdown now
# shutdown 13:20
# shutdown -p now      #poweroff the machine
# shutdown -H now      #halt the machine
# shutdown -r09:35     #reboot the machine at 09:35am
```

To cancel a pending shutdown, simply type the command below:

```
# shutdown -c
```

Halt Command

halt instructs the hardware to stop all CPU functions, but leaves it powered on. You can use it to get the system to a state where you can perform low level maintenance.

Note that in some cases it completely shuts down the system. Below are examples of halt commands:

```
# halt          #halt the machine
# halt -p       #poweroff the machine
# halt --reboot #reboot the machine
```

Power off Command

poweroff sends an ACPI signal which instructs the system to power down.

The following are examples of poweroff commands:

```
# poweroff      #poweroff the machine
# poweroff --halt #halt the machine
# poweroff --reboot #reboot the machine
```

Reboot Command

reboot instructs the system to restart.

```
# reboot        #reboot the machine
# reboot --halt #halt the machine
# reboot -p     #poweroff the machine
```

That's all! As mentioned earlier on, understanding these commands will enable to effectively and reliably manage Linux server in a multi-user environment. Do you have any additional ideas? Share them with us via the comments section below.

How to Install Ubuntu 16.10/16.04 Alongside With Windows 10 or 8 in Dual-Boot

Ubuntu 16.10 released with a support of **9** months until **July 2017** and **Ubuntu 16.04** has been released in wild by Canonical with a life circle of **5** years support.

This tutorial will guide you on how you can perform the installation of **Ubuntu 16.10** and **Ubuntu 16.04** in dual-boot with a Microsoft Operating System on machines that come pre-installed with **Windows 10**.

For fresh Ubuntu 16.04/16.10 installation, read our article about [Ubuntu 16.04 Desktop Installation Guide](#) and [Ubuntu 16.10 Installation Guide](#)

This guide assumes that your machine comes pre-installed with **Windows 10 OS** or an older version of Microsoft Windows, such as **Windows 8.1** or **8**.

In case your hardware uses **UEFI** then you should modify the **EFI** settings and disable **Secure Boot** feature.

If your computer has no other Operating System already installed and you plan to use a Windows variant alongside **Ubuntu 16.04/16.10**, you should first install Microsoft Windows and then proceed with Ubuntu 16.04 installation.

In this particular case, on Windows installation steps, when formatting the hard disk, you should allocate a free space on the disk with at least 20 GB in size in order use it later as a partition for Ubuntu installation.

Requirements

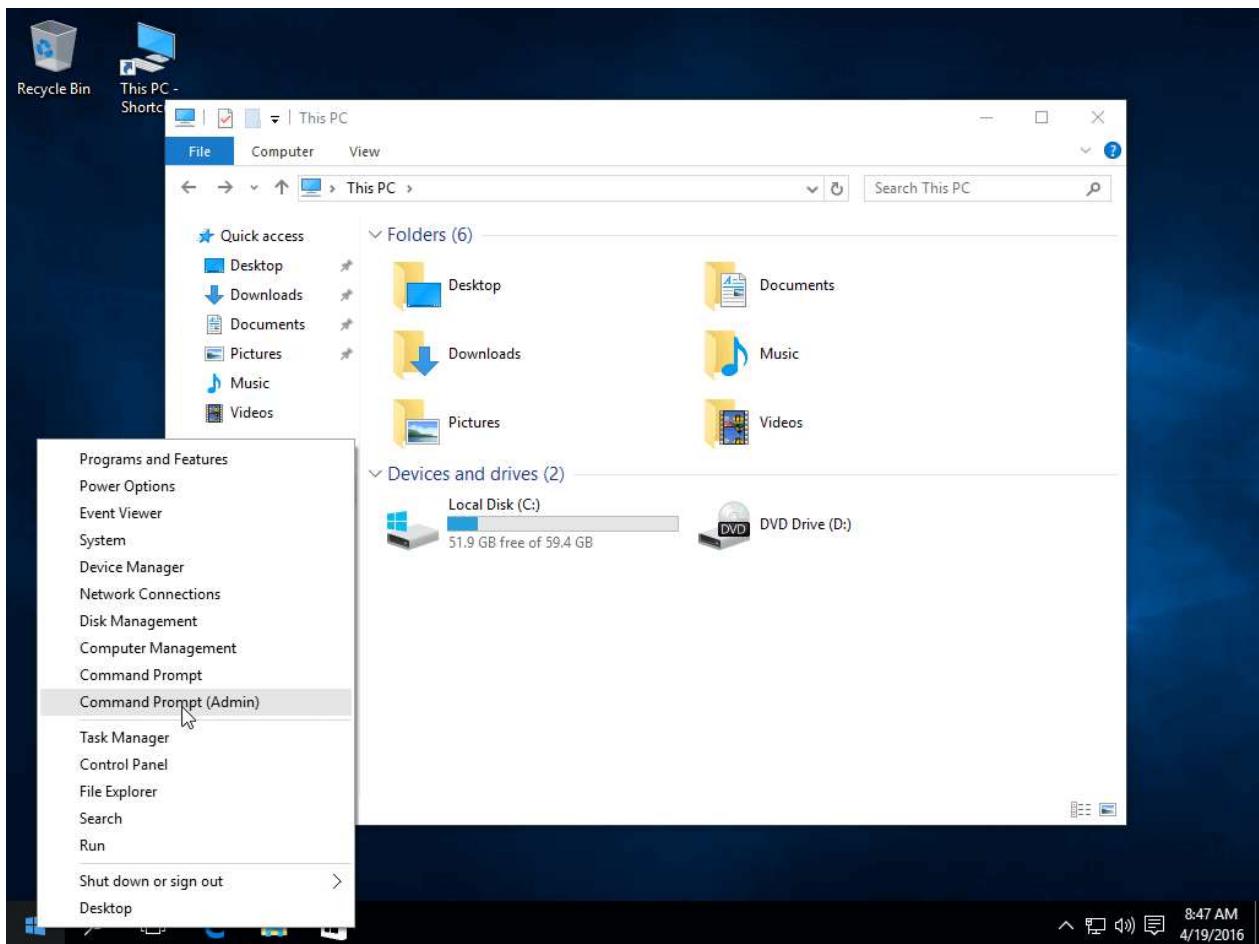
Download **Ubuntu 16.04** and **Ubuntu 16.10** ISO Image as per your system architecture using following link:

1. [Ubuntu 16.10 Desktop](#)
2. [Ubuntu 16.04 Desktop](#)

Step 1: Prepare Windows Machine for Dual-Boot

1. The first thing you need to take care is to create a free space on the computer hard disk in case the system is installed on a single partition.

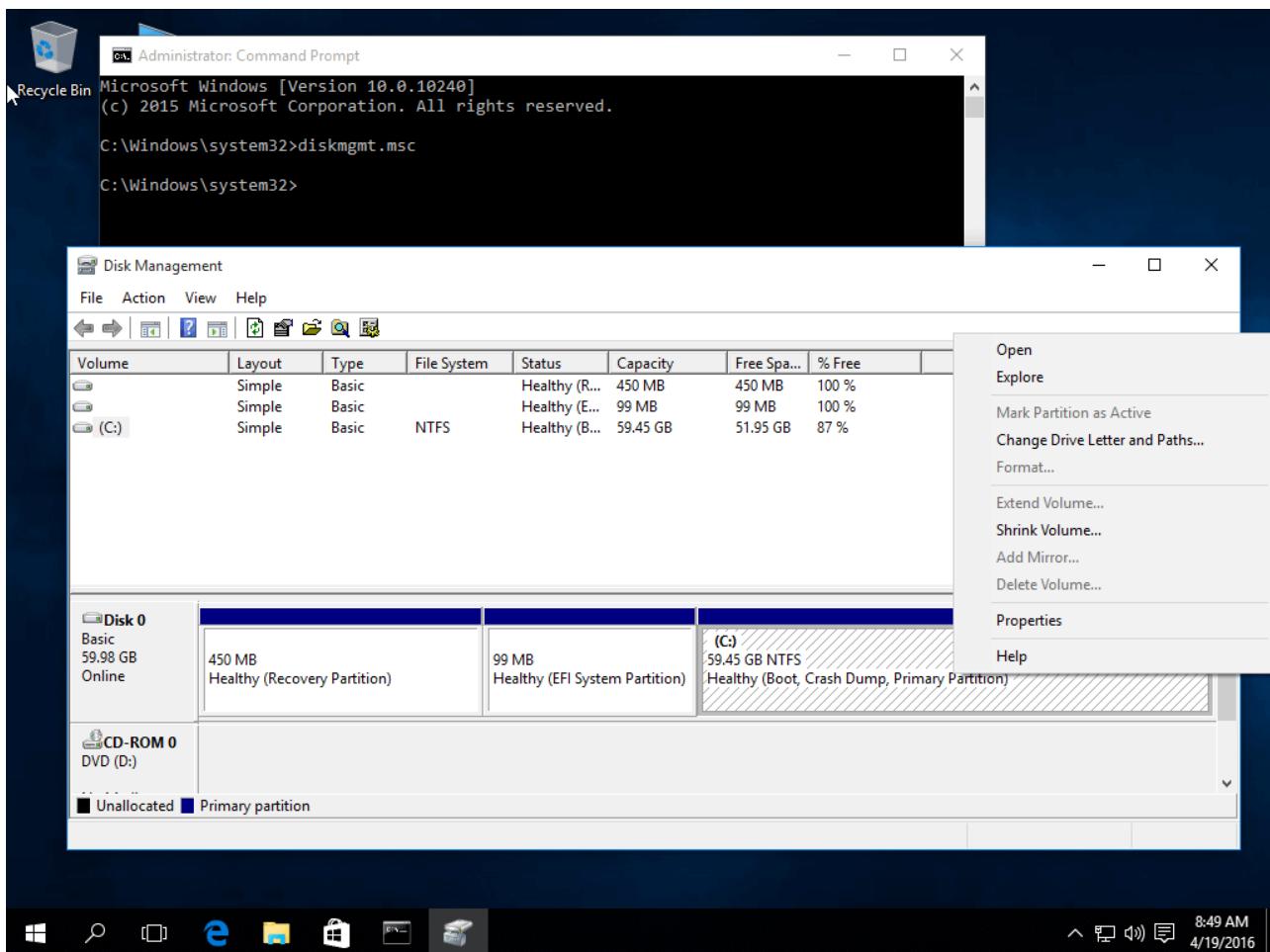
Login to your Windows machine with an administrative account and right click on the **Start Menu -> Command Prompt** (Admin) in order to enter Windows Command Line.



Preparing Windows for Dual-Boot with Ubuntu 16.04

2. Once in **CLI**, type `diskmgmt.msc` on prompt and the **Disk Management** utility should open. From here, right click on C: partition and select **Shrink Volume** in order to resize the partition.

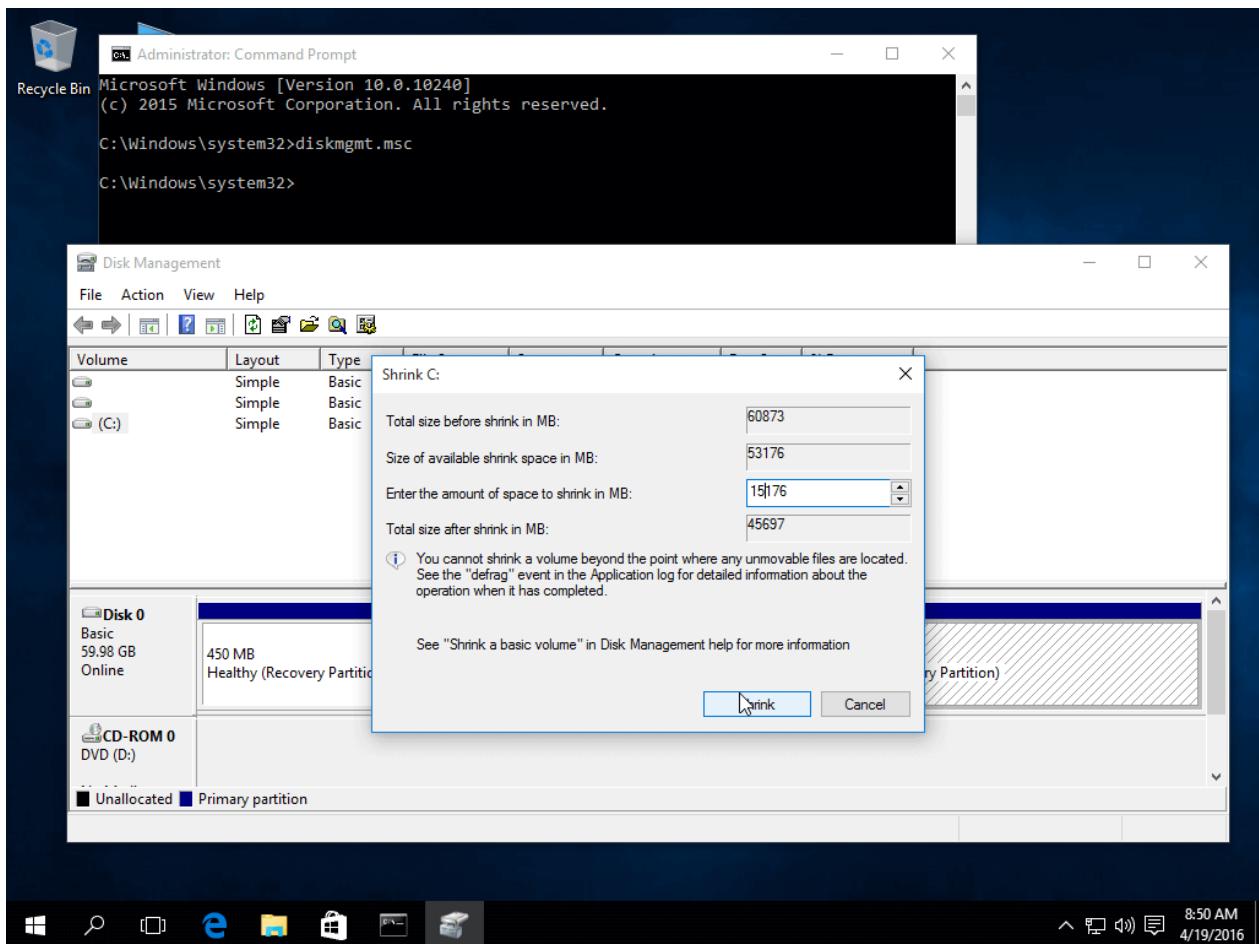
C:\Windows\system32\>**diskmgmt.msc**



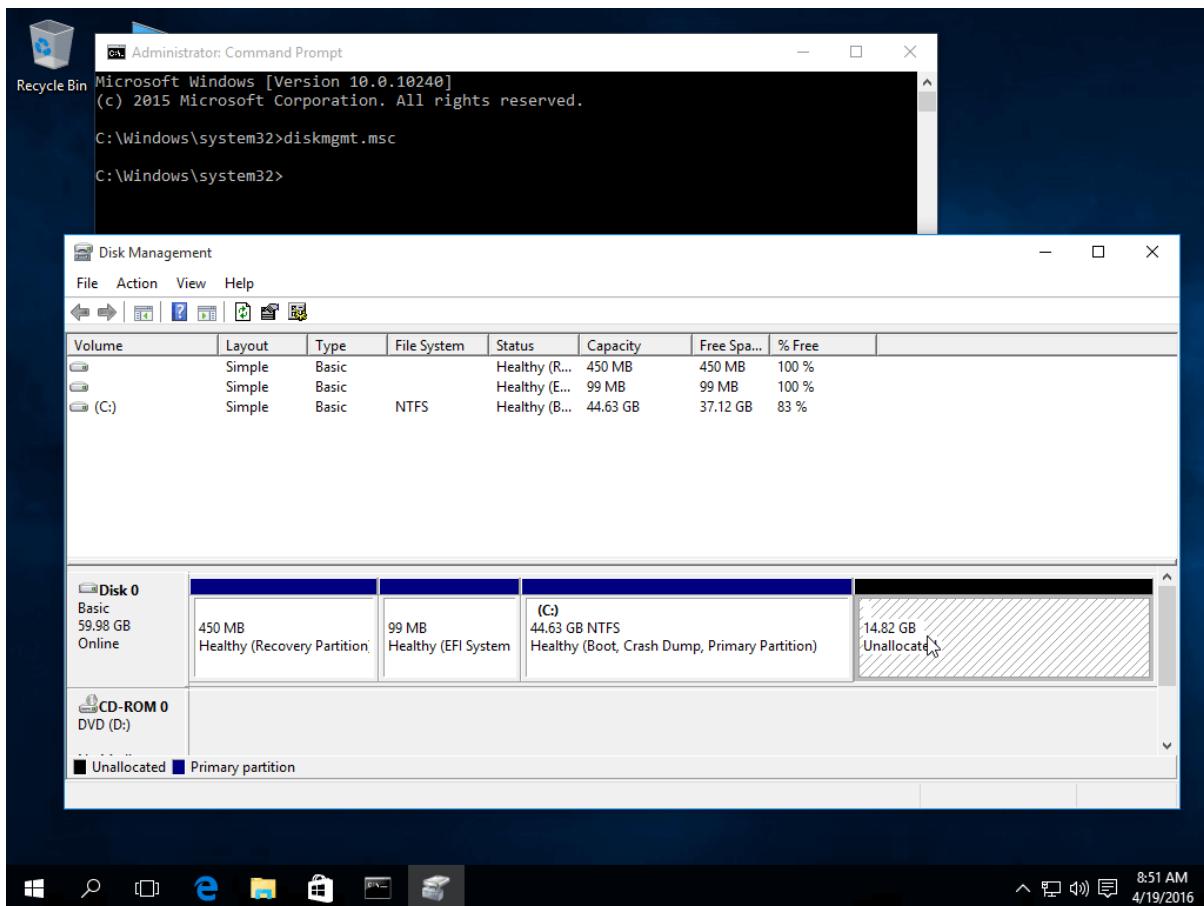
Shrink Volume to Resize Partition

3. On Shrink c: enter a value on space to shrink in MB (use at least **20000 MB** depending on the C: partition size) and hit **Shrink** to start partition resize as illustrated below (the value of space shrink from below image is lower and only used for demonstration purposes).

Once the space has been resized you will see a new unallocated space on the hard drive. Leave it as default and reboot the computer in order to proceed with Ubuntu 16.04 installation.



Create Windows Partition for Ubuntu 16.04 Installation



Windows Partition for Dual Boot Ubuntu 16.04

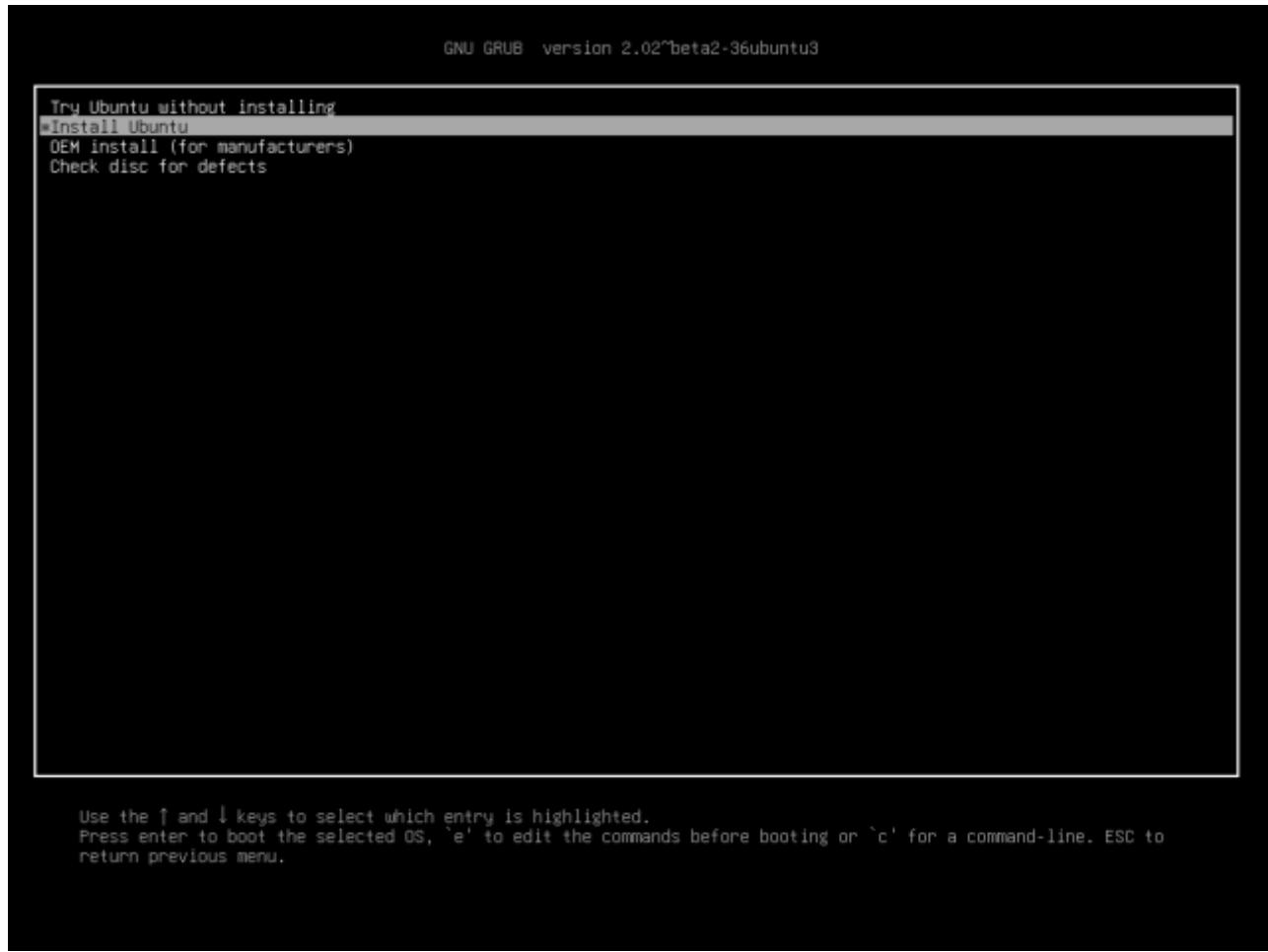
Step 2: Install Ubuntu 16.04 with Windows Dual-Boot

4. Now it's time to install **Ubuntu 16.04**. Go the download link from the topic description and grab **Ubuntu Desktop 16.04 ISO** image.

Burn the image to a DVD or create a bootable USB stick using a utility such as **Universal USB Installer** (BIOS compatible) or **Rufus** (UEFI compatible).

Place the USB stick or DVD in the appropriate drive, reboot the machine and instruct the **BIOS/UEFI** to boot-up from the DVD/USB by pressing a special function key (usually **F12**, **F10** or **F2** depending on the vendor specifications).

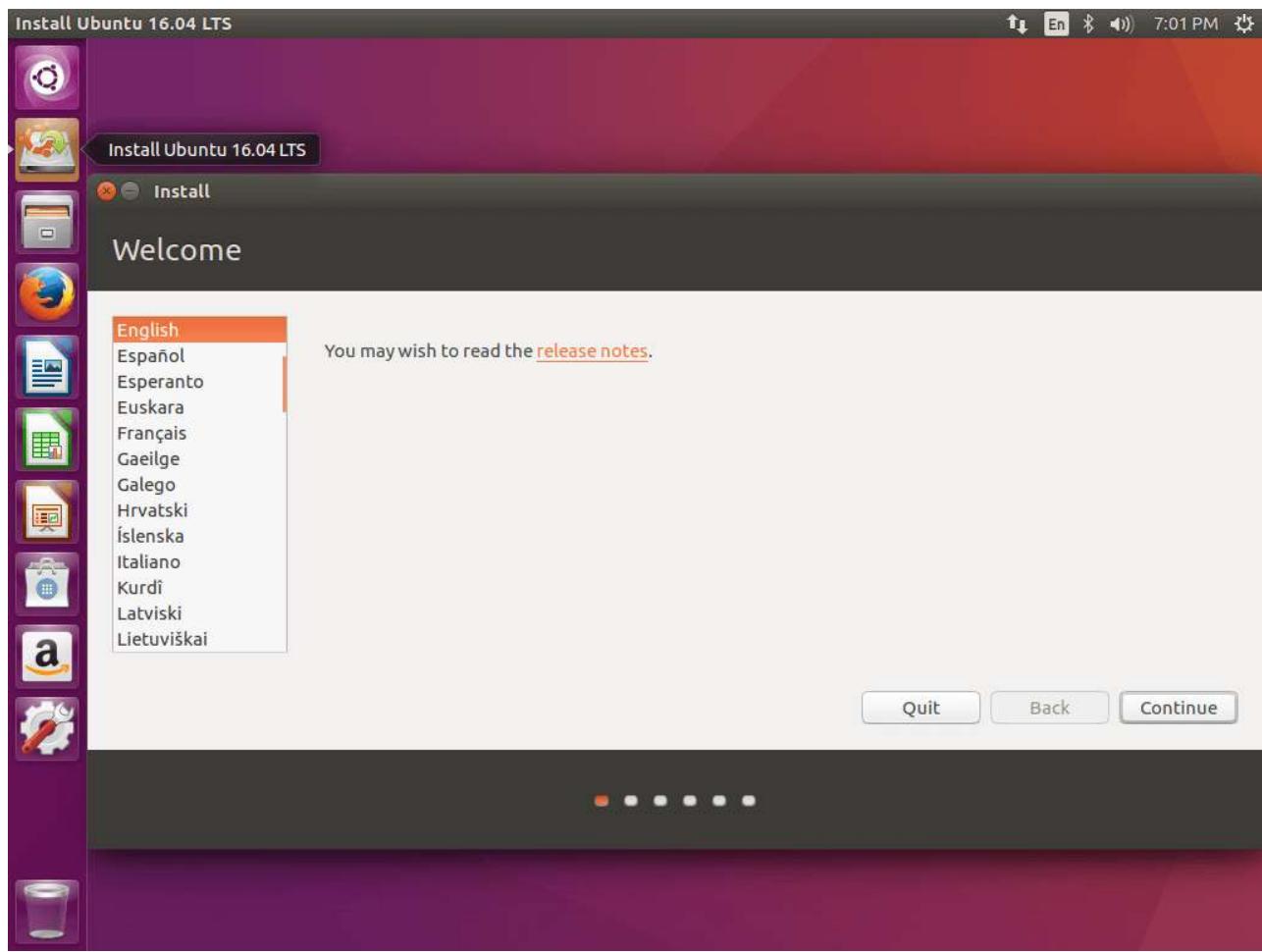
Once the media boot-up a new grub screen should appear on your monitor. From the menu select **Install Ubuntu** and hit **Enter** to continue.



Ubuntu 16.04 Install Boot Screen

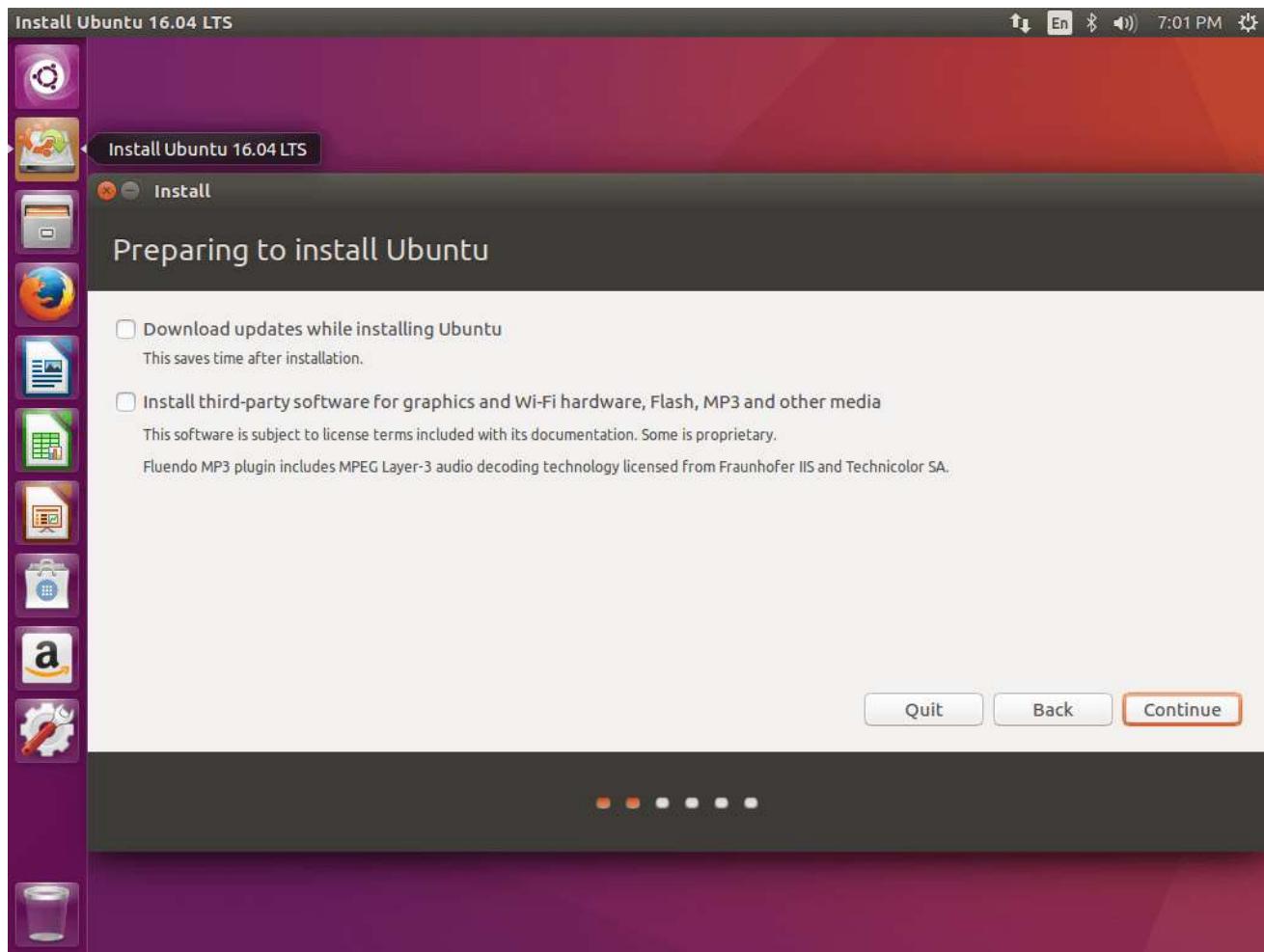
5. After the boot media finishes loading into RAM you will end-up with a completely functional Ubuntu system running in live-mode.

On the Launcher hit on the second icon from top, **Install Ubuntu 16.04 LTS**, and the installer utility will start. Choose the language you wish to perform the installation and click on **Continue** button to proceed further.



Select Ubuntu 16.04 Installation Language

6. Next, leave both options from **Preparing to Install Ubuntu** unchecked and hit on **Continue** button again.

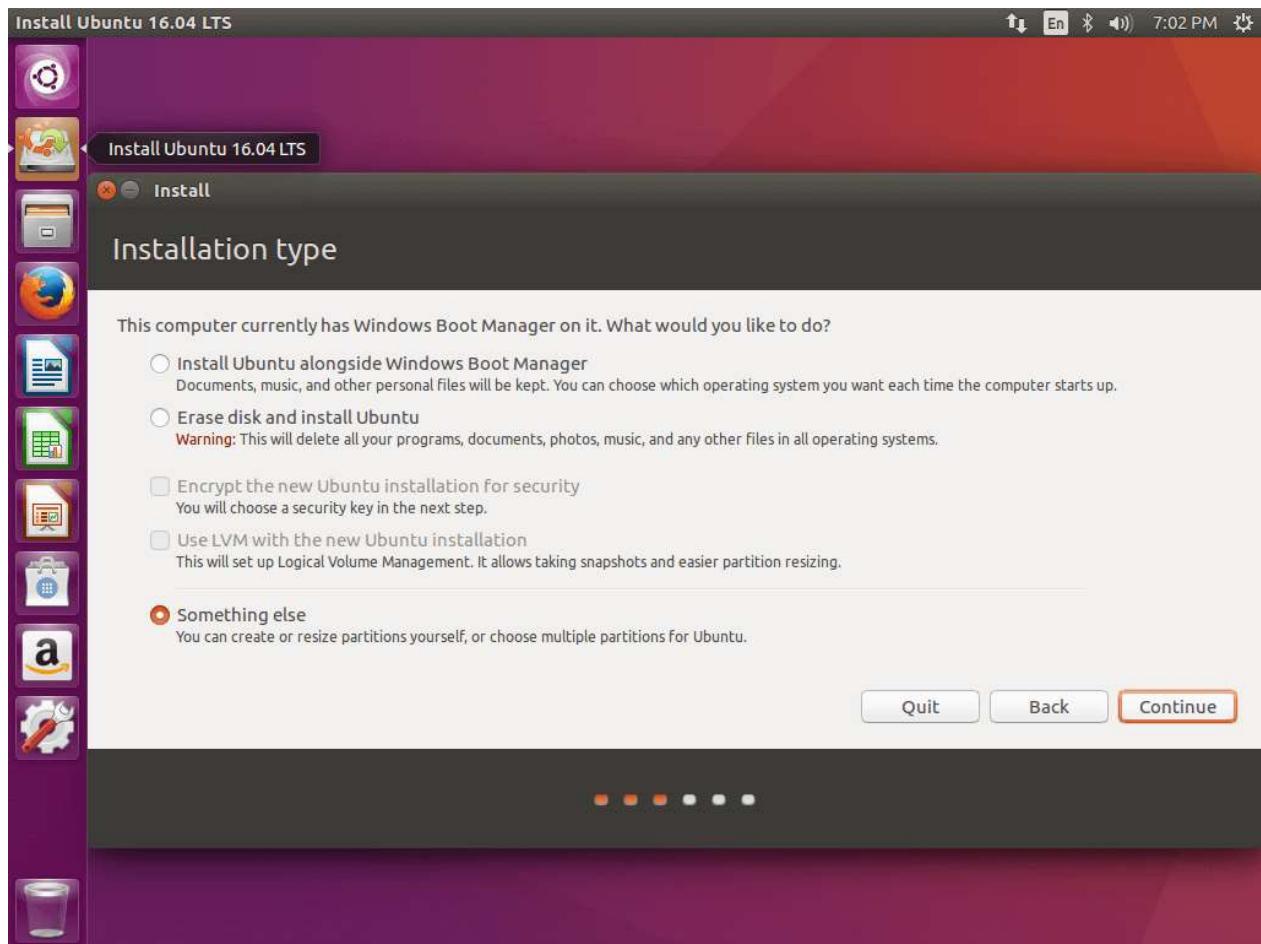


Preparing Ubuntu 16.04 Installation

7. Now it's time to select an Installation Type. You can choose to **Install Ubuntu** alongside **Windows Boot Manager**, option that will automatically take care of all the partition steps.

Use this option if you don't require personalized partition scheme. In case you want a custom partition layout, check the **Something else** option and hit on **Continue** button to proceed further.

The option **Erase disk** and install Ubuntu should be avoided on dual-boot because is potentially dangerous and will wipe out your disk.

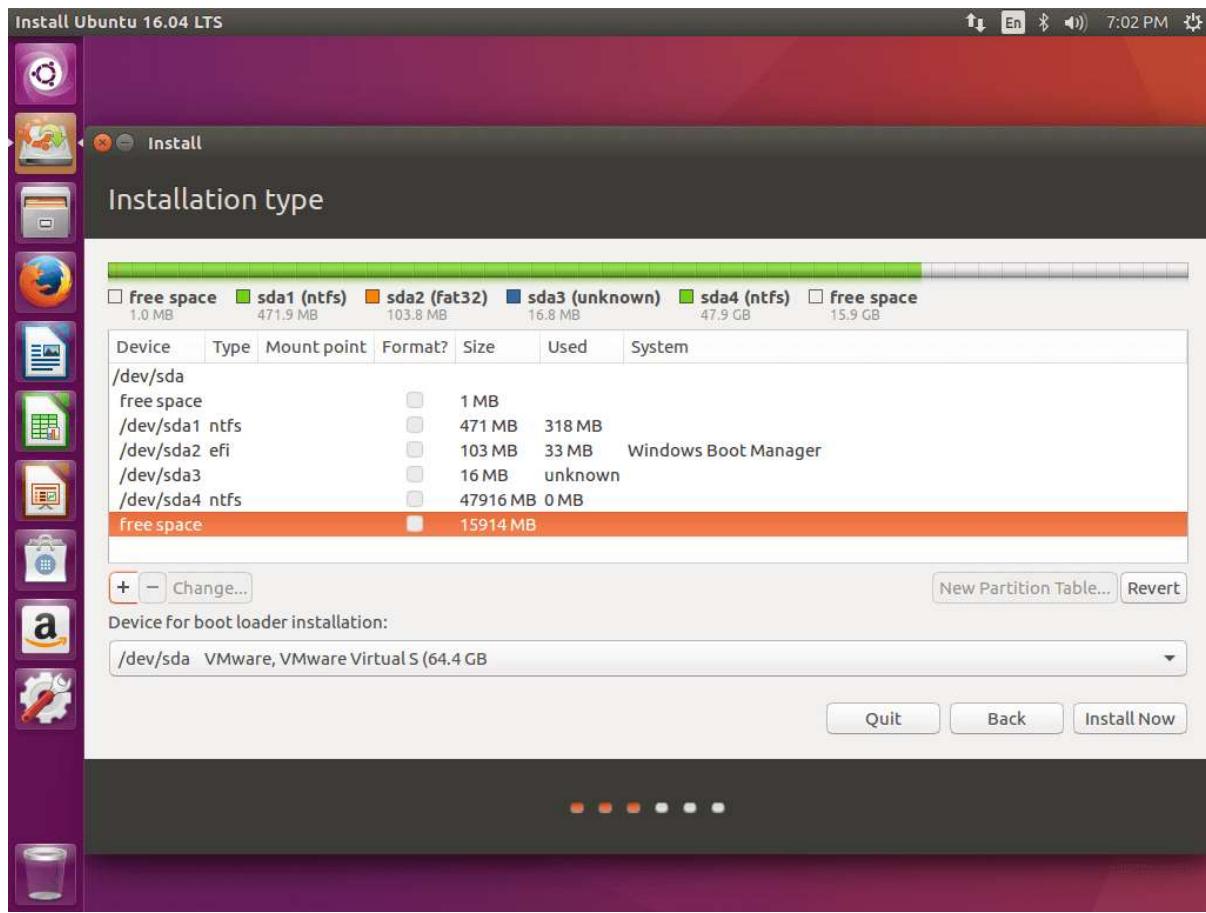


Select Ubuntu 16.04 Installation Type

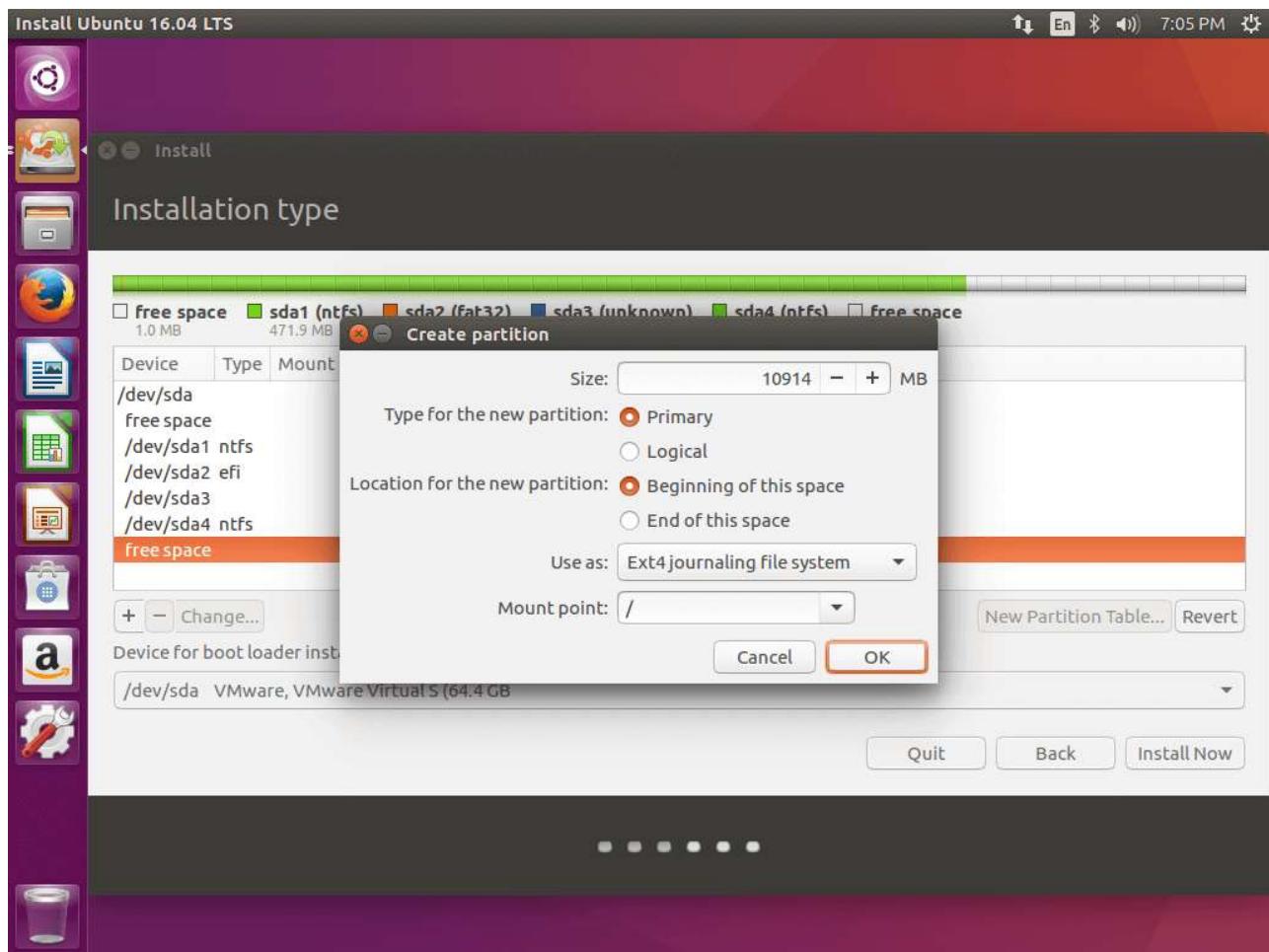
8. On this step we'll create our custom partition layout for **Ubuntu 16.04**. On this guide will recommend that you create two partitions, one for `root` and the other for `home` accounts data and no partition for `swap` (use a swap partition only if you have limited RAM resources or you use a fast SSD).

To create the first partition, the `root` partition, select the free space (the shrink space from Windows created earlier) and hit on the `+` icon below. On partition settings use the following configurations and hit **OK** to apply changes:

1. Size = at least **20000 MB**
2. Type for the new partition = **Primary**
3. Location for the new partition = **Beginning**
4. Use as = **EXT4** journaling file system
5. Mount point = `/`



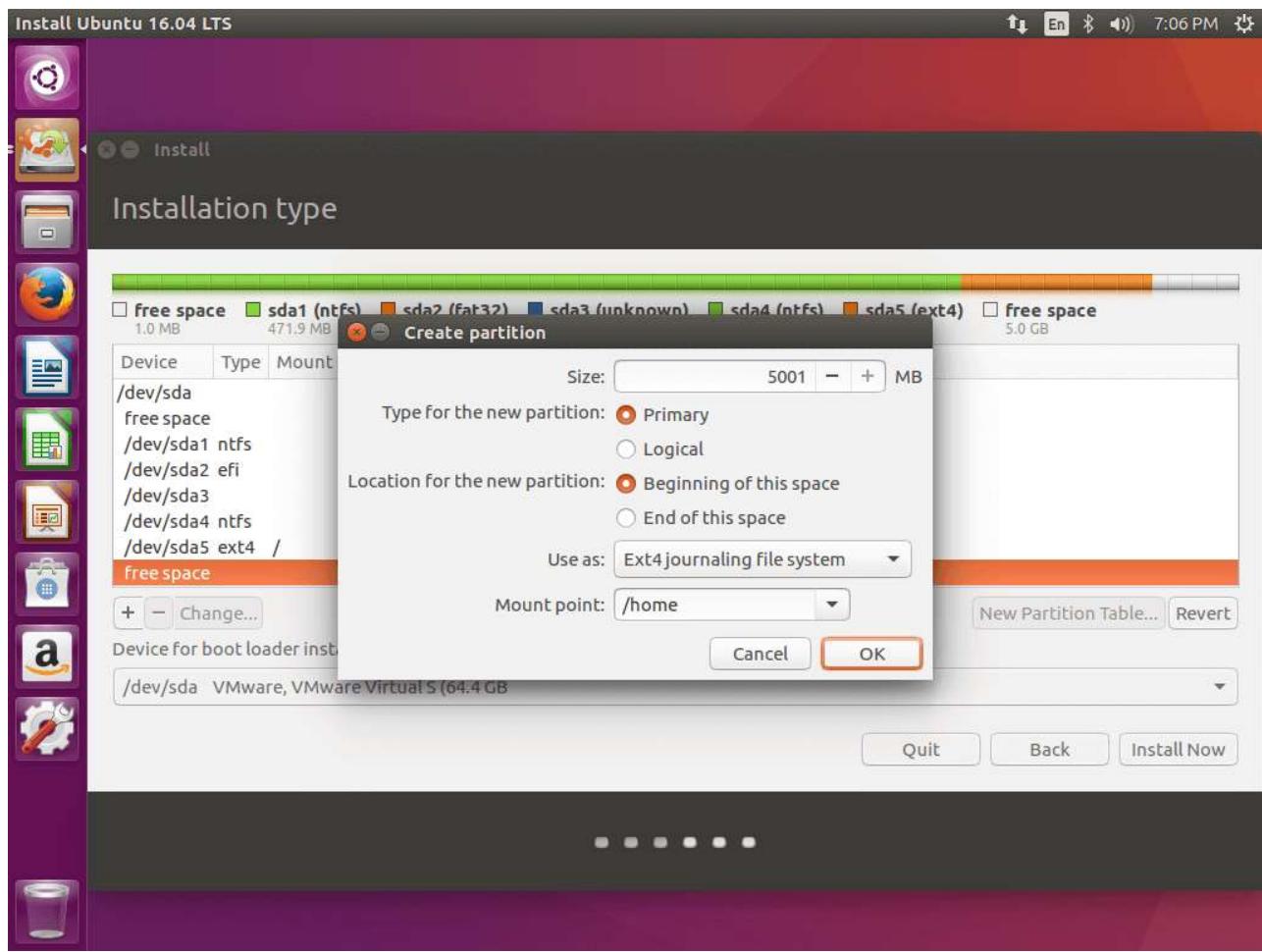
Create Ubuntu 16.04 Partitions



Create Root Partition for Ubuntu 16.04

Create the `home` partition using the same steps as above. Use all the available free space left for home partition size. The partition settings should look like this:

1. Size = all remaining free space
2. Type for the new partition = **Primary**
3. Location for the new partition = **Beginning**
4. Use as = **EXT4** journaling file system
5. Mount point = **/home**

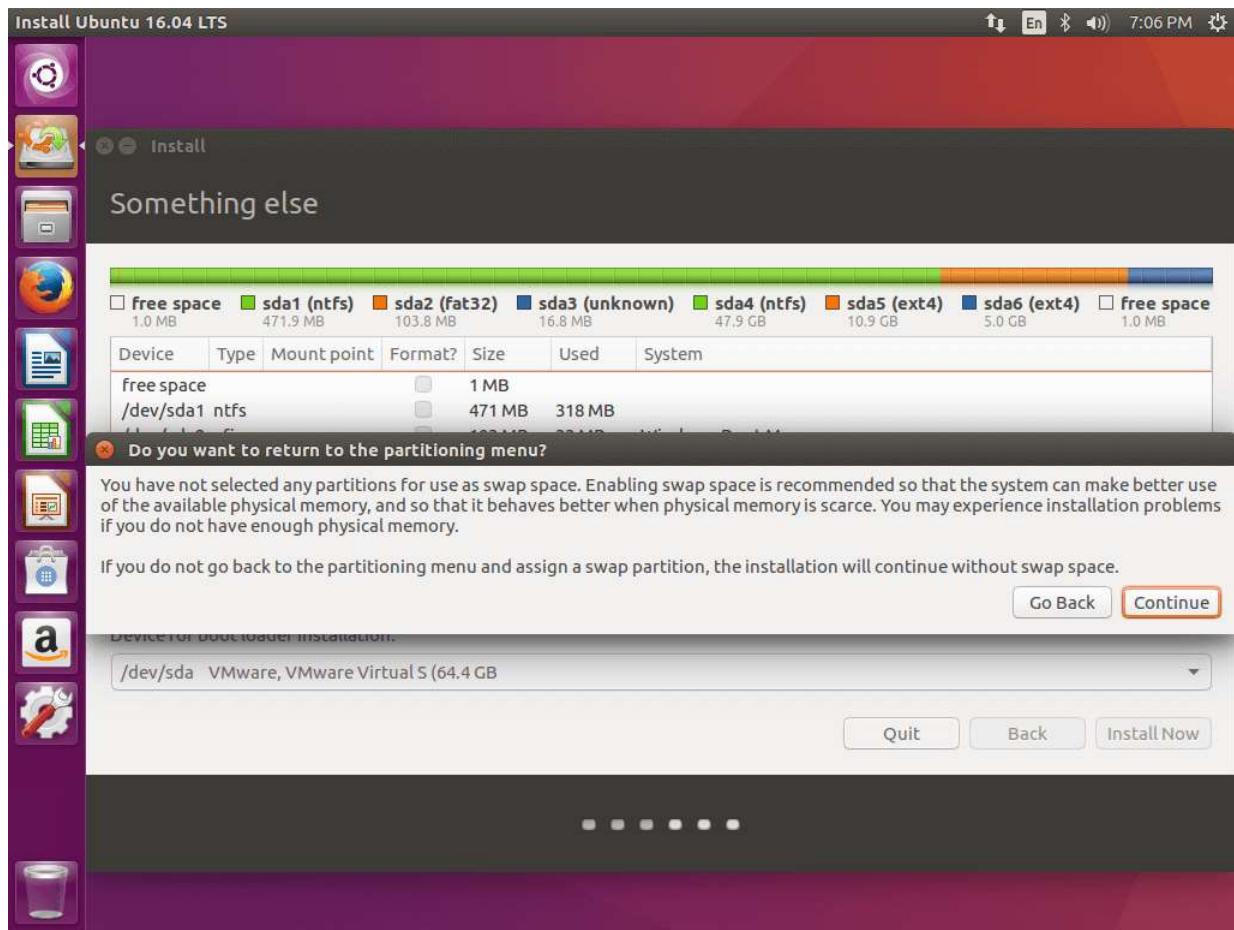


Create Home Partition for Ubuntu 16.04

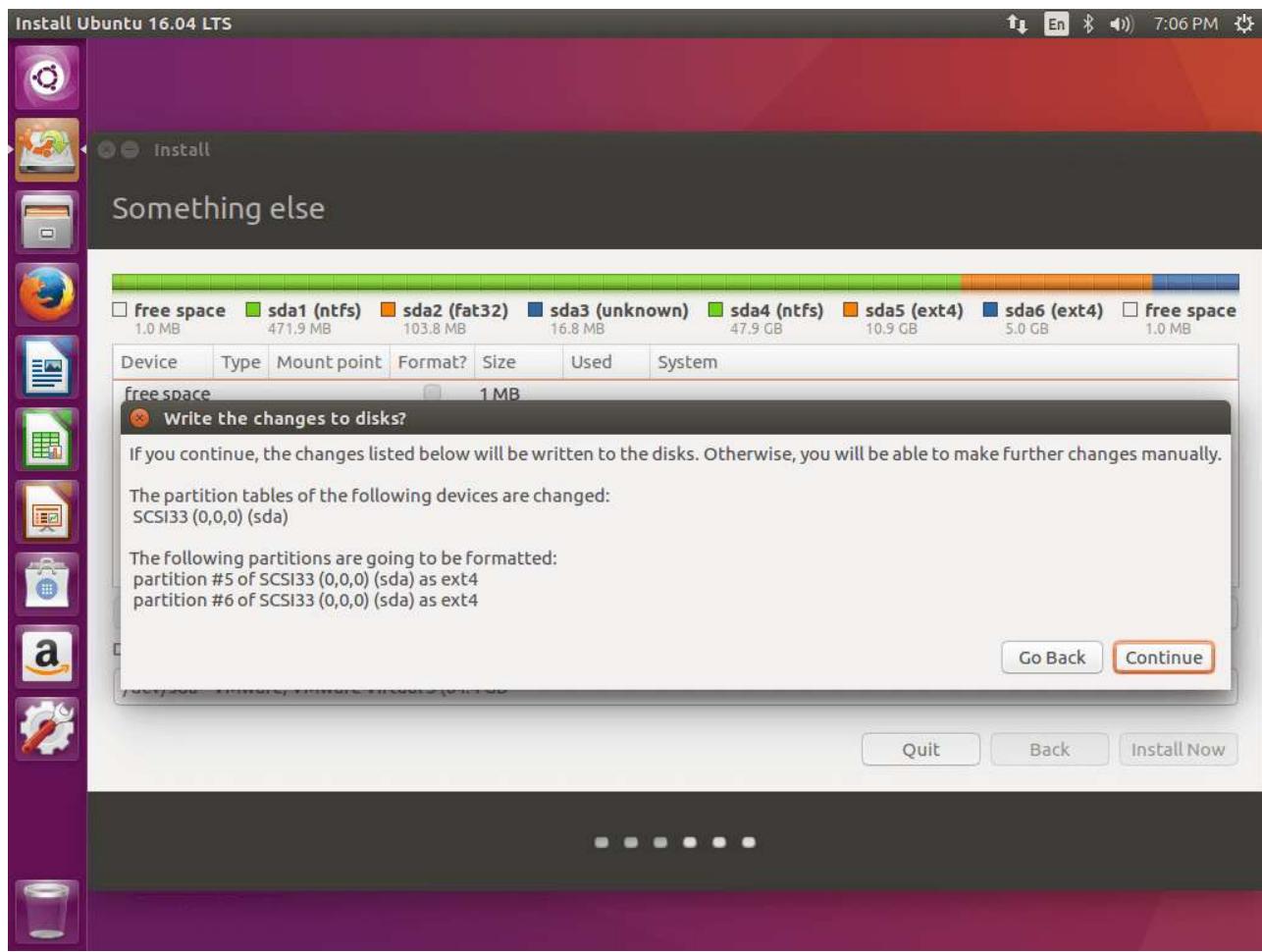
9. When finished, hit the **Install Now** button in order to apply changes to disk and start the installation process.

A pop-up window should appear to inform you about **swap** space. Ignore the alert by pressing on **Continue** button.

Next a new pop-up window will ask you if you agree with committing changes to disk. Hit **Continue** to write changes to disk and the installation process will now start.

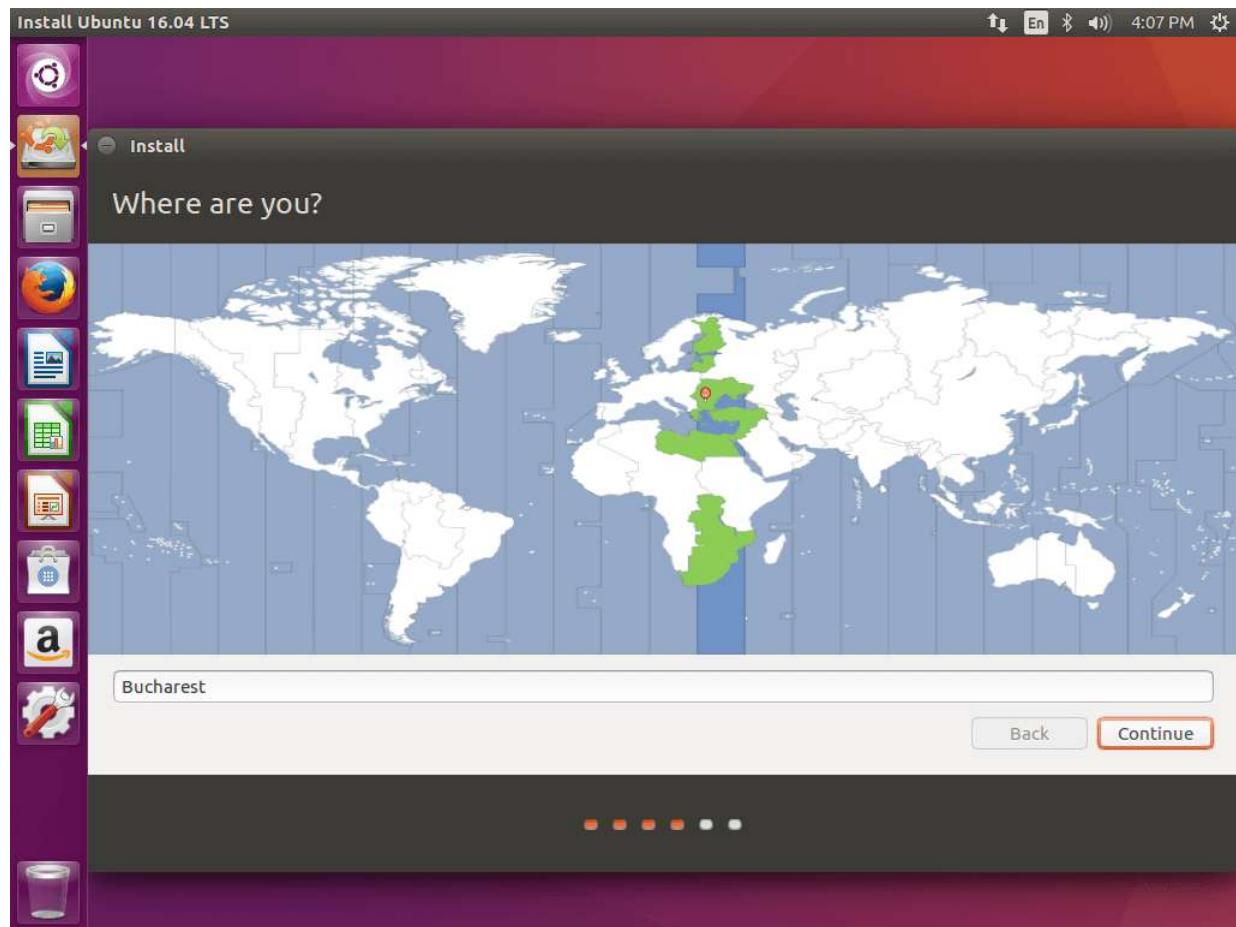


Confirm Partition Changes



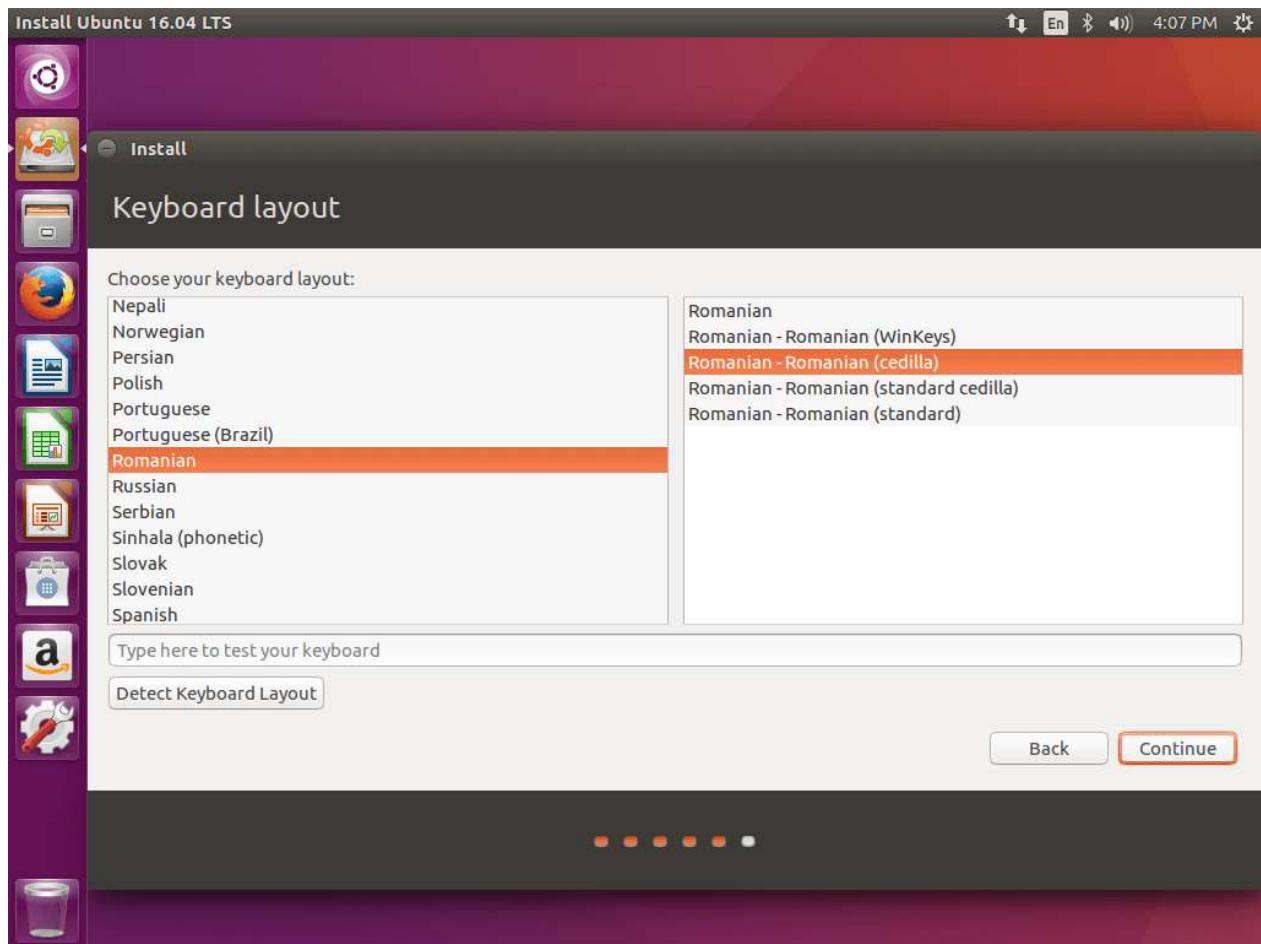
Confirm Write Changes to Disk

- 10.** On the next screen adjust your machine physical location by selecting a city nearby from the map. When done hit **Continue** to move ahead.



Select Your City Location

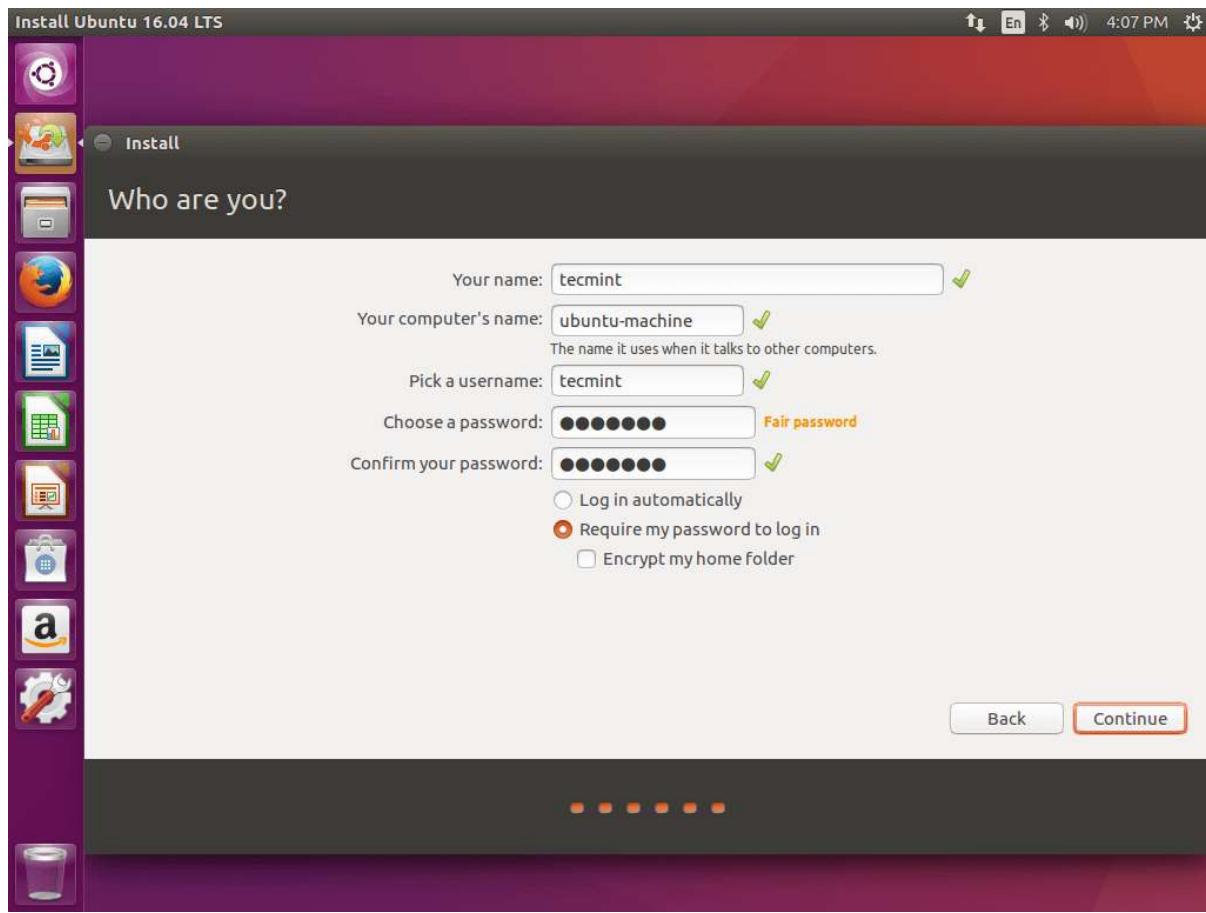
11. Next, select your **keyboard** layout and click on **Continue** button.



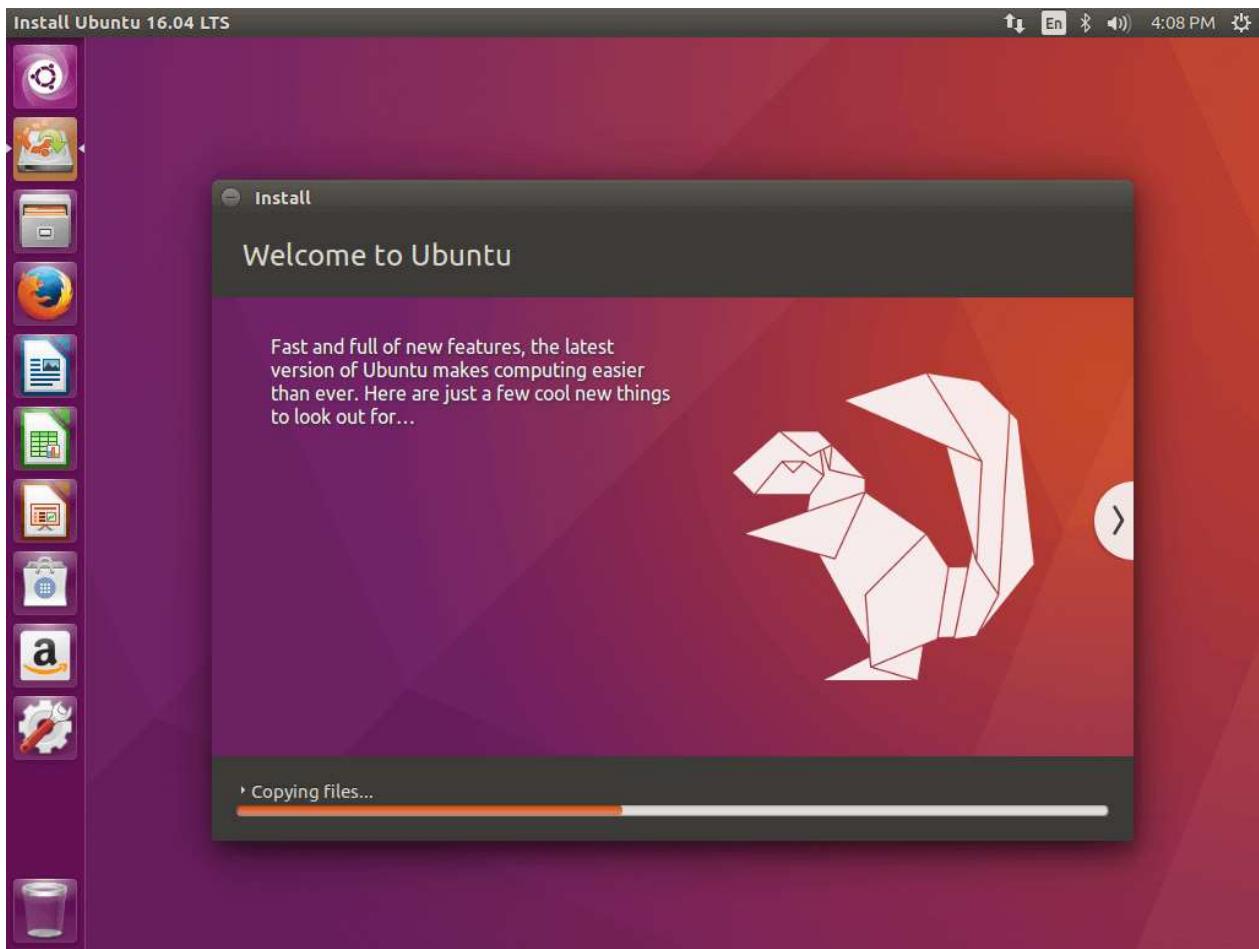
Select Keyboard Layout

12. Pick up a username and password for your administrative **sudo** account, enter a descriptive name for your computer and hit **Continue** to finalize the installation.

This are all the settings required for customizing **Ubuntu 16.04** installation. From here on the installation process will run automatically until it reaches the end.



Create User Account for Ubuntu 16.04

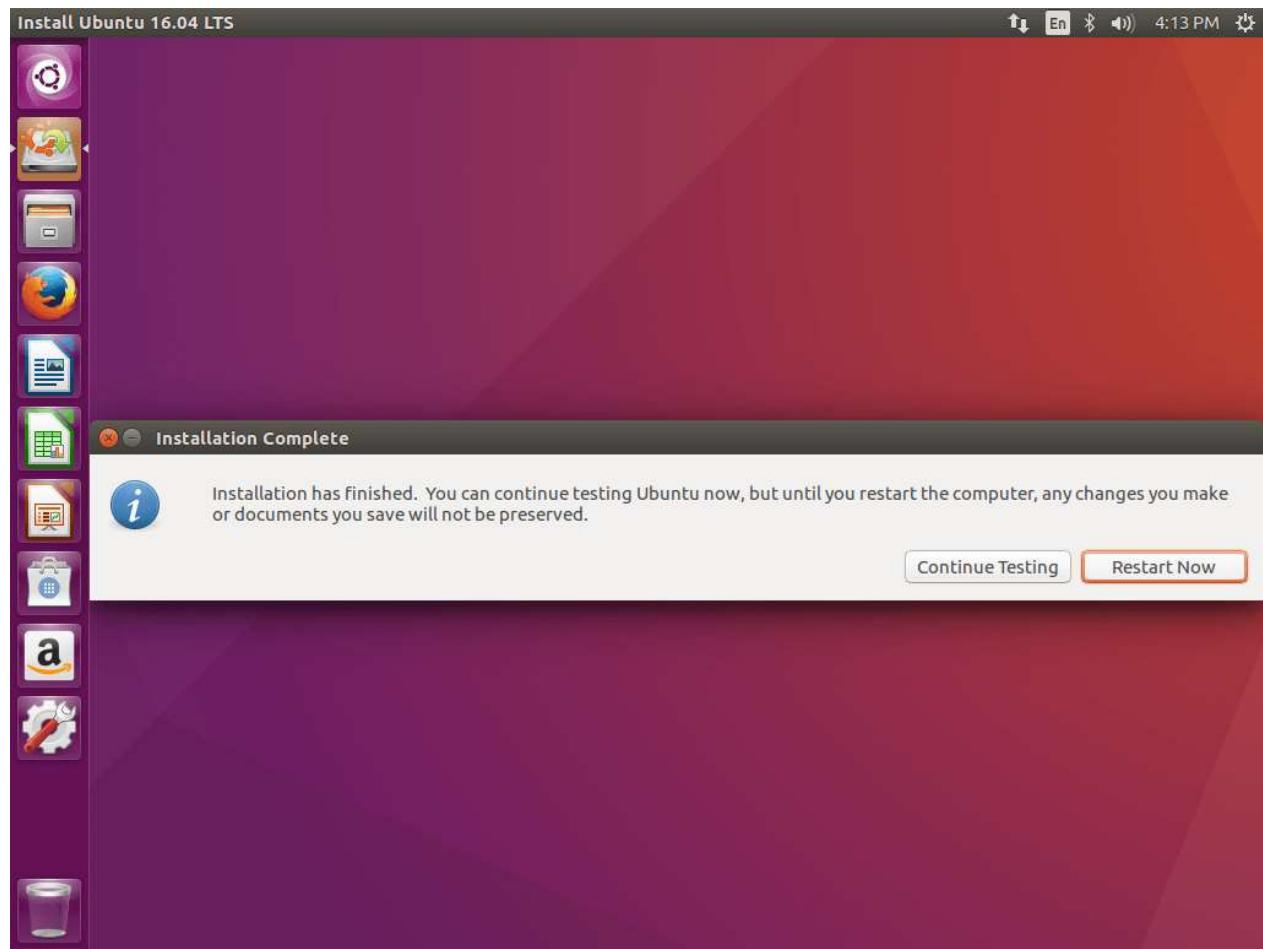


Ubuntu 16.04 Installation Process

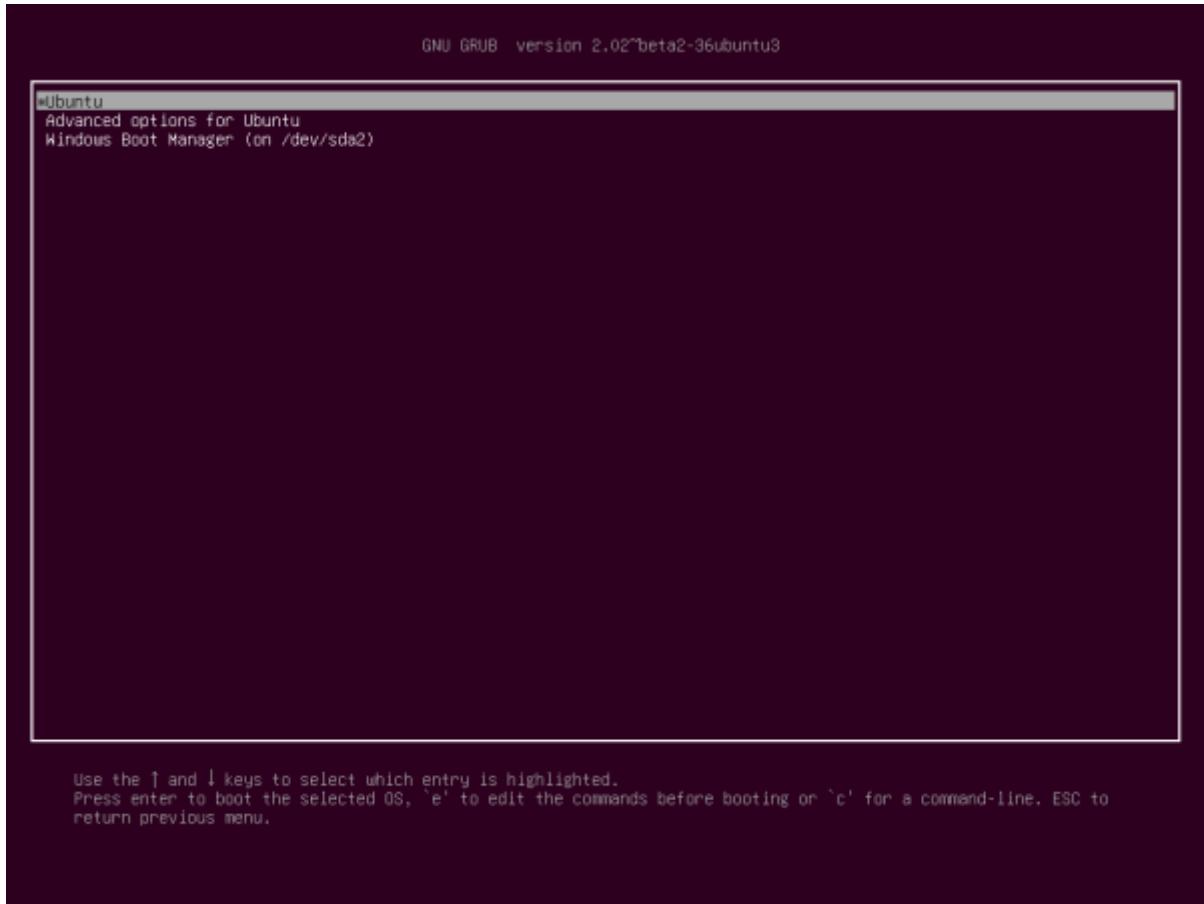
13. After the installation process reaches its end hit on **Restart Now** button in order to complete the installation.

The machine will reboot into the **Grub** menu, where for ten seconds, you will be presented to choose what OS you wish to use further: **Ubuntu 16.04** or **Microsoft Windows**.

Ubuntu is designated as default OS to boot from. Thus, just press **Enter** key or wait for those **10** seconds timeout to drain.

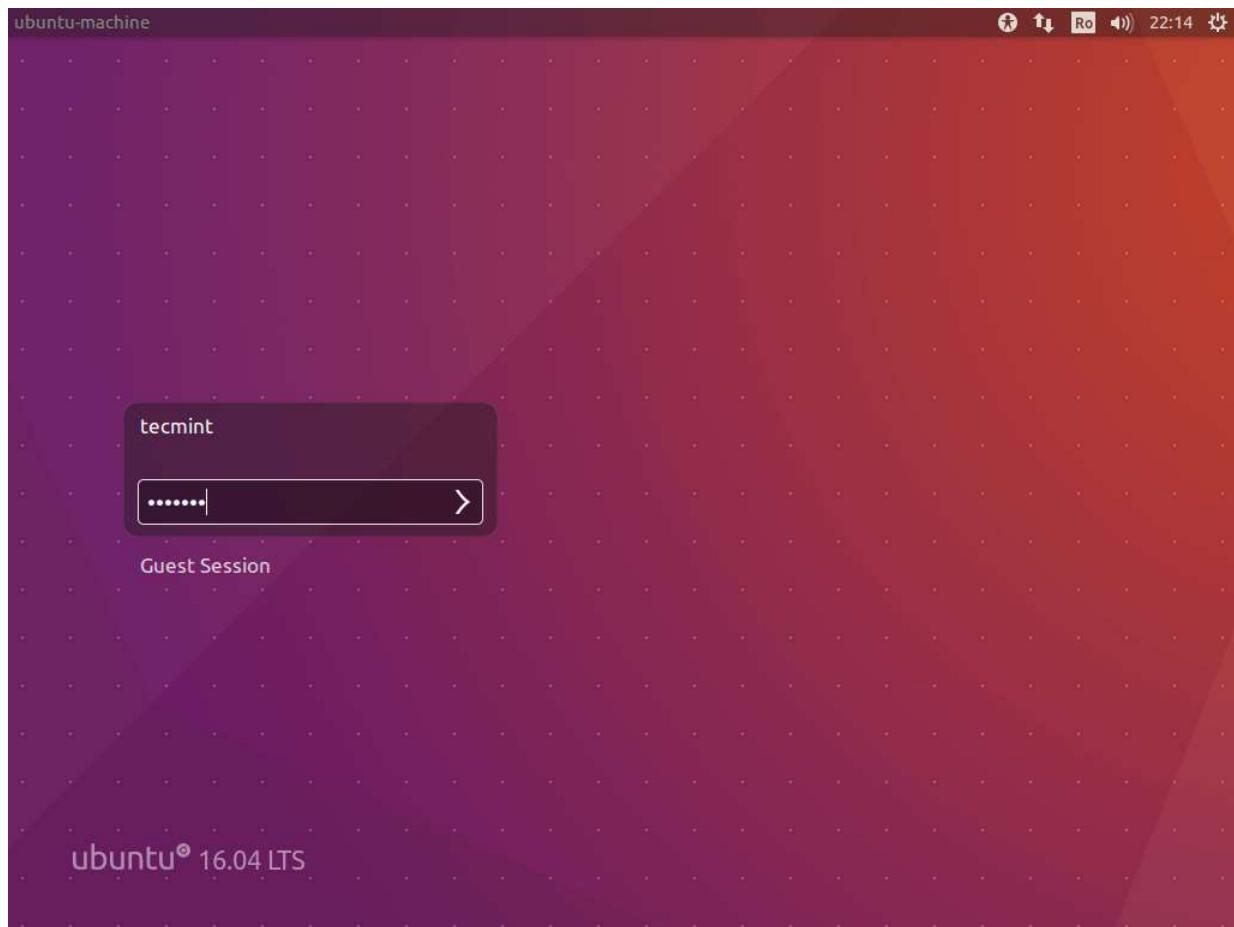


Ubuntu 16.04 Installation Completed

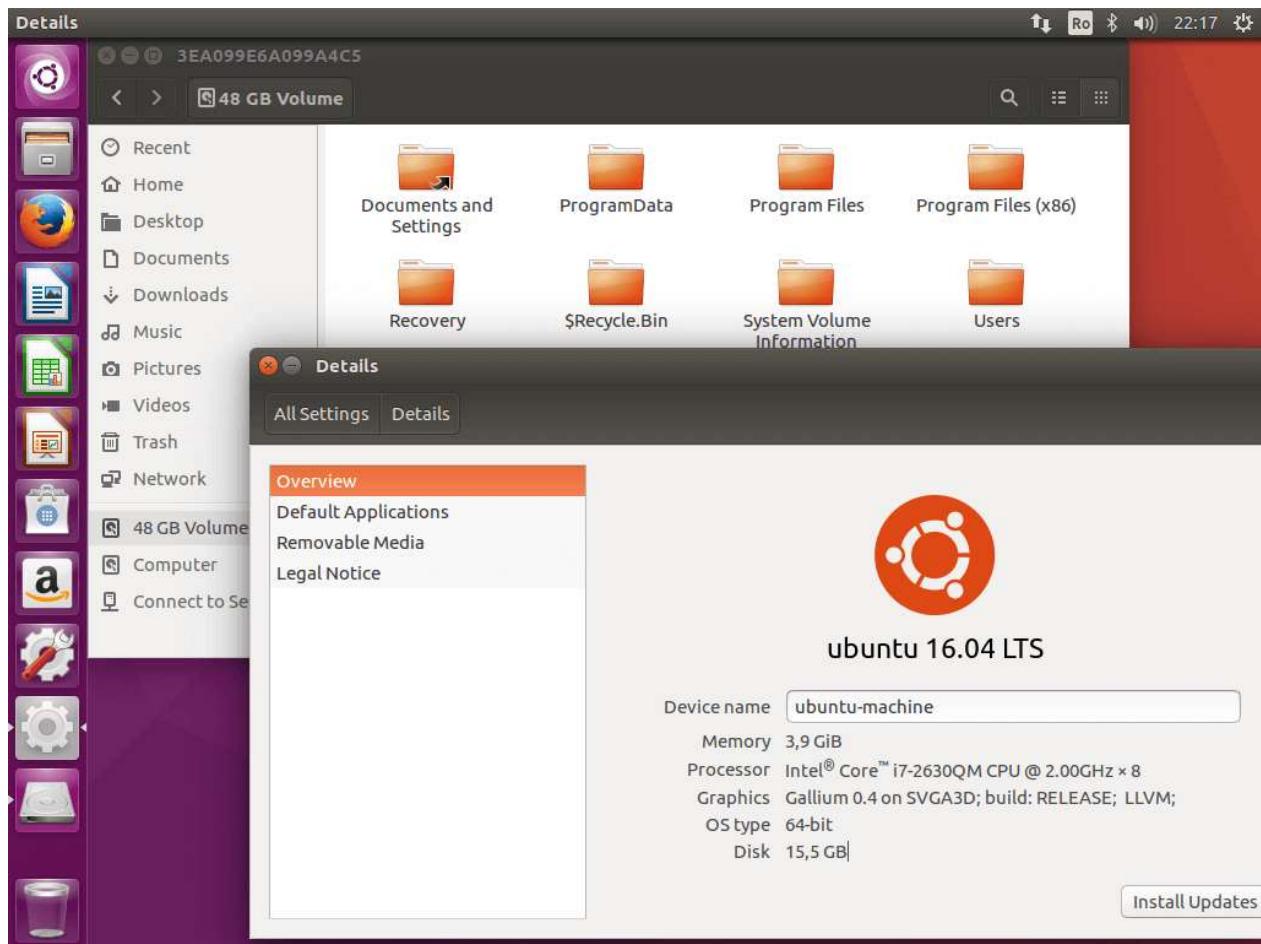


Grub Menu Select Ubuntu or Windows to Boot

- 14.** After Ubuntu finishes loading, login with the credentials created during the installation process and enjoy it. **Ubuntu 16.04** provides **NTFS** file system support automatically so you can access the files from Windows partitions just by clicking on the **Windows** volume.



Ubuntu 16.04 Login



Access Windows Partitions from Ubuntu 16.04

That's it! In case you need to switch back to **Windows**, just reboot the computer and select **Windows** from the **Grub** menu.

If you want to install some additional software packages and customize **Ubuntu 16.04**, then read our article [Top 7 Things to Do After Ubuntu 16.04 Installation](#).

How to Find Difference Between Two Directories Using Diff and Meld Tools

In an earlier article, we reviewed [9 best file comparison and difference \(Diff\) tools for Linux](#) and in this article, we will describe how to find the difference between two directories in Linux.

Normally, to [compare two files in Linux](#), we use the **diff** – a simple and original Unix command-line tool that shows you the difference between two computer files; compares files line by line and it is easy to use, comes with pre-installed on most if not all Linux distributions.

The question is how do we get the difference between two directories in Linux? Here, we want to know what files/subdirectories are common in the two directories, those that are present in one directory but not in the other.

The conventional syntax for running diff is as follows:

```
$ diff [OPTION]... FILES  
$ diff options dir1 dir2
```

By default, its output is ordered alphabetically by file/subdirectory name as shown in the screenshot below. In this command, the **-q** switch tells diff to report only when files differ.

```
$ diff -q directory-1/ directory-2/  
tecmint@TecMint ~/Testing-Linux-Tools $ diff -q directory-1/ directory-2/  
Only in directory-2/: Firefox-for-Linux.png  
Only in directory-2/: Flatpak-for-Linux.png  
Only in directory-2/: Flat-Plat-Theme-Gnome-Shell.png  
Only in directory-2/: Flat-Remix-Theme.png  
Only in directory-1/: Whatever-Icon-Tray.png  
Only in directory-1/: Wickr-Messenger.png  
tecmint@TecMint ~/Testing-Linux-Tools $ _
```

Difference Between Two Directories

Again diff doesn't go into the subdirectories, but we can use the **-r** switch to read the subdirectories as well like this.

```
$ diff -qr directory-1/ directory-2/
```

Using Meld Visual Diff and Merge Tool

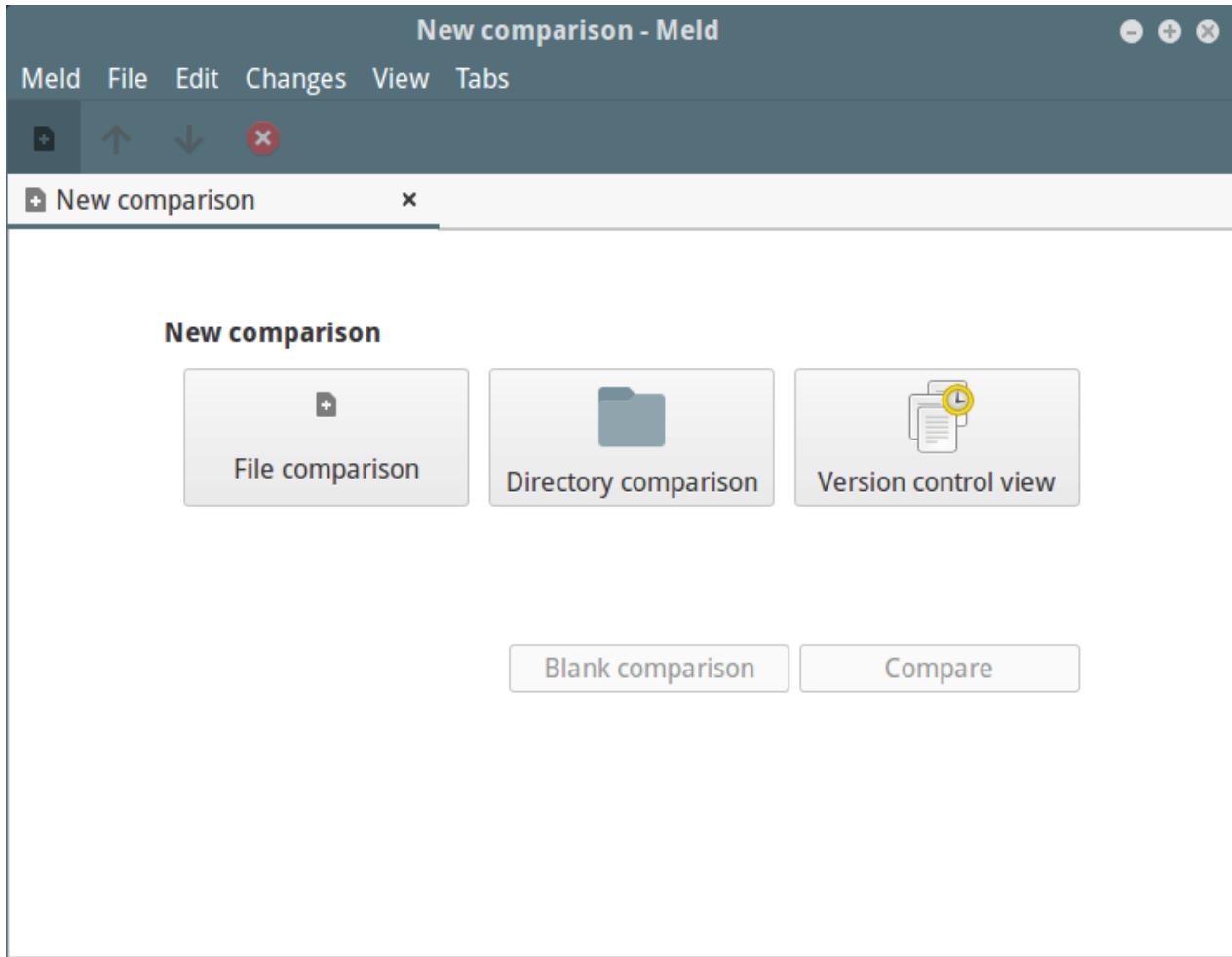
There is a cool graphical option called **meld** (a visual diff and merge tool for the GNOME Desktop) for those who enjoy using the mouse, you can install it as follows.

```
$ sudo apt install meld [Debian/Ubuntu systems]
```

```
$ sudo yum install meld [RHEL/CentOS systems]
$ sudo dnf install meld [Fedora 22+]
```

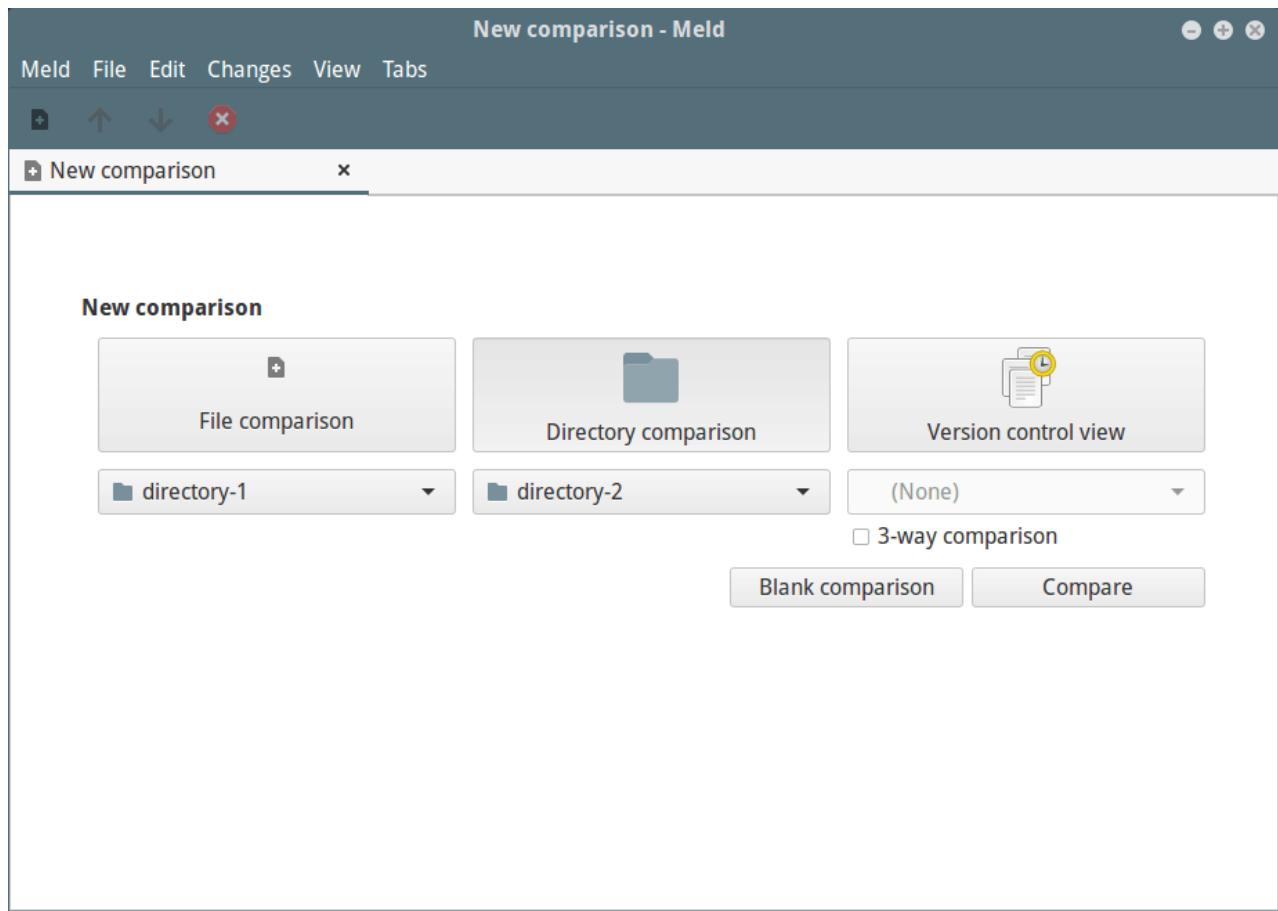
Once you have installed it, search for “**meld**” in the **Ubuntu Dash or Linux Mint Menu**, in **Activities Overview** in Fedora or CentOS desktop and launch it.

You will see the **Meld** interface below, where you can choose file or directory comparison as well as version control view. Click on directory comparison and move to the next interface.



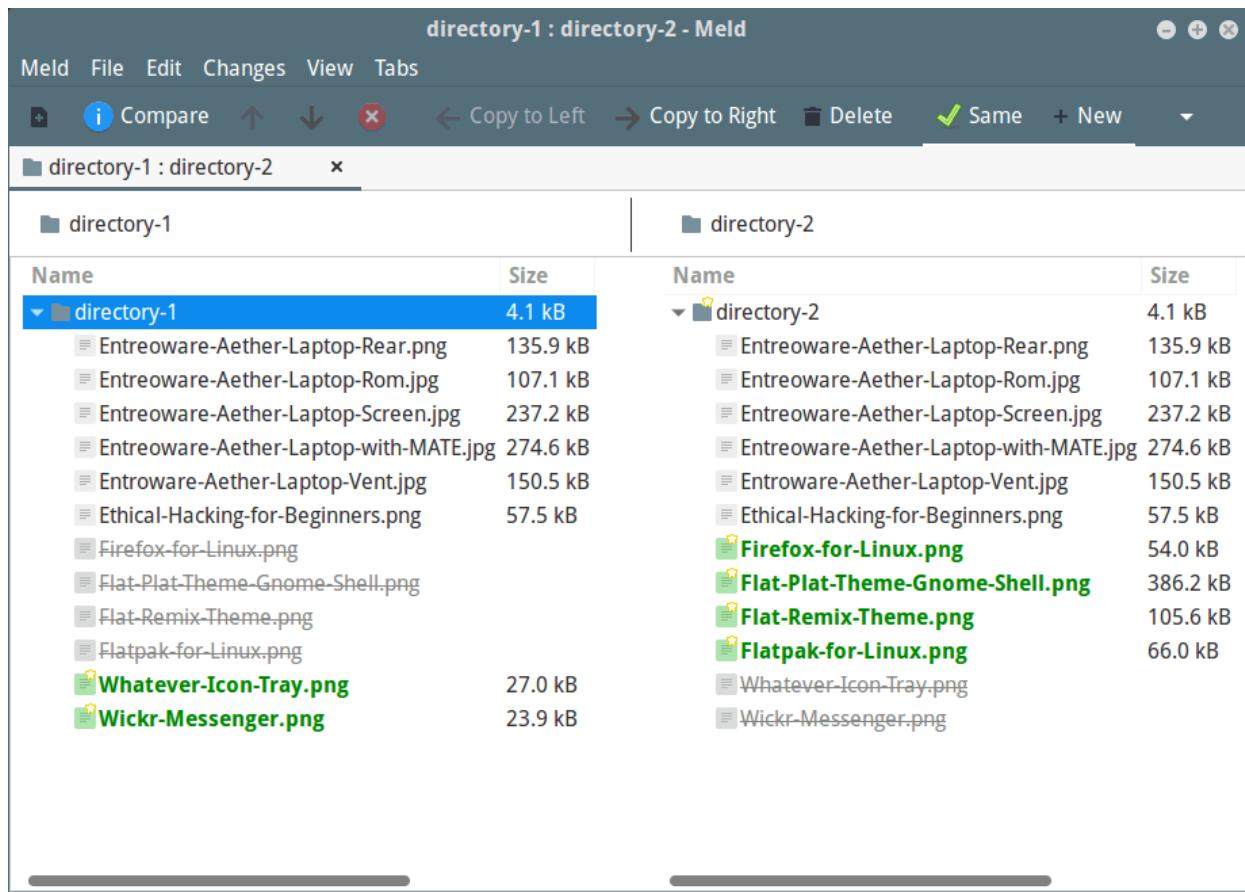
Meld Comparison Tool

Select the directories you want to compare, note that you can add a third directory by checking the option “**3-way Comparison**”.



Select Comparison Directories

Once you selected the directories, click on “**Compare**”.



Listing Difference Between Directories

In this article, we described how to find the difference between two directories in Linux. If you know any other commandline or gui way don't forget to share your thoughts to this article via the comment section below.

10 Useful ‘locate’ Command Practical Examples for Linux Newbies

One of most obnoxious experiences that most new users of the Linux platform usually face is the inability to find the simplest and yet more effective means of looking up files on their system.

Linux, like almost any other operating system, utilizes several mechanisms to answer search queries for users. Two of the most popular file searching utilities accessible to users are called [find](#) and [locate](#).

Now, it is important to note that both search processes work extremely well but nonetheless, the center of this article will be more on the **locate** utility, which is the more convenient of the two as it uses more efficient ways to quickly process queries inputted by the users.

The **locate** utility works better and faster than its **find** counterpart because instead of searching the file system when a file search is initiated – Something find does – locate would look through a database. This database contains bits and parts of files and their corresponding paths on your system.

Here are ten simple locate commands to set you up in becoming more productive with your Linux machine.

1. Using locate Command

Firing locate command to look for a file is pretty easy and straightforward. All you need to do is type:

```
$ locate LAMP-Setup.odt
/home/tecmint/LAMP-Setup.odt
/home/tecmint/TecMint.com/LAMP-Setup.odt
```

2. Limit Search Queries to a Specific Number

You can limit your search returns to a required number to avoid redundancy with your search results using the **-n** command.

For example, if you want just **20** results from your queries, you can type the following command:

```
$ locate "*.*html" -n 20
/home/tecmint/.config/google-
chrome/Default/Extensions/aapocclcgogkmnckokdopfmhonfmgoek/0.9_0/main.html
/home/tecmint/.config/google-
chrome/Default/Extensions/aohghmighlieiainnegkcijnfilokake/0.9_0/main.html
/home/tecmint/.config/google-
chrome/Default/Extensions/felcaaldnbndncclmgdcncolepebgiejap/1.1_0/main.html
```

```
/home/tecmint/.config/google-
chrome/Default/Extensions/kbfnbcaep1bcioakkpcpgfkobkghlhen/14.752.848_0/forge
.html
/home/tecmint/.config/google-
chrome/Default/Extensions/kbfnbcaep1bcioakkpcpgfkobkghlhen/14.752.848_0/src/p
opup.html
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/additiona
l-feature.html
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/background
d.html
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/edit.html
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/help.html
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/options.h
tml
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/popup.htm
l
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/purchase.
html
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/upload.ht
ml
/home/tecmint/.config/google-
chrome/Default/Extensions/nlipoenfbbikpbjkfpfillcgkoblgpmj/3.9.16_0/oauth2/oa
uth2.html
/home/tecmint/.config/google-
chrome/Default/Extensions/nmmhkkegccagldgiimedpiccmgmieda/1.0.0.2_0/html/cra
w_window.html
/home/tecmint/.config/google-
chrome/Default/Extensions/pkedcjkdefgpdelpbcmbmeomcjbeemfm/5516.1005.0.3_0/ca
st_route_details.html
/home/tecmint/.config/google-
chrome/Default/Extensions/pkedcjkdefgpdelpbcmbmeomcjbeemfm/5516.1005.0.3_0/fe
edback.html
/home/tecmint/.config/google-
chrome/Default/Extensions/pkedcjkdefgpdelpbcmbmeomcjbeemfm/5516.1005.0.3_0/ca
st_setup/devices.html
/home/tecmint/.config/google-
chrome/Default/Extensions/pkedcjkdefgpdelpbcmbmeomcjbeemfm/5516.1005.0.3_0/ca
st_setup/index.html
/home/tecmint/.config/google-
chrome/Default/Extensions/pkedcjkdefgpdelpbcmbmeomcjbeemfm/5516.1005.0.3_0/ca
st_setup/offers.html
```

The results will show the first 20 files that end with .html.

3. Display The Number of Matching Entries

If you want to display the count of all matching entries of file “**tecmint**”, use the **locate -c** command.

```
$ locate -c [tecmint]*  
1550
```

4. Ignore Case Sensitive Locate Outputs

By default, **locate** is configured to process queries in a case sensitive manner meaning **TEXT.TXT** will point you to a different result than **text.txt**.

To have **locate** command ignore case sensitivity and show results for both uppercase and lowercase queries, input commands with the **-i** option.

```
$ locate -i *text.txt*  
/home/tecmint/TEXT.txt  
/home/tecmint/text.txt
```

5. Refresh mlocate Database

Since **locate** command relies on a database called **mlocate**. The said database needs to be updated regularly for the command utility to work efficiently.

To update the **mlocate** database, you use a utility called **updatedb**. It should be noted that you will need superuser privileges for this to work properly, as it needs to be executed as root or sudo privileges.

```
$ sudo updatedb
```

6. Display Only Files Present in Your System

When you have an updated **mlocate** database**, **locate** command still produces results of files whose physical copies are deleted from your system.

To avoid seeing results of files not present in your machine at the time of punching in the command, you will need to use the **locate-e** command. The process searches your system to verify the existence of the file you’re looking for even if it is still present in your **mlocate.db**.

```
$ locate -i -e *text.txt*  
/home/tecmint/text.txt
```

7. Separate Output Entries Without New Line

locate command’s default separator is the newline (\n) character. But if you prefer to use a different separator like the **ASCII NUL**, you can do so using the **-0** command line option.

```
$ locate -i -0 *text.txt*
/home/tecmint/TEXT.txt/home/tecmint/text.txt
```

8. Review Your Locate Database

If you're in doubt as to the current status of your **mlocate.db**, you can easily view the locate database statistics by using the **-s** command.

```
$ locate -s
Database /var/lib/mlocate/mlocate.db:
32,246 directories
4,18,850 files
2,92,36,692 bytes in file names
1,13,64,319 bytes used to store database
```

9. Suppress Error Messages in Locate

Constantly trying to access your locate database does sometimes yield unnecessary error messages stating that you do not have the required privileges to have root access to the **mlocate.db**, because you're only a normal user and not the required Superuser.

To completely do away with these message, use the **-q** command.

```
$ locate "\*.dat" -q*
```

10. Choose a Different mlocate Location

If you're inputting queries looking for results not present in the default **mlocate** database and want answers from a different **mlocate.db** located somewhere else in your system, you can point the locate command to a different **mlocate** database at a different part of your system with the **-d** command.

```
$ locate -d <new db path> <filename>
```

locate command might seem like one of those utilities that does everything you asked it to do without much of a hustle but in truth, in order for the process to keep its efficiency, the **mlocate.db** needs to be fed with information every now and then. Failure to do so might render the program a bit useless.

30 Useful Linux Commands for System Administrators

In this article we are going to review some of the useful and frequently used **Linux** or **Unix** commands for **Linux System Administrators** that are used in their daily life. This is not a complete but it's a compact list of commands to refer when needed. Let us start one by one how we can use those commands with examples.



30 Useful Linux System Administration Commands

1. Uptime Command

In Linux **uptime** command shows since how long your system is running and the number of users are currently logged in and also displays load average for **1,5** and **15** minutes intervals.

```
# uptime
08:16:26 up 22 min,  1 user,  load average: 0.00, 0.03, 0.22
```

Check Uptime Version

Uptime command don't have other options other than **uptime** and **version**. It gives information only in **hours:mins** if it less than **1** day.

```
[tecmint@tecmint ~]$ uptime -V
procps version 3.2.8
```

2. W Command

It will displays users currently logged in and their process along-with shows **load averages**. also shows the **login name**, **tty name**, **remote host**, **login time**, **idle time**, **JCPU**, **PCPU**, command and processes.

```
# w
08:27:44 up 34 min, 1 user, load average: 0.00, 0.00, 0.08
USER      TTY      FROM           LOGIN@     IDLE     JCPU      PCPU WHAT
tecmint  pts/0    192.168.50.1    07:59     0.00s   0.29s   0.09s w
```

Available options

1. **-h** : displays no header entries.
2. **-s** : without JCPU and PCPU.
3. **-f** : Removes from field.
4. **-V** : (upper letter) – Shows versions.

3. Users Command

Users command displays currently logged in users. This command don't have other parameters other than help and version.

```
# users
tecmint
```

4. Who Command

who command simply return **user name, date, time** and **host information**. **who** command is similar to **w** command. Unlike **w** command **who** doesn't print what users are doing. Lets illustrate and see the different between **who** and **w** commands.

```
# who
tecmint  pts/0          2012-09-18 07:59 (192.168.50.1)
# w
08:43:58 up 50 min, 1 user, load average: 0.64, 0.18, 0.06
USER      TTY      FROM           LOGIN@     IDLE     JCPU      PCPU WHAT
tecmint  pts/0    192.168.50.1    07:59     0.00s   0.43s   0.10s w
```

Who command Options

1. **-b** : Displays last system reboot date and time.
2. **-r** : Shows current runlet.
3. **-a, -all** : Displays all information in cumulatively.

5. Whoami Command

whoami command print the name of current user. You can also use “**who am i**” command to display the current user. If you are logged in as a root using sudo command “**whoami**” command return **root** as current user. Use “**who am i**” command if you want to know the exact user logged in.

```
# whoami
tecmint
```

6. ls Command

ls command display list of files in human readable format.

```
# ls -l
total 114
dr-xr-xr-x.    2 root root  4096 Sep 18 08:46 bin
dr-xr-xr-x.    5 root root  1024 Sep  8 15:49 boot
```

Sort file as per last modified time.

```
# ls -ltr
total 40
-rw-r--r--. 1 root root  6546 Sep 17 18:42 install.log.syslog
-rw-r--r--. 1 root root 22435 Sep 17 18:45 install.log
-rw-----. 1 root root  1003 Sep 17 18:45 anaconda-ks.cfg
```

For more examples of ls command, please check out our article on [15 Basic ‘ls’ Command Examples in Linux](#).

7. Crontab Command

List schedule jobs for current user with **crontab** command and **-l** option.

```
# crontab -l
00 10 * * * /bin/ls >/ls.txt
```

Edit your **crontab** with **-e** option. In the below example will open schedule jobs in **VI editor**. Make a necessary changes and quit pressing **:wq** keys which saves the setting automatically.

```
# crontab -e
```

For more examples of **Linux Cron Command**, please read our earlier article on [11 Cron Scheduling Task Examples in Linux](#).

8. Less Command

less command allows quickly view file. You can page up and down. Press ‘**q**’ to quit from less window.

```
# less install.log
Installing setup-2.8.14-10.el6.noarch
warning: setup-2.8.14-10.el6.noarch: Header V3 RSA/SHA256 Signature, key ID
c105b9de: NOKEY
Installing filesystem-2.4.30-2.1.el6.i686
Installing ca-certificates-2010.63-3.el6.noarch
Installing xml-common-0.6.3-32.el6.noarch
Installing tzdata-2010l-1.el6.noarch
Installing iso-codes-3.16-2.el6.noarch
```

9. More Command

more command allows quickly view file and shows details in percentage. You can page up and down. Press ‘q’ to quit out from more window.

```
# more install.log
Installing setup-2.8.14-10.el6.noarch
warning: setup-2.8.14-10.el6.noarch: Header V3 RSA/SHA256 Signature, key ID
c105b9de: NOKEY
Installing filesystem-2.4.30-2.1.el6.i686
Installing ca-certificates-2010.63-3.el6.noarch
Installing xml-common-0.6.3-32.el6.noarch
Installing tzdata-2010l-1.el6.noarch
Installing iso-codes-3.16-2.el6.noarch
--More-- (10%)
```

10. CP Command

Copy file from source to destination preserving same mode.

```
# cp -p fileA fileB
```

You will be prompted before overwrite to file.

```
# cp -i fileA fileB
```

11. MV Command

Rename **fileA** to **fileB**. **-i** options prompt before overwrite. Ask for confirmation if exist already.

```
# mv -i fileA fileB
```

12. Cat Command

cat command used to view multiple file at the same time.

```
# cat fileA fileB
```

You combine **more** and **less** command with cat command to view file contain if that doesn't fit in single screen / page.

```
# cat install.log | less
# cat install.log | more
```

For more examples of Linux cat command read our article on [13 Basic Cat Command Examples in Linux](#).

13. Cd command (change directory)

with cd command (change directory) it will goes to **fileA** directory.

```
# cd /fileA
```

14. pwd command (print working directory)

pwd command return with present working directory.

```
# pwd  
/root
```

15. Sort command

Sorting lines of text files in ascending order. with **-r** options will sort in descending order.

```
#sort fileA.txt  
#sort -r fileA.txt
```

16. VI Command

Vi is a most popular text editor available most of the **UNIX-like OS**. Below examples open file in read only with **-R** option. Press '**:q**' to quit from vi window.

```
# vi -R /etc/shadows
```

17. SSH Command (Secure Shell)

SSH command is used to login into remote host. For example the below ssh command will connect to remote host (**192.168.50.2**) using user as **narad**.

```
# ssh narad@192.168.50.2
```

To check the version of ssh use option **-V** (uppercase) shows version of ssh.

```
# ssh -V  
OpenSSH_5.3p1, OpenSSL 1.0.0-fips 29 Mar 2010
```

18. Ftp or sftp Command

ftp or **sftp** command is used to connect to remote ftp host. **ftp** is (**file transfer protocol**) and **sftp** is (**secure file transfer protocol**). For example the below commands will connect to ftp host (**192.168.50.2**).

```
# ftp 192.168.50.2  
# sftp 192.168.50.2
```

Putting multiple files in remote host with **mput** similarly we can do **mget** to download multiple files from remote host.

```
# ftp > mput *.txt  
# ftp > mget *.txt
```

19. Service Command

Service command call script located at **/etc/init.d/** directory and execute the script. There are two ways to start the any service. For example we start the service called **httpd** with service command.

```
# service httpd start  
OR  
# /etc/init.d/httpd start
```

20. Free command

Free command shows **free**, **total** and **swap memory** information in bytes.

```
# free  
total        used         free        shared       buffers       cached  
Mem:    1030800      735944      294856          0      51648      547696  
-/+ buffers/cache:  136600      894200  
Swap:     2064376          0     2064376
```

Free with **-t** options shows **total memory** used and available to use in bytes.

```
# free -t  
total        used         free        shared       buffers       cached  
Mem:    1030800      736096      294704          0      51720      547704  
-/+ buffers/cache:  136672      894128  
Swap:     2064376          0     2064376  
Total:   3095176      736096     2359080
```

21. Top Command

top command displays processor activity of your system and also displays tasks managed by kernel in real-time. It'll show **processor** and **memory** are being used. Use top command with '**u**' option this will display specific User process details as shown below. Press '**O**' (**uppercase letter**) to sort as per desired by you. Press '**q**' to quit from top screen.

```
# top -u tecmint  
top - 11:13:11 up  3:19,  2 users,  load average: 0.00, 0.00, 0.00  
Tasks: 116 total,   1 running, 115 sleeping,   0 stopped,   0 zombie  
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  
0.0%st  
Mem:  1030800k total,   736188k used,   294612k free,   51760k buffers  
Swap: 2064376k total,        0k used,  2064376k free,  547704k cached  
PID USER      PR NI VIRT  RES SHR S %CPU %MEM      TIME+ COMMAND  
1889 tecmint    20  0 11468 1648  920 S  0.0  0.2  0:00.59 sshd  
1890 tecmint    20  0  5124 1668 1416 S  0.0  0.2  0:00.44 bash  
6698 tecmint    20  0 11600 1668  924 S  0.0  0.2  0:01.19 sshd  
6699 tecmint    20  0  5124 1596 1352 S  0.0  0.2  0:00.11 bash
```

For more about top command we've already compiled a list of [12 TOP Command Examples in Linux](#).

22. Tar Command

tar command is used to compress files and folders in Linux. For example the below command will create a archive for **/home** directory with file name as **archive-name.tar**.

```
# tar -cvf archive-name.tar /home
```

To extract tar archive file use the option as follows.

```
# tar -xvf archive-name.tar
```

To understand more about **tar command** we've created a complete **how-to guide** on tar command at [18 Tar Command Examples in Linux](#).

23. Grep Command

grep search for a given string in a file. Only **tecmint** user displays from **/etc/passwd** file. we can use **-i** option for ignoring case sensitive.

```
# grep tecmint /etc/passwd
tecmint:x:500:500::/home/tecmint:/bin/bash
```

24. Find Command

Find command used to search **files**, **strings** and **directories**. The below example of find command search **tecmint** word in '/' partition and return the output.

```
# find / -name tecmint
/var/spool/mail/tecmint
/home/tecmint
/root/home/tecmint
```

For complete guide on **Linux find command** examples fount at [35 Practical Examples of Linux Find Command](#).

25. Lsof Command

lsof mean List of all open files. Below lsof command list of all opened files by user **tecmint**.

```
# lsof -u tecmint
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
sshd 1889 tecmint cwd DIR 253,0 4096 2 /
sshd 1889 tecmint txt REG 253,0 532336 298069 /usr/sbin/sshd
sshd 1889 tecmint DEL REG 253,0 412940
/lib/libc_err.so.2.1
```

```

sshd      1889 tecmint  DEL    REG      253,0        393156 /lib/ld-2.12.so
sshd      1889 tecmint  DEL    REG      253,0        298643
/usr/lib/libcrypto.so.1.0.0
sshd      1889 tecmint  DEL    REG      253,0        393173 /lib/libnsl-
2.12.so
sshd      1889 tecmint  DEL    REG      253,0        412937
/lib/libkrb5support.so.0.1
sshd      1889 tecmint  DEL    REG      253,0        412961 /lib/libplc4.so

```

For more **lsof command examples** visit [10 lsof Command Examples in Linux](#).

26. last command

With **last** command we can watch user's activity in the system. This command can execute normal user also. It will display complete user's info like **terminal, time, date, system reboot or boot and kernel version**. Useful command to troubleshoot.

```
# last
tecmint pts/1      192.168.50.1      Tue Sep 18 08:50  still logged in
tecmint pts/0      192.168.50.1      Tue Sep 18 07:59  still logged in
reboot   system boot 2.6.32-279.el6.i Tue Sep 18 07:54 - 11:38  (03:43)
root     pts/1      192.168.50.1      Sun Sep 16 10:40 - down  (03:53)
root     pts/0      :0.0              Sun Sep 16 10:36 - 13:09  (02:32)
root     tty1       :0                Sun Sep 16 10:07 - down  (04:26)
reboot   system boot 2.6.32-279.el6.i Sun Sep 16 09:57 - 14:33  (04:35)
narad    pts/2      192.168.50.1      Thu Sep 13 08:07 - down  (01:15)
```

You can use **last** with **username** to know for specific user's activity as shown below.

```
# last tecmint
tecmint pts/1      192.168.50.1      Tue Sep 18 08:50  still logged in
tecmint pts/0      192.168.50.1      Tue Sep 18 07:59  still logged in
tecmint pts/1      192.168.50.1      Thu Sep 13 08:07 - down  (01:15)
tecmint pts/4      192.168.50.1      Wed Sep 12 10:12 - 12:29  (02:17)
```

27. ps command

ps command displays about processes running in the system. Below example show **init** process only.

```
# ps -ef | grep init
root      1      0  0 07:53 ?          00:00:04 /sbin/init
root    7508  6825  0 11:48 pts/1      00:00:00 grep init
```

28. kill command

Use **kill** command to terminate process. First find process **id** with **ps** command as shown below and kill process with **kill -9** command.

```
# ps -ef | grep init
root      1      0  0 07:53 ?          00:00:04 /sbin/init
```

```
root      7508  6825  0 11:48 pts/1    00:00:00 grep init
# kill -9 7508
```

29. rm command

rm command used to remove or delete a file without prompting for confirmation.

```
# rm filename
```

Using **-i** option to get confirmation before removing it. Using options '**-r**' and '**-f**' will remove the file forcefully without confirmation.

```
# rm -i test.txt
rm: remove regular file `test.txt'?
```

30. mkdir command example.

mkdir command is used to create directories under Linux.

```
# mkdir directoryname
```

This is a handy day to day useable basic commands in Linux / Unix-like operating system.
Kindly share through our comment box if we missed out.

15 Basic ‘ls’ Command Examples in Linux

[ls command](#) is one of the most frequently used command in Linux. I believe **ls** command is the first command you may use when you get into the command prompt of Linux Box.

We use **ls** command daily basis and frequently even though we may not aware and never use all the [ls option available](#). In this article, we'll be discussing basic **ls** command where we have tried to cover as much parameters as possible.



Linux ls Command

1. List Files using ls with no option

ls with no option list files and directories in bare format where we won't be able to view details like file types, size, modified date and time, permission and links etc.

```
# ls
0001.pcap      Desktop    Downloads      index.html  install.log.syslog
Pictures   Templates
anaconda-ks.cfg  Documents  fbcmd_update.php  install.log  Music
Public      Videos
```

2 List Files With option -l

Here, **ls -l** (-l is character not one) shows file or directory, size, modified date and time, file or folder name and owner of file and it's permission.

```
# ls -l
total 176
-rw-r--r--. 1 root root  683 Aug 19 09:59 0001.pcap
-rw-----. 1 root root 1586 Jul 31 02:17 anaconda-ks.cfg
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Desktop
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Documents
drwxr-xr-x. 4 root root 4096 Aug 16 02:55 Downloads
-rw-r--r--. 1 root root 21262 Aug 12 12:42 fbcmd_update.php
-rw-r--r--. 1 root root 46701 Jul 31 09:58 index.html
```

```
-rw-r--r--. 1 root root 48867 Jul 31 02:17 install.log
-rw-r--r--. 1 root root 11439 Jul 31 02:13 install.log.syslog
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Music
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Pictures
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Public
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Templates
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Videos
```

3. View Hidden Files

List all files including hidden file starting with ‘.’.

```
# ls -a
. bashrc Documents .gconfd install.log
.nautilus .pulse-cookie .. .cache Downloads .gnome2
install.log.syslog .netstat.swp .recently-used.xbel
0001.pcap .config .elinks .gnome2_private .kde
.opera .spice-vdagent anaconda-ks.cfg .cshrc .esd_auth .gtk-bookmarks .libreoffice
Pictures .tcshrc .bash_history .dbus .fbcmd .gvfs .local
.pki Templates .bash_logout Desktop fbcmd_update.php .ICEauthority .mozilla
Public Videos .bash_profile .digrc .gconf index.html Music
.pulse .wireshark
```

4. List Files with Human Readable Format with option -lh

With combination of **-lh** option, shows sizes in human readable format.

```
# ls -lh
total 176K
-rw-r--r--. 1 root root 683 Aug 19 09:59 0001.pcap
-rw-----. 1 root root 1.6K Jul 31 02:17 anaconda-ks.cfg
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Desktop
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Documents
drwxr-xr-x. 4 root root 4.0K Aug 16 02:55 Downloads
-rw-r--r--. 1 root root 21K Aug 12 12:42 fbcmd_update.php
-rw-r--r--. 1 root root 46K Jul 31 09:58 index.html
-rw-r--r--. 1 root root 48K Jul 31 02:17 install.log
-rw-r--r--. 1 root root 12K Jul 31 02:13 install.log.syslog
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Music
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Pictures
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Public
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Templates
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Videos
```

5. List Files and Directories with ‘/’ Character at the end

Using **-F** option with **ls** command, will add the ‘/’ Character at the end each directory.

```
# ls -F
0001.pcap      Desktop/    Downloads/        index.html
install.log.syslog  Pictures/   Templates/
anaconda-ks.cfg  Documents/  fbcmd_update.php  install.log  Music/
Public/        Videos/
```

6. List Files in Reverse Order

The following command with **ls -r** option display files and directories in reverse order.

```
# ls -r
Videos      Public      Music           install.log  fbcmd_update.php
Documents   anaconda-ks.cfg
Templates   Pictures    install.log.syslog  index.html  Downloads
Desktop    0001.pcap
```

7. Recursively list Sub-Directories

ls -R option will list very long listing directory trees. See an example of output of the command.

```
# ls -R
total 1384
-rw-----. 1 root      root      33408 Aug  8 17:25 anaconda.log
-rw-----. 1 root      root      30508 Aug  8 17:25 anaconda.program.log
./httpd:
total 132
-rw-r--r-- 1 root      root      0 Aug 19 03:14 access_log
-rw-r--r--. 1 root      root  61916 Aug 10 17:55 access_log-20120812
./lighttpd:
total 68
-rw-r--r-- 1 lighttpd lighttpd  7858 Aug 21 15:26 access.log
-rw-r--r--. 1 lighttpd lighttpd 37531 Aug 17 18:21 access.log-20120819
./nginx:
total 12
-rw-r--r--. 1 root      root      0 Aug 12 03:17 access.log
-rw-r--r--. 1 root      root    390 Aug 12 03:17 access.log-20120812.gz
```

8. Reverse Output Order

With combination of **-ltr** will shows latest modification file or directory date as last.

```
# ls -ltr
total 176
-rw-r--r--. 1 root      root  11439 Jul 31 02:13 install.log.syslog
-rw-r--r--. 1 root      root  48867 Jul 31 02:17 install.log
-rw-----. 1 root      root   1586 Jul 31 02:17 anaconda-ks.cfg
drwxr-xr-x. 2 root      root  4096 Jul 31 02:48 Desktop
drwxr-xr-x. 2 root      root  4096 Jul 31 02:48 Videos
drwxr-xr-x. 2 root      root  4096 Jul 31 02:48 Templates
drwxr-xr-x. 2 root      root  4096 Jul 31 02:48 Public
drwxr-xr-x. 2 root      root  4096 Jul 31 02:48 Pictures
drwxr-xr-x. 2 root      root  4096 Jul 31 02:48 Music
drwxr-xr-x. 2 root      root  4096 Jul 31 02:48 Documents
```

```
-rw-r--r--. 1 root root 46701 Jul 31 09:58 index.html
-rw-r--r--. 1 root root 21262 Aug 12 12:42 fbcmd_update.php
drwxr-xr-x. 4 root root 4096 Aug 16 02:55 Downloads
-rw-r--r--. 1 root root 683 Aug 19 09:59 0001.pcap
```

9. Sort Files by File Size

With combination of **-ls** displays file size in order, will display big in size first.

```
# ls -ls
total 176
-rw-r--r--. 1 root root 48867 Jul 31 02:17 install.log
-rw-r--r--. 1 root root 46701 Jul 31 09:58 index.html
-rw-r--r--. 1 root root 21262 Aug 12 12:42 fbcmd_update.php
-rw-r--r--. 1 root root 11439 Jul 31 02:13 install.log.syslog
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Desktop
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Documents
drwxr-xr-x. 4 root root 4096 Aug 16 02:55 Downloads
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Music
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Pictures
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Public
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Templates
drwxr-xr-x. 2 root root 4096 Jul 31 02:48 Videos
-rw-----. 1 root root 1586 Jul 31 02:17 anaconda-ks.cfg
-rw-r--r--. 1 root root 683 Aug 19 09:59 0001.pcap
```

10. Display Inode number of File or Directory

We can see some number printed before file / directory name. With **-i** options list file / directory with inode number.

```
# ls -i
20112 0001.pcap      23610 Documents          23793 index.html
23611 Music          23597 Templates          22 install.log
23564 anaconda-ks.cfg 23595 Downloads         35 install.log.syslog
23612 Pictures        23613 Videos
23594 Desktop         23585 fbcmd_update.php
23601 Public
```

11. Shows version of ls command

Check version of ls command.

```
# ls --version
ls (GNU coreutils) 8.4
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Written by Richard M. Stallman and David MacKenzie.
```

12. Show Help Page

List help page of ls command with their option.

```
# ls --help
Usage: ls [OPTION]... [FILE]...
```

13. List Directory Information

With **ls -l** command list files under directory **/tmp**. Wherein with **-ld** parameters displays information of **/tmp** directory.

```
# ls -l /tmp
total 408
drwx----- 2 narad narad 4096 Aug  2 02:00 CRX_75DAF8CB7768
-r----- 1 root  root 384683 Aug  4 12:28 htop-1.0.1.tar.gz
drwx----- 2 root  root 4096 Aug  4 11:20 keyring-6Mfjnk
drwx----- 2 root  root 4096 Aug 16 01:33 keyring-pioZJr
drwx----- 2 gdm   gdm  4096 Aug 21 11:26 orbit-gdm
drwx----- 2 root  root 4096 Aug 19 08:41 pulse-g16o4ZdxQVrX
drwx----- 2 narad narad 4096 Aug  4 08:16 pulse-UDH76ExwUVoU
drwx----- 2 gdm   gdm  4096 Aug 21 11:26 pulse-wJtcweUCtvhn
-rw----- 1 root  root 300  Aug 16 03:34 yum_save_tx-2012-08-16-03-
34LJTAal.yumtx
# ls -ld /tmp/
drwxrwxrwt. 13 root root 4096 Aug 21 12:48 /tmp/
```

14. Display UID and GID of Files

To display **UID** and **GID** of files and directories. use option **-n** with ls command.

```
# ls -n
total 36
drwxr-xr-x. 2 500 500 4096 Aug  2 01:52 Downloads
drwxr-xr-x. 2 500 500 4096 Aug  2 01:52 Music
drwxr-xr-x. 2 500 500 4096 Aug  2 01:52 Pictures
-rw-rw-r--. 1 500 500    12 Aug 21 13:06 tmp.txt
drwxr-xr-x. 2 500 500 4096 Aug  2 01:52 Videos
```

15. ls command and it's Aliases

We have made alias for **ls** command, when we execute ls command it'll take **-l** option by default and display long listing as mentioned earlier.

```
# alias ls="ls -l"
```

Note: We can see number of alias available in your system with below alias command and same can be unalias as shown below example.

```
# alias
alias cp='cp -i'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
```

```
alias ls='ls --color=auto'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --
show-tilde'
```

To remove an alias previously defined, just use the unalias command.

```
# unalias ls
```

11 Cron Scheduling Task Examples in Linux

by [Ravi Saive](#) | Published: September 13, 2012 | Last Updated: May 25, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this article we are going to review and see how we can schedule and run tasks in the background automatically at regular intervals using **Crontab** command. Dealing a frequent job manually is a daunting task for system administrator. Such process can be schedule and run automatically in the background without human intervene using cron daemon in Linux or Unix-like operating system.

For instance, you can automate process like **backup**, **schedule updates** and **synchronization of files** and many more. **Cron** is a daemon to run schedule tasks. Cron wakes up every minute and checks schedule tasks in crontable. **Crontab (CRON TABLE)** is a table where we can schedule such kind of repeated tasks.

Tips: Each user can have their own crontab to create, modify and delete tasks. By default **cron** is enable to users, however we can restrict adding entry in **/etc/cron.deny** file.



11 Cron Command Examples in Linux

Crontab file consists of command per line and have six fields actually and separated either of space or tab. The beginning five fields represent time to run tasks and last field is for command.

1. Minute (hold values between **0-59**)
2. Hour (hold values between **0-23**)
3. Day of Month (hold values between **1-31**)
4. Month of the year (hold values between **1-12** or **Jan-Dec**, you can use first three letters of each month's name i.e **Jan** or **Jun.**)
5. Day of week (hold values between **0-6** or **Sun-Sat**, Here also you can use first three letters of each day's name i.e **Sun** or **Wed.**)

6. Command

1. List Crontab Entries

List or manage the task with crontab command with **-l** option for current user.

```
# crontab -l  
00 10 * * * /bin/ls >/ls.txt
```

2. Edit Crontab Entries

To edit crontab entry, use **-e** option as shown below. In the below example will open schedule jobs in **VI** editor. Make a necessary changes and quit pressing **:wq** keys which saves the setting automatically.

```
# crontab -e
```

3. List Scheduled Cron Jobs

To list scheduled jobs of a particular user called **tecmint** using option as **-u** (User) and **-l** (List).

```
# crontab -u tecmint -l  
no crontab for tecmint
```

Note: Only **root** user have complete privileges to see other users crontab entry. Normal user can't view it others.

4. Remove Crontab Entry

Caution: Crontab with **-r** parameter will remove complete scheduled jobs without confirmation from crontab. Use **-i** option before deleting user's crontab.

```
# crontab -r
```

5. Prompt Before Deleting Crontab

crontab with **-i** option will prompt you confirmation from user before deleting user's crontab.

```
# crontab -i -r  
crontab: really delete root's crontab?
```

6. Allowed special character (*, -, /, ?, #)

1. **Asterik(*)** – Match all values in the field or any possible value.
2. **Hyphen(-)** – To define range.
3. **Slash (/)** – 1st field /10 meaning every ten minute or increment of range.
4. **Comma (,)** – To separate items.

7. System Wide Cron Schedule

System administrator can use predefine cron directory as shown below.

1. /etc/cron.d
2. /etc/cron.daily
3. /etc/cron.hourly
4. /etc/cron.monthly
5. /etc/cron.weekly

[8. Schedule a Jobs for Specific Time](#)

The below jobs delete empty files and directory from **/tmp** at **12:30** am daily. You need to mention user name to perform crontab command. In below example **root** user is performing cron job.

```
# crontab -e
30 0 * * *    root    find /tmp -type f -empty -delete
```

[9. Special Strings for Common Schedule](#)

Strings	Meanings
@reboot	Command will run when the system reboot.
@daily	Once per day or may use @midnight.
@weekly	Once per week.
@yearly	Once per year. we can use @annually keyword also.

Need to replace five fields of cron command with keyword if you want to use the same.

[10. Multiple Commands with Double amper-sand\(&&\)](#)

In below example command1 and command2 run daily.

```
# crontab -e
@daily <command1> && <command2>
```

[11. Disable Email Notification.](#)

By default cron send mail to user account executing cronjob. If you want to disable it add your cron job similar to below example. Using **>/dev/null 2>&1** option at the end of the file will redirect all the output of the cron results under **/dev/null**.

```
[root@tecmint ~]# crontab -e
* * * * * >/dev/null 2>&1
```

conclusion: Automation of tasks may help us to perform our task better ways, error free and efficiently. You may refer manual page of crontab for more information typing ‘**man crontab**’ command in your terminal.

13 Basic Cat Command Examples in Linux

by [Ravi Saive](#) | Published: August 25, 2012 | Last Updated: January 11, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

The cat (short for “**concatenate**“) command is one of the most frequently used command in Linux/Unix like operating systems. **cat** command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files. In this article, we are going to find out handy use of **cat** commands with their examples in Linux.

Read Also: [Learn How to use ‘cat’ and ‘tac’ \(Reverse of cat Command\) in Linux](#)



13 Basic Linux Cat Commands

General Syntax

```
cat [OPTION] [FILE]...
```

1. Display Contents of File

In the below example, it will show contents of **/etc/passwd** file.

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
narad:x:500:500::/home/narad:/bin/bash
```

2. View Contents of Multiple Files in terminal

In below example, it will display contents of **test** and **test1** file in terminal.

```
# cat test test1
Hello everybody
```

```
Hi world,
```

3. Create a File with Cat Command

We will create a file called **test2** file with below command.

```
# cat >test2
```

Awaits input from user, type desired text and press **CTRL+D** (hold down **Ctrl Key** and type '**d**') to exit. The text will be written in **test2** file. You can see content of file with following **cat** command.

```
# cat test2
hello everyone, how do you do?
```

4. Use Cat Command with More & Less Options

If file having large number of content that won't fit in output terminal and screen scrolls up very fast, we can use parameters more and less with **cat** command as show above.

```
# cat song.txt | more
# cat song.txt | less
```

5. Display Line Numbers in File

With **-n** option you could see the line numbers of a file **song.txt** in the output terminal.

```
# cat -n song.txt
1 "Heal The World"
2 There's A Place In
3 Your Heart
4 And I Know That It Is Love
5 And This Place Could
6 Be Much
7 Brighter Than Tomorrow
8 And If You Really Try
9 You'll Find There's No Need
10 To Cry
11 In This Place You'll Feel
12 There's No Hurt Or Sorrow
```

6. Display \$ at the End of File

In the below, you can see with **-e** option that '\$' is shows at the end of line and also in space showing '\$' if there is any gap between paragraphs. This options is useful to squeeze multiple lines in a single line.

```
# cat -e test
hello everyone, how do you do?$
$
```

```
Hey, am fine.$
How's your training going on?$
$
```

7. Display Tab separated Lines in File

In the below output, we could see **TAB** space is filled up with ‘^I‘ character.

```
# cat -T test
hello ^Ieveryone, how do you do?
Hey, ^Iam fine.
^I^IHow's your training ^Igoing on?
Let's do ^Isome practice in Linux.
```

8. Display Multiple Files at Once

In the below example we have three files **test**, **test1** and **test2** and able to view the contents of those file as shown above. We need to separate each file with ; (semi colon).

```
# cat test; cat test1; cat test2
This is test file
This is test1 file.
This is test2 file.
```

9. Use Standard Output with Redirection Operator

We can redirect standard output of a file into a new file else existing file with ‘>‘ (greater than) symbol. Careful, existing contents of **test1** will be overwritten by contents of **test** file.

```
# cat test > test1
```

10. Appending Standard Output with Redirection Operator

Appends in existing file with ‘>>‘ (double greater than) symbol. Here, contents of **test** file will be appended at the end of **test1** file.

```
# cat test >> test1
```

11. Redirecting Standard Input with Redirection Operator

When you use the redirect with standard input ‘<‘ (less than symbol), it use file name **test2** as a input for a command and output will be shown in a terminal.

```
# cat < test2
This is test2 file.
```

12. Redirecting Multiple Files Contain in a Single File

This will create a file called **test3** and all output will be redirected in a newly created file.

```
# cat test test1 test2 > test3
```

13. Sorting Contents of Multiple Files in a Single File

This will create a file **test4** and output of **cat** command is piped to sort and result will be redirected in a newly created file.

```
# cat test test1 test2 test3 | sort > test4
```

This article shows the basic commands that may help you to explore **cat** command. You may refer man page of **cat** command if you want to know more options. In our next article we will cover more advanced cat commands. Please share it if you find this article useful through our comment box below.

12 TOP Command Examples in Linux

by [Ravi Saive](#) | Published: March 4, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

This is the part of our on-going series of commands in Linux. We have covered basic [ls command](#) and [cat command](#). In this article, we are trying to explore **top** command which is one of the most frequently used commands in our daily system administrative jobs. **top** command displays processor activity of your Linux box and also displays tasks managed by kernel in real-time. It'll show processor and memory are being used and other information like running processes. This may help you to take correct action. **top** command found in UNIX-like operating systems.



Linux Top Command Examples

You might also be interested in following tutorials :

1. [Htop \(Linux Process Monitoring\) tool for RHEL, CentOS & Fedora](#)

2. [Iotop \(Monitor Linux Disk I/O\) in RHEL, CentOS and Fedora](#)

1. Display of Top Command

In this example, it will show information like **tasks**, **memory**, **cpu** and **swap**. Press ‘q’ to quit window.

```
# top
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
25454	root	20	0	397m	107m	13m	S	2.3	10.8	25:19.18	skype
269	root	20	0	0	0	0	S	0.3	0.0	0:51.24	scsi_eh_1
1	root	20	0	2872	1400	1200	S	0.0	0.1	0:00.89	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.16	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:51.23	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:00.33	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	0:00.10	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	6:44.34	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:00.27	watchdog/1
11	root	20	0	0	0	0	S	0.0	0.0	0:04.05	events/0
12	root	20	0	0	0	0	S	0.0	0.0	0:04.87	events/1
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cgroup
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	netns
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	async/mgr
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pm
18	root	20	0	0	0	0	S	0.0	0.0	0:00.23	sync_supers

Linux Top Command

2. Sorting with -O (Uppercase Letter ‘O’).

Press (**Shift+O**) to Sort field via field letter, for example press ‘a’ letter to sort process with PID (**Process ID**).

root@tecmint:~

http://www.tecmint.com

Current Sort Field: **K** for window 1:Def

Select sort field via field letter, type any other key to return []

a: PID	= Process Id	y: WCHAN	= Sleeping in Function
b: PPID	= Parent Process Pid	z: Flags	= Task Flags <sched.h>
c: RUSER	= Real user name	Note1:	
d: UID	= User Id	If a selected sort field can't be shown due to screen width or your field order, the '<' and '>' keys will be unavailable until a field within viewable range is chosen.	
e: USER	= User Name	Note2:	
f: GROUP	= Group Name	Field sorting uses internal values, not those in column display. Thus, the TTY & WCHAN fields will violate strict ASCII collating sequence. (shame on you if WCHAN is chosen)	
g: TTY	= Controlling Tty		
h: PR	= Priority		
i: NI	= Nice value		
j: P	= Last used cpu (SMP)		
* K: %CPU	= CPU usage		
l: TIME	= CPU Time		
m: TIME+	= CPU Time, hundredths		
n: %MEM	= Memory usage (RES)		
o: VIRT	= Virtual Image (kb)		
p: SWAP	= Swapped size (kb)		
q: RES	= Resident size (kb)		
r: CODE	= Code size (kb)		
s: DATA	= Data+Stack size (kb)		
t: SHR	= Shared Mem size (kb)		
u: nFLT	= Page Fault count		
v: nDRT	= Dirty Pages count		
w: S	= Process Status		
x: COMMAND	= Command name/line		

Sorting Process ID's with Top

Type any key to return to main top window with sorted **PID** order as shown in below screen.
Press '**q**' to quit exit the window.

```
root@tecmint:~ http://www.tecmint.com
top - 11:43:48 up 1 day, 22:58,  2 users,  load average: 0.22, 0.08, 0.07
Tasks: 141 total,   1 running, 139 sleeping,   0 stopped,   1 zombie
Cpu(s): 0.7%us, 0.5%sy, 0.0%ni, 98.7%id, 0.1%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1021108k total, 983100k used, 38008k free, 134584k buffers
Swap: 2046968k total,     0k used, 2046968k free, 599576k cached

PID USER      PR NI VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
31290 root      18 -2 2584 872 380 S  0.0  0.1  0:00.00 udevd
31084 root      20  0 11168 2816 2352 S  0.0  0.3  0:00.00 gconf-helper
31074 root      20  0 20424 6516 5136 S  0.0  0.6  0:07.93 gnome-screensav
31069 root      20  0 112m 11m 8772 S  0.0  1.1  0:00.15 nm-applet
31063 root      9 -11 99.4m 9524 7736 S  0.0  0.9  0:08.48 pulseaudio
31059 root      20  0 24884 8324 6884 S  0.0  0.8  0:00.07 gpk-update-icon
31058 root      20  0 6592 3700 2836 S  0.0  0.4  0:00.05 polkitd
31054 root      20  0 18444 5472 4644 S  0.0  0.5  0:00.01 polkit-gnome-au
31050 root      20  0 19628 6872 5760 S  0.0  0.7  0:00.44 gnome-power-man
31045 root      20  0 153m 10m 8432 S  0.0  1.0  0:00.14 gnome-volume-co
31042 root      20  0 25664 8436 7048 S  0.0  0.8  0:00.04 notification-ar
31041 root      20  0 46772 13m 10m S  0.0  1.3  0:01.41 clock-applet
31039 root      20  0 82600 10m 8956 S  0.0  1.1  0:00.06 gdm-user-switch
31031 root      20  0 5544 736 528 S  0.0  0.1  0:12.84 udisks-daemon
31030 root      20  0 5760 2960 2432 S  0.0  0.3  0:03.90 udisks-daemon
31028 root      20  0 7536 2660 2344 S  0.0  0.3  0:00.00 gvfsd-trash
31024 root      20  0 7744 2776 2396 S  0.0  0.3  0:00.12 gvfs-gdu-volume
31022 root      20  0 72000 11m 9416 S  0.0  1.1  0:01.73 wnck-applet
31021 root      20  0 71472 10m 8708 S  0.0  1.1  0:00.07 trashapplet
31014 root      20  0 42660 3276 2596 S  0.0  0.3  0:00.04 bonobo-activati
```

Sorting Process ID's

3. Display Specific User Process

Use top command with ‘u’ option will display specific **User** process details.

```
# top -u tecmint
```

```
tecmint@tecmint:/root
http://www.tecmint.com
top - 11:48:42 up 1 day, 23:03, 2 users, load average: 0.21, 0.18, 0.11
Tasks: 143 total, 1 running, 141 sleeping, 0 stopped, 1 zombie
Cpu(s): 1.0%us, 0.5%sy, 0.0%ni, 98.5%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1021108k total, 983968k used, 37140k free, 134592k buffers
Swap: 2046968k total, 0k used, 2046968k free, 599660k cached

 PID USER      PR  NI    VIRT    RES   SHR S %CPU %MEM     TIME+ COMMAND
26173 tecmint  20   0   5104  1680 1428 S  0.0  0.2   0:00.00 bash
26185 tecmint  20   0   2680  1136  876 R  0.0  0.1   0:00.01 top
```

Top with Specific User Processes

4. Highlight Running Process in Top

Press ‘z’ option in running top command will display running process in color which may help you to identified running process easily.

```
root@tecmint:~ http://www.tecmint.com
top - 11:54:36 up 1 day, 23:09, 2 users, load average: 0.00, 0.09, 0.09
Tasks: 141 total, 2 running, 138 sleeping, 0 stopped, 1 zombie
Cpu(s): 1.0%us, 0.5%sy, 0.0%ni, 98.5%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1021108k total, 983116k used, 37992k free, 134592k buffers
Swap: 2046968k total, 0k used, 2046968k free, 599660k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 9 root 20 0 0 0 0 S 3.0 0.0 6:50.87 ksoftirqd/1
25454 root 20 0 397m 107m 13m S 2.3 10.8 25:44.48 skype
29927 root 20 0 3840 1156 1020 S 0.3 0.1 0:46.89 bald-addon-stor-
 1 root 20 0 2872 1400 1200 S 0.0 0.1 0:00.89 init
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
 3 root RT 0 0 0 0 S 0.0 0.0 0:00.16 migration/0
 4 root 20 0 0 0 0 S 0.0 0.0 0:52.12 ksoftirqd/0
 5 root RT 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
 6 root RT 0 0 0 0 S 0.0 0.0 0:00.33 watchdog/0
 7 root RT 0 0 0 0 S 0.0 0.0 0:00.10 migration/1
 8 root RT 0 0 0 0 S 0.0 0.0 0:00.00 migration/1
10 root RT 0 0 0 0 S 0.0 0.0 0:00.27 watchdog/1
11 root 20 0 0 0 0 S 0.0 0.0 0:04.12 events/0
12 root 20 0 0 0 R 0.0 0.0 0:04.91 events/1
13 root 20 0 0 0 S 0.0 0.0 0:00.00 cgroup
14 root 20 0 0 0 S 0.0 0.0 0:00.00 khelper
15 root 20 0 0 0 S 0.0 0.0 0:00.00 netns
16 root 20 0 0 0 S 0.0 0.0 0:00.00 async/mgr
17 root 20 0 0 0 S 0.0 0.0 0:00.00 pm
18 root 20 0 0 0 S 0.0 0.0 0:00.23 sync_supers
```

Top Process with Colorful

5. Shows Absolute Path of Processes

Press ‘c’ option in running top command, it will display absolute path of running process.

```
root@tecmint:~ http://www.tecmint.com
top - 11:59:10 up 1 day, 23:14, 2 users, load average: 0.00, 0.03, 0.06
Tasks: 141 total, 1 running, 139 sleeping, 0 stopped, 1 zombie
Cpu(s): 0.8%us, 0.5%sy, 0.0%ni, 98.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1021108k total, 983224k used, 37884k free, 134592k buffers
Swap: 2046968k total, 0k used, 2046968k free, 599660k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 9 root 20 0 0 0 0 S 3.3 0.0 6:52.74 [ksoftirqd/1]
25454 root 20 0 397m 107m 13m S 2.3 10.8 25:50.75 /opt/skype/skype --resour
26192 root 20 0 2680 1136 880 R 0.3 0.1 0:00.01 top
 1 root 20 0 2872 1400 1200 S 0.0 0.1 0:00.89 /sbin/init
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [kthreadd]
 3 root RT 0 0 0 0 0 S 0.0 0.0 0:00.16 [migration/0]
 4 root 20 0 0 0 0 S 0.0 0.0 0:52.52 [ksoftirqd/0]
 5 root RT 0 0 0 0 0 S 0.0 0.0 0:00.00 [migration/0]
 6 root RT 0 0 0 0 0 S 0.0 0.0 0:00.33 [watchdog/0]
 7 root RT 0 0 0 0 0 S 0.0 0.0 0:00.10 [migration/1]
 8 root RT 0 0 0 0 0 S 0.0 0.0 0:00.00 [migration/1]
10 root RT 0 0 0 0 0 S 0.0 0.0 0:00.27 [watchdog/1]
11 root 20 0 0 0 0 S 0.0 0.0 0:04.13 [events/0]
12 root 20 0 0 0 0 S 0.0 0.0 0:04.92 [events/1]
13 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [cgroup]
14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [khelper]
15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [netns]
16 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [async/mgr]
17 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [pm]
18 root 20 0 0 0 0 S 0.0 0.0 0:00.23 [sync_supers]
19 root 20 0 0 0 0 S 0.0 0.0 0:00.25 [bdi-default]
20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 [kintegrityd/0]
```

Top with Specific Process Path

6. Change Delay or Set ‘Screen Refresh Interval’ in Top

By default screen refresh interval is **3.0** seconds, same can be change pressing ‘**d**‘ option in running top command and change it as desired as shown below.

```
root@tecmint:~ top - 12:04:29 up 1 day, 23:19,  2 users,  load average: 0.00, 0.00, 0.03
Tasks: 141 total,   2 running, 138 sleeping,   0 stopped,   1 zombie
Cpu(s):  0.8%us,  0.5%sy,  0.0%ni, 98.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem: 1021108k total,  983240k used,   37868k free, 134600k buffers
Swap: 2046968k total,      0k used, 2046968k free, 599668k cached
Change delay from 3.0 to: 2
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
25454	root	20	0	397m	107m	13m	S	2.3	10.8	25:58.00	/opt/skype/skype --resour
9	root	20	0	0	0	0	R	0.3	0.0	6:55.87	[ksoftirqd/1]
26192	root	20	0	2680	1136	880	R	0.3	0.1	0:00.68	top
1	root	20	0	2872	1400	1200	S	0.0	0.1	0:00.89	/sbin/init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kthreadd]
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.16	[migration/0]
4	root	20	0	0	0	0	S	0.0	0.0	0:52.92	[ksoftirqd/0]
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	[migration/0]
6	root	RT	0	0	0	0	S	0.0	0.0	0:00.34	[watchdog/0]
7	root	RT	0	0	0	0	S	0.0	0.0	0:00.10	[migration/1]
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	[migration/1]
10	root	RT	0	0	0	0	S	0.0	0.0	0:00.27	[watchdog/1]
11	root	20	0	0	0	0	S	0.0	0.0	0:04.15	[events/0]
12	root	20	0	0	0	0	S	0.0	0.0	0:04.93	[events/1]
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[cgroup]
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[khelper]
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[netns]
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[async/mgr]
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[pm]
18	root	20	0	0	0	0	S	0.0	0.0	0:00.23	[sync_supers]
19	root	20	0	0	0	0	S	0.0	0.0	0:00.25	[bdi-default]
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	[kintegrityd/0]

Top – Set Refresh Time

7. Kill running process with argument ‘k’

You can kill a process after finding **PID** of process by pressing ‘**k**’ option in running top command without exiting from top window as shown below.

```
root@tecmint:~# top - 12:07:57 up 1 day, 23:23,  2 users,  load average: 0.00, 0.00, 0.01
Tasks: 141 total,   1 running, 139 sleeping,   0 stopped,   1 zombie
Cpu(s): 0.7%us, 0.5%sy, 0.0%ni, 98.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1021108k total, 983128k used, 37980k free, 134600k buffers
Swap: 2046968k total,     0k used, 2046968k free, 599668k cached
PID to kill: 25454
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
25454	root	20	0	397m	107m	13m	S	2.3	10.8	26:02.74	skype
11	root	20	0	0	0	0	S	0.3	0.0	0:04.17	events/0
26208	root	20	0	2680	1136	876	R	0.3	0.1	0:00.01	top
1	root	20	0	2872	1400	1200	S	0.0	0.1	0:00.89	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.16	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:53.09	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:00.34	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	0:00.10	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	6:57.17	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:00.27	watchdog/1
12	root	20	0	0	0	0	S	0.0	0.0	0:04.94	events/1
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cgroup
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	netns
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	async/mgr
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pm
18	root	20	0	0	0	0	S	0.0	0.0	0:00.23	sync_supers
19	root	20	0	0	0	0	S	0.0	0.0	0:00.25	bdi-default
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kintegrityd/0
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kintegrityd/1

Top – Kill Process ID

8. Sort by CPU Utilisation

Press (**Shift+P**) to sort processes as per **CPU** utilization. See screenshot below.

```
root@tecmint:~ top - 12:14:34 up 1 day, 23:29,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 141 total,   1 running, 139 sleeping,   0 stopped,   1 zombie
Cpu(s): 0.7%us, 0.5%sy, 0.0%ni, 98.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1021108k total,  983252k used,   37856k free, 134600k buffers
Swap: 2046968k total,      0k used, 2046968k free, 599672k cached

 PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
25454 root      20   0 397m 107m 13m S  2.3 10.8 26:11.86 skype
  1 root      20   0 2872 1400 1200 S  0.0  0.1  0:00.89 init
  2 root      20   0      0      0  0 S  0.0  0.0  0:00.00 kthreadd
  3 root      RT   0      0      0  0 S  0.0  0.0  0:00.16 migration/0
  4 root      20   0      0      0  0 S  0.0  0.0  0:53.85 ksoftirqd/0
  5 root      RT   0      0      0  0 S  0.0  0.0  0:00.00 migration/0
  6 root      RT   0      0      0  0 S  0.0  0.0  0:00.34 watchdog/0
  7 root      RT   0      0      0  0 S  0.0  0.0  0:00.10 migration/1
  8 root      RT   0      0      0  0 S  0.0  0.0  0:00.00 migration/1
  9 root      20   0      0      0  0 S  0.0  0.0  6:59.99 ksoftirqd/1
 10 root     RT   0      0      0  0 S  0.0  0.0  0:00.27 watchdog/1
 11 root     20   0      0      0  0 S  0.0  0.0  0:04.19 events/0
 12 root     20   0      0      0  0 S  0.0  0.0  0:04.95 events/1
 13 root     20   0      0      0  0 S  0.0  0.0  0:00.00 cgroup
 14 root     20   0      0      0  0 S  0.0  0.0  0:00.00 khelper
 15 root     20   0      0      0  0 S  0.0  0.0  0:00.00 netns
 16 root     20   0      0      0  0 S  0.0  0.0  0:00.00 async/mgr
 17 root     20   0      0      0  0 S  0.0  0.0  0:00.00 pm
 18 root     20   0      0      0  0 S  0.0  0.0  0:00.23 sync_supers
 19 root     20   0      0      0  0 S  0.0  0.0  0:00.25 bdi-default
 20 root     20   0      0      0  0 S  0.0  0.0  0:00.00 kintegrityd/0
 21 root     20   0      0      0  0 S  0.0  0.0  0:00.00 kintegrityd/1
 22 root     20   0      0      0  0 S  0.0  0.0  0:01.07 kblockd/0
```

Top – High CPU Utilization

9. Renice a Process

You can use ‘r’ option to change the priority of the process also called Renice.

root@tecmint:- http://www.tecmint.com

```
top - 12:16:01 up 1 day, 23:31,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 141 total,   1 running, 139 sleeping,   0 stopped,   1 zombie
Cpu(s): 0.8%us, 0.7%sy, 0.0%ni, 98.5%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1021108k total,  983128k used,  37980k free, 134600k buffers
Swap: 2046968k total,      0k used, 2046968k free, 599672k cached
PID to renice: 25454
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
25454	root	20	0	397m	107m	13m	S	2.3	10.8	26:13.86	skype
26211	root	20	0	2680	1140	884	R	0.7	0.1	0:00.34	top
31031	root	20	0	5544	736	528	S	0.3	0.1	0:13.15	udisks-daemon
1	root	20	0	2872	1400	1200	S	0.0	0.1	0:00.89	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.16	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:53.86	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:00.34	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	0:00.10	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	7:00.62	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:00.27	watchdog/1
11	root	20	0	0	0	0	S	0.0	0.0	0:04.19	events/0
12	root	20	0	0	0	0	S	0.0	0.0	0:04.95	events/1
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cgroup
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	netns
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	async/mgr
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pm
18	root	20	0	0	0	0	S	0.0	0.0	0:00.23	sync_supers
19	root	20	0	0	0	0	S	0.0	0.0	0:00.25	bdi-default
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	Kintegrityd/0

Top – Renice Process

10. Save Top Command Results

Press (**Shift+W**) to save the running top command results under **/root/.toprc**.

The screenshot shows a terminal window titled "root@tecmint:~" with the URL "http://www.tecmint.com" at the top. The window displays the output of the "top" command. The output includes system statistics like CPU usage and memory, followed by a detailed list of processes. The processes table has columns: PID, USER, PR, NI, VIRT, RES, SHR, %CPU, %MEM, TIME+, and COMMAND. The COMMAND column lists various kernel threads and daemons.

PID	USER	PR	NI	VIRT	RES	SHR	%CPU	%MEM	TIME+	COMMAND
9	root	20	0	0	0	0 S	3.8	0.0	7:01.96	ksoftirqd/1
25454	root	20	0	397m	107m	13m S	2.7	10.8	26:16.29	skype
4	root	20	0	0	0	0 S	0.4	0.0	0:53.95	ksoftirqd/0
26212	root	20	0	2680	1140	884 R	0.2	0.1	0:00.03	top
1	root	20	0	2872	1400	1200 S	0.0	0.1	0:00.89	init
2	root	20	0	0	0	0 S	0.0	0.0	0:00.00	kthreadd
3	root	RT	0	0	0	0 S	0.0	0.0	0:00.16	migration/0
5	root	RT	0	0	0	0 S	0.0	0.0	0:00.00	migration/0
6	root	RT	0	0	0	0 S	0.0	0.0	0:00.34	watchdog/0
7	root	RT	0	0	0	0 S	0.0	0.0	0:00.10	migration/1
8	root	RT	0	0	0	0 S	0.0	0.0	0:00.00	migration/1
10	root	RT	0	0	0	0 S	0.0	0.0	0:00.27	watchdog/1
11	root	20	0	0	0	0 S	0.0	0.0	0:04.20	events/0
12	root	20	0	0	0	0 S	0.0	0.0	0:04.95	events/1
13	root	20	0	0	0	0 S	0.0	0.0	0:00.00	cgroup
14	root	20	0	0	0	0 S	0.0	0.0	0:00.00	khelper
15	root	20	0	0	0	0 S	0.0	0.0	0:00.00	netns
16	root	20	0	0	0	0 S	0.0	0.0	0:00.00	async/mgr
17	root	20	0	0	0	0 S	0.0	0.0	0:00.00	pm
18	root	20	0	0	0	0 S	0.0	0.0	0:00.23	sync_supers
19	root	20	0	0	0	0 S	0.0	0.0	0:00.25	bdi-default
20	root	20	0	0	0	0 S	0.0	0.0	0:00.00	kintegrityd/0
21	root	20	0	0	0	0 S	0.0	0.0	0:00.00	kintegrityd/1

Top Command Save Results

11. Getting Top Command Help

Press '**h**' option to obtain the top command help.

The screenshot shows a terminal window with the title bar "root@tecmin:~ http://www.tecmint.com". The window displays the help documentation for the top command, which is version 3.2.8. The help text includes various keyboard shortcuts and their descriptions, such as 'z,B' for changing color mappings, 'f,o' for fields/columns, and 'q' for quitting. It also mentions options like 'cumulative mode Off.', 'delay 3.0 secs', and 'secure mode Off.'.

```
root@tecmin:~ http://www.tecmint.com
Help for Interactive Commands - procs version 3.2.8
Window 1:Def: Cumulative mode Off. System: Delay 3.0 secs; Secure mode Off.

Z,B      Global: 'Z' change color mappings; 'B' disable/enable bold
I,t,m    Toggle Summaries: 'I' load avg; 't' task/cpu stats; 'm' mem info
I,I      Toggle SMP view: 'I' single/separate states; 'I' Irix/Solaris mode

f,o      . Fields/Columns: 'f' add or remove; 'o' change display order
F or O   . Select sort field
<,>     . Move sort field: '<' next col left; '>' next col right
R,H     . Toggle: 'R' normal/reverse sort; 'H' show threads
C,i,S   . Toggle: 'C' cmd name/line; 'i' idle tasks; 'S' cumulative time
X,Y     . Toggle highlights: 'X' sort field; 'Y' running tasks
Z,B     . Toggle: 'Z' color/mono; 'B' bold/reverse (only if 'X' or 'Y')
U       . Show specific user only
N or #   . Set maximum tasks displayed

K,R     Manipulate tasks: 'K' kill; 'R' renice
D or S   Set update interval
W       Write configuration file
Q       Quit
( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
any other key to continue [
```

Top Command Help

12. Exit Top Command After Specific repetition

Top output keep refreshing until you press 'q'. With below command top command will automatically exit after 10 number of repetition.

```
# top -n 10
```

There are number of arguments to know more about **top** command you may refer man page of **top** command. Please share it if you find this article useful through our comment box below.

10 lsof Command Examples in Linux

by [Narad Shrestha](#) | Published: August 31, 2012 | Last Updated: September 13, 2012

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

This is our on-going series of Linux commands and in this article we are going to review **lsof** command with practical examples. **lsof** meaning ‘LiSt Open Files’ is used to find out which files are open by which process. As we all know **Linux/Unix** considers everything as a files (**pipes, sockets, directories, devices** etc). One of the reason to use **lsof** command is when a disk cannot be unmounted as it says the files are being used. With the help of this command we can easily identify the files which are in use.



10 Linux lsof Command Examples

1. List all Open Files with lsof Command

In the below example, it will show long listing of open files some of them are extracted for better understanding which displays the columns like **Command, PID, USER, FD, TYPE** etc.

```
# lsof
COMMAND   PID   USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
init      1   root    cwd   DIR    253,0    4096   2   /
init      1   root    rtd   DIR    253,0    4096   2   /
init      1   root    txt   REG    253,0  145180 147164
/sbin/init
init      1   root    mem   REG    253,0  1889704 190149
/lib/libc-2.12.so
init      1   root    0u    CHR    1,3     0t0    3764
/dev/null
init      1   root    1u    CHR    1,3     0t0    3764
/dev/null
init      1   root    2u    CHR    1,3     0t0    3764
/dev/null
init      1   root    3r    FIFO    0,8     0t0    8449 pipe
init      1   root    4w    FIFO    0,8     0t0    8449 pipe
```

```

init      1      root   5r      DIR      0,10      0      1
inotify
init      1      root   6r      DIR      0,10      0      1
inotify
init      1      root   7u      unix 0xc1513880      0t0      8450 socket

```

Sections and it's values are self-explanatory. However, we'll review **FD & TYPE** columns more precisely.

FD – stands for File descriptor and may seen some of the values as:

1. **cwd** current working directory
2. **rtd** root directory
3. **txt** program text (code and data)
4. **mem** memory-mapped file

Also in **FD** column numbers like **1u** is actual file descriptor and followed by u,r,w of it's mode as:

1. **r** for read access.
2. **w** for write access.
3. **u** for read and write access.

TYPE – of files and it's identification.

1. **DIR** – Directory
2. **REG** – Regular file
3. **CHR** – Character special file.
4. **FIFO** – First In First Out

2. List User Specific Opened Files

The below command will display the list of all opened files of user **tecmint**.

```
# lsof -u tecmint
COMMAND  PID  USER   FD   TYPE      DEVICE SIZE/OFF NODE NAME
sshd    1838 tecmint cwd      DIR  253,0    4096   2 /
sshd    1838 tecmint rtd      DIR  253,0    4096   2 /
sshd    1838 tecmint txt      REG  253,0  532336 188129 /usr/sbin/sshd
sshd    1838 tecmint mem      REG  253,0   19784 190237 /lib/libdl-
2.12.so
sshd    1838 tecmint mem      REG  253,0  122436 190247
/lib/libselinux.so.1
sshd    1838 tecmint mem      REG  253,0  255968 190256
/lib/libgssapi_krb5.so.2.2
sshd    1838 tecmint mem      REG  253,0   874580 190255
/lib/libkrb5.so.3.3
```

3. Find Processes running on Specific Port

To find out all the running process of specific port, just use the following command with option -i. The below example will list all running process of port **22**.

```
# lsof -i TCP:22
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
sshd 1471 root 3u IPv4 12683 0t0 TCP *:ssh (LISTEN)
sshd 1471 root 4u IPv6 12685 0t0 TCP *:ssh (LISTEN)
```

4. List Only IPv4 & IPv6 Open Files

In below example shows only **IPv4** and **IPv6** network files open with separate commands.

```
# lsof -i 4
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rpcbind 1203 rpc 6u IPv4 11326 0t0 UDP *:sunrpc
rpcbind 1203 rpc 7u IPv4 11330 0t0 UDP *:954
rpcbind 1203 rpc 8u IPv4 11331 0t0 TCP *:sunrpc (LISTEN)
avahi-dae 1241 avahi 13u IPv4 11579 0t0 UDP *:mdns
avahi-dae 1241 avahi 14u IPv4 11580 0t0 UDP *:58600
# lsof -i 6
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rpcbind 1203 rpc 9u IPv6 11333 0t0 UDP *:sunrpc
rpcbind 1203 rpc 10u IPv6 11335 0t0 UDP *:954
rpcbind 1203 rpc 11u IPv6 11336 0t0 TCP *:sunrpc (LISTEN)
rpc.statd 1277 rpcuser 10u IPv6 11858 0t0 UDP *:55800
rpc.statd 1277 rpcuser 11u IPv6 11862 0t0 TCP *:56428 (LISTEN)
cupsd 1346 root 6u IPv6 12112 0t0 TCP localhost:ipp
(LISTEN)
```

5. List Open Files of TCP Port ranges 1-1024

To list all the running process of open files of **TCP Port ranges from 1-1024**.

```
# lsof -i TCP:1-1024
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rpcbind 1203 rpc 11u IPv6 11336 0t0 TCP *:sunrpc (LISTEN)
cupsd 1346 root 7u IPv4 12113 0t0 TCP localhost:ipp (LISTEN)
sshd 1471 root 4u IPv6 12685 0t0 TCP *:ssh (LISTEN)
master 1551 root 13u IPv6 12898 0t0 TCP localhost:smtp (LISTEN)
sshd 1834 root 3r IPv4 15101 0t0 TCP 192.168.0.2:ssh-
>192.168.0.1:conclave-cpp (ESTABLISHED)
sshd 1838 tecmint 3u IPv4 15101 0t0 TCP 192.168.0.2:ssh-
>192.168.0.1:conclave-cpp (ESTABLISHED)
sshd 1871 root 3r IPv4 15842 0t0 TCP 192.168.0.2:ssh-
>192.168.0.1:groove (ESTABLISHED)
httpd 1918 root 5u IPv6 15991 0t0 TCP *:http (LISTEN)
httpd 1918 root 7u IPv6 15995 0t0 TCP *:https (LISTEN)
```

6. Exclude User with '^' Character

Here, we have excluded **root** user. You can exclude a particular user using '^' with command as shown above.

```
# lsof -i -u^root
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rpcbind 1203 rpc 6u IPv4 11326 0t0 UDP *:sunrpc
rpcbind 1203 rpc 7u IPv4 11330 0t0 UDP *:954
rpcbind 1203 rpc 8u IPv4 11331 0t0 TCP *:sunrpc (LISTEN)
rpcbind 1203 rpc 9u IPv6 11333 0t0 UDP *:sunrpc
rpcbind 1203 rpc 10u IPv6 11335 0t0 UDP *:954
rpcbind 1203 rpc 11u IPv6 11336 0t0 TCP *:sunrpc (LISTEN)
avahi-dae 1241 avahi 13u IPv4 11579 0t0 UDP *:mdns
avahi-dae 1241 avahi 14u IPv4 11580 0t0 UDP *:58600
rpc.statd 1277 rpcuser 5r IPv4 11836 0t0 UDP *:soap-beep
rpc.statd 1277 rpcuser 8u IPv4 11850 0t0 UDP *:55146
rpc.statd 1277 rpcuser 9u IPv4 11854 0t0 TCP *:32981 (LISTEN)
rpc.statd 1277 rpcuser 10u IPv6 11858 0t0 UDP *:55800
rpc.statd 1277 rpcuser 11u IPv6 11862 0t0 TCP *:56428 (LISTEN)
```

7. Find Out who's Looking What Files and Commands?

Below example shows user **tecmint** is using command like **ping** and **/etc** directory .

```
# lsof -i -u tecmint
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash 1839 tecmint cwd DIR 253,0 12288 15 /etc
ping 2525 tecmint cwd DIR 253,0 12288 15 /etc
```

8. List all Network Connections

The following command with option ‘**-i**’ shows the list of all network connections ‘**LISTENING & ESTABLISHED**’.

```
# lsof -i
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
rpcbind 1203 rpc 6u IPv4 11326 0t0 UDP *:sunrpc
rpcbind 1203 rpc 7u IPv4 11330 0t0 UDP *:954
rpcbind 1203 rpc 11u IPv6 11336 0t0 TCP *:sunrpc (LISTEN)
avahi-dae 1241 avahi 13u IPv4 11579 0t0 UDP *:mdns
avahi-dae 1241 avahi 14u IPv4 11580 0t0 UDP *:58600
rpc.statd 1277 rpcuser 11u IPv6 11862 0t0 TCP *:56428 (LISTEN)
cupsd 1346 root 6u IPv6 12112 0t0 TCP localhost:ipp
(LISTEN)
cupsd 1346 root 7u IPv4 12113 0t0 TCP localhost:ipp
(LISTEN)
sshd 1471 root 3u IPv4 12683 0t0 TCP *:ssh (LISTEN)
master 1551 root 12u IPv4 12896 0t0 TCP localhost:smtp
(LISTEN)
master 1551 root 13u IPv6 12898 0t0 TCP localhost:smtp
(LISTEN)
sshd 1834 root 3r IPv4 15101 0t0 TCP 192.168.0.2:ssh-
>192.168.0.1:conclave-cpp (ESTABLISHED)
httpd 1918 root 5u IPv6 15991 0t0 TCP *:http (LISTEN)
httpd 1918 root 7u IPv6 15995 0t0 TCP *:https (LISTEN)
clock-app 2362 narad 21u IPv4 22591 0t0 TCP 192.168.0.2:45284-
>www.gov.com:http (CLOSE_WAIT)
```

```
chrome      2377    narad      61u   IPv4  25862          0t0   TCP  192.168.0.2:33358-
>maa03s04-in-f3.1e100.net:http (ESTABLISHED)
chrome      2377    narad      80u   IPv4  25866          0t0   TCP  192.168.0.2:36405-
>bom03s01-in-f15.1e100.net:http (ESTABLISHED)
```

9. Search by PID

The below example only shows whose **PID** is **1 [One]**.

```
# lsof -p 1
COMMAND  PID  USER   FD   TYPE      DEVICE SIZE/OFF NODE NAME
init     1  root    cwd   DIR      253,0    4096   2  /
init     1  root   rtd   DIR      253,0    4096   2  /
init     1  root   txt   REG      253,0  145180 147164 /sbin/init
init     1  root   mem   REG      253,0 1889704 190149 /lib/libc-2.12.so
init     1  root   mem   REG      253,0 142472 189970 /lib/ld-2.12.so
```

10. Kill all Activity of Particular User

Sometimes you may have to kill all the processes for a specific user. Below command will kills all the processes of **tecmint** user.

```
# kill -9 `lsof -t -u tecmint`
```

Note: Here, it's not possible to give example of all available options, this guide is only to show how **lsof** command can be use. You may refer man page of **lsof** command to know more about it. Please share it if you find this article is useful through our comment box below.

Linux and Unix cut command tutorial with examples

Tutorial on using cut, a UNIX and Linux command for cutting sections from each line of files. Examples of cutting by character, byte position, cutting based on delimiter and how to modify the output delimiter.

Estimated reading time: 3 minutes

Table of contents

- - [What is the cut command in UNIX?](#)
 - [How to cut by byte position](#)
 - [How to cut by character](#)
 - [How to cut based on a delimiter](#)
 - [How to cut by complement pattern](#)
 - [How to modify the output delimiter](#)
 - [Further reading](#)

The screenshot shows the first few sections of the man page for the 'cut' command. The title 'CUT(1)' is at the top left, and 'User Commands' is at the top right. The 'NAME' section is highlighted in white, containing the text 'cut - remove sections from files'.

```
CUT(1)                               User Commands

NAME
    cut - remove sections from files
```

What is the cut command in UNIX?

The `cut` command in UNIX is a command line utility for cutting sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and delimiter. It can also be used to cut data from file formats like CSV.

How to cut by byte position

To cut out a section of a line by specifying a byte position use the `-b` option.

```
echo 'baz' | cut -b 2  
a  
echo 'baz' | cut -b 1-2  
ba  
echo 'baz' | cut -b 1,3  
bz
```

How to cut by character

To cut by character use the `-c` option. This selects the characters given to the `-c` option. This can be a list of comma separated numbers, a range of numbers or a single number.

Where your input stream is character based `-c` can be a better option than selecting by bytes as often characters are more than one byte.

In the following example character ‘♣’ is three bytes. By using the `-c` option the character can be correctly selected along with any other characters that are of interest.

```
echo '♣foobar' | cut -c 1,6  
♣  
echo '♣foobar' | cut -c 1-3  
♣f
```

How to cut based on a delimiter

To cut using a delimiter use the `-d` option. This is normally used in conjunction with the `-f` option to specify the field that should be cut.

In the following example a CSV file exists and is saved as `names.csv`.

```
John,Smith,34,London  
Arthur,Evans,21,Newport  
George,Jones,32,Truro
```

The delimiter can be set to a comma with `-d ','`. `cut` can then pull out the fields of interest with the `-f` flag. In the following example the first field is cut.

```
cut -d ',' -f 1 names.csv  
John
```

```
Arthur
George
```

Multiple fields can be cut by passing a comma separated list.

```
cut -d ',' -f 1,4 names.csv
John,London
Arthur,Newport
George,Truro
```

How to cut by complement pattern

To cut by complement us the `--complement` option. Note this option is not available on the BSD version of `cut`. The `--complement` option selects the *inverse* of the options passed to `sort`.

In the following example the `-c` option is used to select the first character. Because the `--complement` option is also passed to `cut` the second and third characters are cut.

```
echo 'foo' | cut --complement -c 1
oo
```

How to modify the output delimiter

To modify the output delimiter use the `--output-delimiter` option. Note that this option is not available on the BSD version of `cut`. In the following example a semi-colon is converted to a space and the first, third and fourth fields are selected.

```
echo 'how;now;brown;cow' | cut -d ';' -f 1,3,4 --output-delimiter=' '
how brown cow
```

10 Practical Linux Cut Command Examples to Select File Columns

by Balakrishnan Mariyappan on June 6, 2013

Linux command cut is used for text processing. You can use this command to extract portion of text from a file by selecting columns.

This tutorial provides few practical examples of cut command that you can use in your day to day command line activities.

For most of the example, we'll be using the following test file.

```
$ cat test.txt
cat command for file oriented operations.
cp command for copy files or directories.
ls command to list out files and directories with its attributes.
```

1. Select Column of Characters

To extract only a desired column from a file use -c option. The following example displays 2nd character from each line of a file test.txt

```
$ cut -c2 test.txt
a
p
s
```

As seen above, the characters a, p, s are the second character from each line of the test.txt file.

2. Select Column of Characters using Range

Range of characters can also be extracted from a file by specifying start and end position delimited with -. The following example extracts first 3 characters of each line from a file called test.txt

```
$ cut -c1-3 test.txt
cat
cp
ls
```

3. Select Column of Characters using either Start or End Position

Either start position or end position can be passed to cut command with -c option.

The following specifies only the start position before the ‘-’. This example extracts from 3rd character to end of each line from test.txt file.

```
$ cut -c3- test.txt
t command for file oriented operations.
command for copy files or directories.
command to list out files and directories with its attributes.
```

The following specifies only the end position after the ‘-’. This example extracts 8 characters from the beginning of each line from test.txt file.

```
$ cut -c-8 test.txt
cat comm
cp comma
ls comma
```

The entire line would get printed when you don't specify a number before or after the ‘-’ as shown below.

```
$ cut -c- test.txt
cat command for file oriented operations.
cp command for copy files or directories.
ls command to list out files and directories with its attributes.
```

4. Select a Specific Field from a File

Instead of selecting x number of characters, if you like to extract a whole field, you can combine option -f and -d. The option -f specifies which field you want to extract, and the option -d specifies what is the field delimiter that is used in the input file.

The following example displays only first field of each lines from /etc/passwd file using the field delimiter : (colon). In this case, the 1st field is the username. The file

```
$ cut -d':' -f1 /etc/passwd
root
daemon
bin
sys
sync
games
bala
```

5. Select Multiple Fields from a File

You can also extract more than one fields from a file or stdout. Below example displays username and home directory of users who has the login shell as “/bin/bash”.

```
$ grep "/bin/bash" /etc/passwd | cut -d':' -f1,6
root:/root
bala:/home/bala
```

To display the range of fields specify start field and end field as shown below. In this example, we are selecting field 1 through 4, 6 and 7

```
$ grep "/bin/bash" /etc/passwd | cut -d':' -f1-4,6,7
root:x:0:0:/root:/bin/bash
bala:x:1000:1000:/home/bala:/bin/bash
```

6. Select Fields Only When a Line Contains the Delimiter

In our /etc/passwd example, if you pass a different delimiter other than : (colon), cut will just display the whole line.

In the following example, we've specified the delimiter as | (pipe), and cut command simply displays the whole line, even when it doesn't find any line that has | (pipe) as delimiter.

```
$ grep "/bin/bash" /etc/passwd | cut -d'|' -f1
root:x:0:0:root:/root:/bin/bash
bala:x:1000:1000:bala,,,:/home/bala:/bin/bash
```

But, it is possible to filter and display only the lines that contains the specified delimiter using -s option.

The following example doesn't display any output, as the cut command didn't find any lines that has | (pipe) as delimiter in the /etc/passwd file.

```
$ grep "/bin/bash" /etc/passwd | cut -d'|' -s -f1
```

7. Select All Fields Except the Specified Fields

In order to complement the selection field list use option –complement.

The following example displays all the fields from /etc/passwd file except field 7

```
$ grep "/bin/bash" /etc/passwd | cut -d':' --complement -s -f7
root:x:0:0:root:/root
bala:x:1000:1000:bala,,,:/home/bala
```

8. Change Output Delimiter for Display

By default the output delimiter is same as input delimiter that we specify in the cut -d option.

To change the output delimiter use the option –output-delimiter as shown below. In this example, the input delimiter is : (colon), but the output delimiter is # (hash).

```
$ grep "/bin/bash" /etc/passwd | cut -d':' -s -f1,6,7 --output-delimiter='#'
root#/root#/bin/bash
bala#/home/bala#/bin/bash
```

9. Change Output Delimiter to Newline

In this example, each and every field of the cut command output is displayed in a separate line. We still used –output-delimiter, but the value is \$'\n' which indicates that we should add a newline as the output delimiter.

```
$ grep bala /etc/passwd | cut -d':' -f1,6,7 --output-delimiter=$'\n'  
bala  
/home/bala  
/bin/bash
```

10. Combine Cut with Other Unix Command Output

The power of cut command can be realized when you combine it with the stdout of some other Unix command.

Once you master the basic usage of cut command that we've explained above, you can wisely use cut command to solve lot of your text manipulation requirements.

The following example indicates how you can extract only useful information from the [ps command](#) output. We also showed how we've filtered the output of ps command using grep and sed before the final output was given to cut command. Here, we've used cut option -d and -f which we've explained in the above examples.

```
$ ps axu | grep python | sed 's/\s\+/ /g' | cut -d' ' -f2,11-  
2231 /usr/bin/python /usr/lib/unity-lens-video/unity-lens-video  
2311 /usr/bin/python /usr/lib/unity-scope-video-remote/unity-scope-video-  
remote  
2414 /usr/bin/python /usr/lib/ubuntuone-client/ubuntuone-syncdaemon  
2463 /usr/bin/python /usr/lib/system-service/system-service-d  
3274 grep --color=auto python
```

How to List Files Installed From a RPM or DEB Package in Linux

by [Aaron Kili](#) | Published: March 30, 2017 | Last Updated: March 30, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Have you ever wondered where the various files contained inside a package are installed (located) in the Linux file system? In this article, we'll show how to list all files installed from or present in a certain package or group of packages in Linux.

This can help you to easily locate important package files like configurations files, documentation and more. Let's look at the different methods of listing files in or installed from a package:

How to List All Files of Installed Package in Linux

You can use the [repoquery command](#) which is part of the [yum-utils to list files installed](#) on a CentOS/RHEL system from a given package.

To install and use **yum-utils**, run the commands below:

```
# yum update  
# yum install yum-utils
```

Now you can list files of an installed RPM package, for example **httpd** web server (note that the package name is case-sensitive). The **--installed** flag means installed packages and **-l** flags enables listing of files:

```
# repoquery --installed -l httpd  
# dnf repoquery --installed -l httpd [On Fedora 22+ versions]
```

```
[root@tecmint ~]# repoquery --installed -l httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/autoindex.conf
/etc/httpd/conf.d/userdir.conf
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf.modules.d
/etc/httpd/conf.modules.d/00-base.conf
/etc/httpd/conf.modules.d/00-dav.conf
/etc/httpd/conf.modules.d/00-lua.conf
/etc/httpd/conf.modules.d/00-mpm.conf
/etc/httpd/conf.modules.d/00-proxy.conf
/etc/httpd/conf.modules.d/00-systemd.conf
/etc/httpd/conf.modules.d/01-cgi.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run
/etc/logrotate.d/httpd
/etc/sysconfig/htcacheload
/etc/sysconfig/httpd
/run/httpd
```

Repoquery List Installed Files of Httpd

Important: In **Fedora 22+** version, the repoquery command is integrated with [dnf package manager](#) for RPM based distribution to list files installed from a package as shown above.

Alternatively, you can as well use the [rpm command](#) below to list the files inside or installed on the system from a .rpm package as follows, where the `-q` and `-l` means to list files in package respectively:

```
# rpm -ql httpd
```

```
[root@tecmint ~]# rpm -ql httpd
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/README
/etc/httpd/conf.d/autoindex.conf
/etc/httpd/conf.d/userdir.conf
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf.modules.d
/etc/httpd/conf.modules.d/00-base.conf
/etc/httpd/conf.modules.d/00-dav.conf
/etc/httpd/conf.modules.d/00-lua.conf
/etc/httpd/conf.modules.d/00-mpm.conf
/etc/httpd/conf.modules.d/00-proxy.conf
/etc/httpd/conf.modules.d/00-systemd.conf
/etc/httpd/conf.modules.d/01-cgi.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run
/etc/logrotate.d/httpd
/etc/sysconfig/htcacheclean
/etc/sysconfig/httpd
/run/httpd
```

RPM Query Package for Installed Files

Another useful option is used to use `-p` to list `.rpm` package files before installing it.

```
# rpm -qplp telnet-server-1.2-137.1.i586.rpm
```

On **Debian/Ubuntu** distributions, you can use the [dpkg command](#) with the `-L` flag to list files installed to your Debian system or its derivatives, from a given `.deb` package.

In this example, we will list files installed from **apache2** web server:

```
$ dpkg -L apache2
```

```
aaronkilik@tecmint ~ $ dpkg -L apache2
/.
/etc
/etc/logrotate.d
/etc/logrotate.d/apache2
/etc/apache2
/etc/apache2/sites-enabled
/etc/apache2/apache2.conf
/etc/apache2/mods-enabled
/etc/apache2/mods-available
/etc/apache2/mods-available/ssl.load
/etc/apache2/mods-available/ldap.load
/etc/apache2/mods-available/proxy_fcgi.load
/etc/apache2/mods-available/file_cache.load
/etc/apache2/mods-available/authz_host.load
/etc/apache2/mods-available/deflate.load
/etc/apache2/mods-available/cgid.load
/etc/apache2/mods-available/userdir.load
/etc/apache2/mods-available/actions.load
/etc/apache2/mods-available/session.load
/etc/apache2/mods-available/suexec.load
/etc/apache2/mods-available/authnz_fcgi.load
/etc/apache2/mods-available/actions.conf
/etc/apache2/mods-available/mime_magic.conf
/etc/apache2/mods-available/include.load
```

dpkg List Installed Packages

Don't forget to check out following useful articles for package management in Linux.

15 Practical Examples of “dpkg commands” for Debian Based Distros

by [Editor](#) | Published: October 1, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Debian GNU/Linux, the mother **Operating System** of a number of Linux distributions including **Knoppix**, **Kali**, **Ubuntu**, **Mint**, etc. uses various package Manager like **dpkg**, **apt**, **aptitude**, **synaptic**, **tasksel**, **deselect**, **dpkg-deb** and **dpkg-split**.



15 dpkg Command Examples

We will be describing each of these briefly before focusing on ‘**dpkg**’ command.

APT Command

Apt stands for **Advanced Package Tool**. It doesn’t deal with ‘**deb**’ package and works directly, but works with ‘**deb**’ archive from the location specified in the “**/etc/apt/sources.list**” file.

Read More : [25 Useful Basic Commands of APT-GET Commands](#)

Aptitude

Aptitude is a text based package manager for **Debian** which is front-end to ‘**apt**’, which enables user to manage packages easily.

Synaptic

Graphical package manager which makes it easy to **install**, **upgrade** and **uninstall** packages even to novice.

Tasksel

Tasksel lets the user to install all the relevant packages related to a specific task, viz., Desktop-environment.

Deselect

A **menu-driven** package management tool, initially used during the first time install and now is replaced with **aptitude**.

Dpkg-deb

Interacts with **Debian** archive.

Dpkg-split

Useful in **splitting** and **merging** large file into chunks of small files to be stored on media of smaller size like **floppy-disk**.

Dpkg Command

dpkg is the main package management program in **Debian** and **Debian** based System. It is used to **install**, **build**, **remove**, and **manage** packages. **Aptitude** is the primary front-end to **dpkg**.

Some the most commonly used **dpkg commands** along with their usages are listed here:

1. Install a Package

For installing an “**.deb**” package, use the command with “**-i**” option. For example, to install an “**.deb**” package called “**flashpluginnonfree_2.8.2+squeeze1_i386.deb**” use the following command.

```
[root@tecmint~]# dpkg -i flashpluginnonfree_2.8.2+squeeze1_i386.deb
Selecting previously unselected package flashplugin-nonfree.
(Reading database ... 465729 files and directories currently installed.)
Unpacking flashplugin-nonfree (from flashplugin-nonfree_3.2_i386.deb) ...
Setting up flashplugin-nonfree (1:3.2) ...
--2013-10-01 16:23:40--
http://fpdownload.macromedia.com/get/flashplayer/pdc/11.2.202.310/install_flash_player_11_linux.i386.tar.gz
Resolving fpdownload.macromedia.com (fpdownload.macromedia.com)...
23.64.66.70
Connecting to fpdownload.macromedia.com
(fpdownload.macromedia.com)|23.64.66.70|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6923724 (6.6M) [application/x-gzip]
```

```
Saving to: '/tmp/flashplugin-nonfree.FPxQ4102fL/install_flash_player_11_linux.i386.tar.gz'
```

2. List all the installed Packages

To view and list all the installed packages, use the “-l” option along with the command.

```
[root@tecmint~]# dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-aWait/Trig-
pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                           Version
Architecture   Description
=====
=====
ii  accerciser                   3.8.0-0ubuntu1      all
interactive Python accessibility explorer for the GNOME desktop
ii  account-plugin-aim           3.6.4-0ubuntu4.1    i386
Messaging account plugin for AIM
ii  account-plugin-facebook     0.10bzr13.03.26-0ubuntu1 i386
GNOME Control Center account plugin for single signon - facebook
ii  account-plugin-flickr       0.10bzr13.03.26-0ubuntu1 i386
GNOME Control Center account plugin for single signon - flickr
ii  account-plugin-generic-oauth 0.10bzr13.03.26-0ubuntu1 i386
GNOME Control Center account plugin for single signon - generic OAuth
ii  account-plugin-google       0.10bzr13.03.26-0ubuntu1 i386
GNOME Control Center account plugin for single signon
rc  account-plugin-identica    0.10bzr13.03.26-0ubuntu1 i386
GNOME Control Center account plugin for single signon - identica
ii  account-plugin-jabber       3.6.4-0ubuntu4.1    i386
Messaging account plugin for Jabber/XMPP
....
```

To view a specific package installed or not use the option “-l” along with package-name. For example, check whether **apache2** package installed or not.

```
[root@tecmint~]# dpkg -l apache2
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-aWait/Trig-
pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                           Version
Architecture   Description
=====
=====
ii  apache2                        2.2.22-6ubuntu5.1    i386
Apache HTTP Server metapackage
```

3. Remove a Package

To remove the “**.deb**” package, we must specify the package name “**flashpluginnonfree**”, not the original name “**flashplugin-nonfree_3.2_i386.deb**”. The “**-r**” option is used to **remove/uninstall** a package.

```
[root@tecmint~]# dpkg -r flashpluginnonfree  
(Reading database ... 142891 files and directories currently installed.)  
Removing flashpluginnonfree ...  
Processing triggers for man-db ...  
Processing triggers for menu ...  
Processing triggers for desktop-file-utils ...  
Processing triggers for gnome-menus ...
```

You can also use ‘**p**’ option in place of ‘**r**’ which will remove the package along with configuration file. The ‘**r**’ option will only remove the package and not configuration files.

```
[root@tecmint~]# dpkg -p flashpluginnonfree
```

4. View the Content of a Package

To view the content of a particular package, use the “**-c**” option as shown. The command will display the contents of a “**.deb**” package in long-list format.

```
[root@tecmint~]# dpkg -c flashplugin-nonfree_3.2_i386.deb  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/bin/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/lib/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/lib/mozilla/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/lib/mozilla/plugins/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/lib/flashplugin-  
nonfree/  
-rw-r--r-- root/root      3920 2009-09-09 22:51 ./usr/lib/flashplugin-  
nonfree/pubkey.asc  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/share/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/share/man/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/share/man/man8/  
-rw-r--r-- root/root      716 2012-12-14 22:54 ./usr/share/man/man8/update-  
flashplugin-nonfree.8.gz  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/share/applications/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/share/icons/  
drwxr-xr-x root/root          0 2012-12-14 22:54 ./usr/share/icons/hicolor/  
drwxr-xr-x root/root          0 2012-12-14 22:54  
./usr/share/icons/hicolor/24x24/  
....
```

5. Check a Package is installed or not

Using “**-s**” option with package name, will display whether an deb package installed or not.

```
[root@tecmint~]# dpkg -s flashplugin-nonfree  
Package: flashplugin-nonfree  
Status: install ok installed
```

```
Priority: optional
Section: contrib/web
Installed-Size: 177
Maintainer: Bart Martens <bartm@debian.org>
Architecture: i386
Version: 1:3.2
Replaces: flashplugin (<< 6)
Depends: debconf | debconf-2.0, wget, gnupg, libatk1.0-0, libcairo2,
libfontconfig1, libfreetype6, libgcc1, libglib2.0-0, libgtk2.0-0 (>= 2.14),
libnspr4, libnss3, libpango1.0-0, libstdc++6, libx11-6, libxext6, libxt6,
libcurl3-gnutls, binutils
Suggests: iceweasel, konqueror-nspugins, ttf-mscorefonts-installer, ttf-
dejavu, ttf-xfree86-nonfree, flashplugin-nonfree-extrasound, hal
Conflicts: flashplayer-mozilla, flashplugin (<< 6), libflash-mozplugin, xfs
(<< 1:1.0.1-5)
Description: Adobe Flash Player - browser plugin
...

```

6. Check the location of Packages installed

To list location of files to be installed to your system from package-name.

```
[root@tecmint~]# dpkg -L flashplugin-nonfree
/.
/usr
/usr/bin
/usr/lib
/usr/lib/mozilla
/usr/lib/mozilla/plugins
/usr/lib/flashplugin-nonfree
/usr/lib/flashplugin-nonfree/pubkey.asc
/usr/share
/usr/share/man
/usr/share/man/man8
/usr/share/man/man8/update-flashplugin-nonfree.8.gz
/usr/share/applications
/usr/share/icons
/usr/share/icons/hicolor
...

```

7. Install all Packages from a Directory

Recursively, install all the regular files matching pattern “*.deb” found at specified directories and all of its subdirectories. This can be used with “-R” and “--install” options. For example, I will install all the “.deb” packages from the directory called “**debpackages**“.

```
[root@tecmint~]# dpkg -R --install debpackages/
(Reading database ... 465836 files and directories currently installed.)
Preparing to replace flashplugin-nonfree 1:3.2 (using .../flashplugin-
nonfree_3.2_i386.deb) ...
Unpacking replacement flashplugin-nonfree ...
Setting up flashplugin-nonfree (1:3.2) ...
Processing triggers for man-db ...
Processing triggers for bamfdaemon ...

```

```
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for gnome-menus ...
```

8. Unpack the Package but dont' Configure

Using action “**–unpack**” will unpack the package, but it will don't install or configure it.

```
[root@tecmint~]# dpkg --unpack flashplugin-nonfree_3.2_i386.deb
(Reading database ... 465836 files and directories currently installed.)
Preparing to replace flashplugin-nonfree 1:3.2 (using flashplugin-
nonfree_3.2_i386.deb) ...
Unpacking replacement flashplugin-nonfree ...
Processing triggers for man-db ...
Processing triggers for bamfdaemon ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for gnome-menus ...
```

9. Reconfigure a Unpacked Package

The option “**–configure**” will reconfigure a already unpacked package.

```
[root@tecmint~]# dpkg --configure flashplugin-nonfree
Setting up flashplugin-nonfree (1:3.2) ...
```

10. Replace available Package information

The “**--update-avail**” option replace the old information with the available information in the Packages file.

```
[root@tecmint~]# dpkg --update-avail package_name
```

11. Erase Existing Available information of Package

The action “**--clear-avaial**” will erase the current information about what packages are available.

```
[root@tecmint~]# dpkg --clear-avail
```

12. Forget Uninstalled and Unavailable Packages

The dpkg command with option “**--forget-old-unavail**” will automatically forget uninstalled and unavailable packages .

```
[root@tecmint~]# dpkg --forget-old-unavail
```

13. Display dpkg Licence

```
[root@tecmint~]# dpkg --licence
```

14. Display dpkg Version

The “**–version**” argument will display dpkg version information.

```
[root@tecmint~]# dpkg --version
Debian `dpkg' package management program version 1.16.10 (i386).
This is free software; see the GNU General Public License version 2 or
later for copying conditions. There is NO warranty.
```

15. Get all the Help about dpkg

The “**–help**” option will display a list of available options of dpkg command.

```
[root@tecmint~]# dpkg --help
Usage: dpkg [<option> ...] <command>
Commands:
-i|--install      <.deb file name> ... | -R|--recursive <directory> ...
--unpack          <.deb file name> ... | -R|--recursive <directory> ...
-A|--record-avail <.deb file name> ... | -R|--recursive <directory> ...
--configure        <package> ... | -a|--pending
--triggers-only   <package> ... | -a|--pending
-r|--remove        <package> ... | -a|--pending
-P|--purge         <package> ... | -a|--pending
--get-selections  [<pattern> ...] Get list of selections to stdout.
--set-selections   Set package selections from stdin.
--clear-selections Deselect every non-essential package.
--update-avail    <Packages-file> Replace available packages info.
--merge-avail     <Packages-file> Merge with info from file.
--clear-avail      Erase existing available info.
--forget-old-unavail Forget uninstalled unavailable pkgs.
-s|--status        <package> ...       Display package status details.
...
```

15 Examples of How to Use New Advanced Package Tool (APT) in Ubuntu/Debian

by [Aaron Kili](#) | Published: April 15, 2016 | Last Updated: April 15, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

One important thing to master under Linux System/Server Administration is package management using different package management tools.

Different Linux distributions install applications in a pre-compiled package that contain binary files, configuration files and also information about the application's dependencies.

Read Also: [Learn 25 ‘apt-get’ and ‘apt-cache’ Command Examples in Debian based Systems](#)

Package management tools help System/Server Administrators in many ways such as:

1. Downloading and installing software
2. Compile software from source
3. Keeping track of all software installed, their updates and upgrades
4. Handling dependencies
5. and also keeping other information about installed software and many more

In this guide, we are going to look at **15** examples of how to use the new **APT** (Advanced Package Tool) on your Ubuntu Linux systems.

APT is a command-line based tool that is used for dealing with packages on a Ubuntu based Linux systems. It presents a command line interface to the package management on your system.

1. Installing a Package

You can install a package as follows by specify a single package name or install many packages at once by listing all their names.

```
$ sudo apt install glances
```

```
tecmint@tecmint ~ $ sudo apt install glances
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required
  putty-tools
Use 'apt-get autoremove' to remove it.
Recommended packages:
  python-jinja2
The following NEW packages will be installed:
  glances
0 upgraded, 1 newly installed, 0 to remove and 230 not upgraded.
Need to get 0 B/509 kB of archives.
After this operation, 1,040 kB of additional disk space will be used.
Selecting previously unselected package glances.
(Reading database ... 234340 files and directories currently installed.)
Preparing to unpack .../glances_1.7.3-2ubuntu1_all.deb ...
Unpacking glances (1.7.3-2ubuntu1) ...
Processing triggers for ureadahead (0.100.0-16) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up glances (1.7.3-2ubuntu1) ...
 * Starting Glances server glances
 * Not starting glances: disabled by /etc/default/glances.
tecmint@tecmint ~ $ |
```

Install a Package

2. Find Location of Installed Package

The following command will help you to list all the files that are contained in a package called [glances \(advance Linux monitoring tool\)](#).

```
$ sudo apt content glances
```

```
tecmint@tecmint ~ $ sudo apt content glances
/.
/etc
/etc/default
/etc/default/glances
/etc/glances
/etc/glances/glances.conf
/etc/init.d
/etc/init.d/glances
/usr
/usr/bin
/usr/bin/glances
/usr/lib
/usr/lib/python2.7
/usr/lib/python2.7/dist-packages
/usr/lib/python2.7/dist-packages/glances
/usr/lib/python2.7/dist-packages/Glances-1.7.3.egg-info
/usr/lib/python2.7/dist-packages/Glances-1.7.3.egg-info/dependency_links
/usr/lib/python2.7/dist-packages/Glances-1.7.3.egg-info/entry_points.txt
/usr/lib/python2.7/dist-packages/Glances-1.7.3.egg-info/PKG-INFO
/usr/lib/python2.7/dist-packages/Glances-1.7.3.egg-info/requirements.txt
/usr/lib/python2.7/dist-packages/Glances-1.7.3.egg-info/top_level.txt
/usr/lib/python2.7/dist-packages/glances/data
/usr/lib/python2.7/dist-packages/glances/data/css
/usr/lib/python2.7/dist-packages/glances/data/css/default.css
/usr/lib/python2.7/dist-packages/glances/data/html
/usr/lib/python2.7/dist-packages/glances/data/html/base.html
/usr/lib/python2.7/dist-packages/glances/data/html/default.html
/usr/lib/python2.7/dist-packages/glances/data/img
```

Find Installed Package Files Location

3. Check All Dependencies of a Package

This will help you to display raw information about dependencies of a particular package that you specify.

```
$ sudo apt depends glances
```

```
tecmint@tecmint ~ $ sudo apt depends glances
glances
Depends: python-psutil
Depends: <python:any>
  python:i386
  python
Depends: <python:any>
  python:i386
  python
Depends: python
Depends: python-pkg-resources
Depends: adduser
Depends: lsb-base
Recommends: python-jinja2
tecmint@tecmint ~ $ |
```

Check Dependencies of Package

4. Search for a Package

The **search** option searches for the given package name and show all the matching packages.

```
$ sudo apt search apache2
```

```
tecmint@tecmint ~ $ sudo apt search apache2
ih  apache2                               - Apache HTTP Server
p   apache2:i386                            - Apache HTTP Server
v   apache2-api-20120211                   -
v   apache2-api-20120211:i386               -
i A  apache2-bin                           - Apache HTTP Server (binary files a
p   apache2-bin:i386                         - Apache HTTP Server (binary files a
i A  apache2-data                           - Apache HTTP Server (common files)
v   apache2-data:i386                        -
p   apache2-dbg                            - Apache debugging symbols
p   apache2-dbg:i386                         - Apache debugging symbols
p   apache2-dev                            - Apache HTTP Server (development he
p   apache2-dev:i386                         - Apache HTTP Server (development he
p   apache2-doc                            - Apache HTTP Server (on-site docume
p   apache2-mpm-event                      - transitional event MPM package fo
p   apache2-mpm-event:i386                  - transitional event MPM package fo
p   apache2-mpm-itk                          - transitional itk MPM package for a
p   apache2-mpm-itk:i386                    - transitional itk MPM package for a
p   apache2-mpm-prefork                     - transitional prefork MPM package f
p   apache2-mpm-prefork:i386                 - transitional prefork MPM package f
p   apache2-mpm-worker                      - transitional worker MPM package fo
p   apache2-mpm-worker:i386                  - transitional worker MPM package fo
v   apache2-prefork-dev                     -
v   apache2-prefork-dev:i386                 -
p   apache2-suexec                         - transitional package for apache2-s
p   apache2-suexec:i386                     - transitional package for apache2-s
p   apache2-suexec-custom                  - Apache HTTP Server configurable su
```

Search For a Package

5. View Information About Package

This will help you display information about package or packages, run the command below by specifying all the packages that you want to display information about.

```
$ sudo apt show firefox
```

```
tecmint@tecmint ~ $ sudo apt show firefox
Package: firefox
State: installed
Automatically installed: no
Version: 43.0+linuxmint1+rosa
Priority: optional
Section: web
Maintainer: Ubuntu Mozilla Team <ubuntu-mozillateam@lists.ubuntu.com>
Architecture: amd64
Uncompressed Size: 106 M
Depends: lsb-release, libasound2 (>= 1.0.16), libatk1.0-0 (>= 1.12.4), libatkmm-1.0-0 (>= 2.17), libcairo2 (>= 1.2.4), libdbus-1-3 (>= 1.0.2), libdbus-glib-1-0 (>= 0.78), libfontconfig1 (>= 2.9.0), libfreetype6 (>= 2.2.1), libgcc1 (>= 1:4.1.1), libgdk-pixbuf2.0-0 (>= 2.22.0), libglib2.0-0 (>= 2.42.0), libgtk2.0-0 (>= 2.24.0), libpango-1.0-0 (>= 1.22.0), libpangocairo-1.0-0 (>= 1.14.0), libstartup-notification0 (>= 0.12.0), libstdc++6 (>= 4.6), libx11-6, libxcomposite1 (>= 1:0.3-1), libxext6 (>= 1:1.1), libxfixes3, libxrender1, libxt6
Recommends: xul-ext-ubufox, libcanberra0, libdbusmenu-glib4, libdbusmenu-glib4:i386
Suggests: ttf-lyx
Conflicts: firefox
Replaces: kubuntu-firefox-installer, kubuntu-firefox-installer
Provides: gnome-www-browser, iceweasel, www-browser
Description: Safe and easy web browser from Mozilla
  Firefox delivers safe, easy web browsing. A familiar user interface, enhanced security features including protection from online identity theft, and integrated search let you get the most out of the web.
```

Show Package Information

6. Verify a Package for any Broken Dependencies

Sometimes during package installation, you may get errors concerning broken package dependencies, to check that you do not have these problems run the command below with the package name.

```
$ sudo apt check firefox
tecmint@tecmint ~ $ sudo apt check firefox
Reading package lists... Done
Building dependency tree
Reading state information... Done
tecmint@tecmint ~ $ |
```

Check Package for Broke Dependencies

7. List Recommended Missing Packages of Given Package

```
$ sudo apt recommends apache2
tecmint@tecmint ~ $ sudo apt recommends apache2
No missing recommended packages were found for apache2
```

View Recommended Missing Packages

8. Check Installed Package Version

The ‘version’ option will show you the installed package version.

```
$ sudo apt version firefox
tecmint@tecmint ~ $ sudo apt version firefox
43.0+linuxmint1+rosa
tecmint@tecmint ~ $ sudo apt version apache2
2.4.7-1ubuntu4.8
tecmint@tecmint ~ $ sudo apt version perl
5.18.2-2ubuntu1
tecmint@tecmint ~ $ |
```

Check Installed Package Version

9. Update System Packages

This will help you to download a list of packages from different repositories included on your system and updates them when there are new versions of packages and their dependencies.

```
$ sudo apt update
```

```
tecmint@tecmint ~ $ sudo apt update
Hit http://download.virtualbox.org trusty InRelease
Hit http://download.virtualbox.org trusty/contrib amd64 Packages
Hit http://download.virtualbox.org trusty/contrib i386 Packages
Ign http://download.virtualbox.org trusty/contrib Translation-en_IN
Ign http://download.virtualbox.org trusty/contrib Translation-en
Ign http://dl.google.com stable InRelease
Hit http://dl.google.com stable Release.gpg
Hit http://dl.google.com stable Release
Hit http://dl.google.com stable/main amd64 Packages
Get:1 http://security.ubuntu.com trusty-security InRelease [65.9 kB]
Ign http://archive.ubuntu.com trusty InRelease
Hit http://ppa.launchpad.net trusty InRelease
Ign http://packages.linuxmint.com rosa InRelease
Ign http://extra.linuxmint.com rosa InRelease
Get:2 http://archive.ubuntu.com trusty-updates InRelease [65.9 kB]
Hit http://ppa.launchpad.net trusty InRelease
Hit http://packages.linuxmint.com rosa Release.gpg
Hit http://extra.linuxmint.com rosa Release.gpg
Hit http://ppa.launchpad.net trusty InRelease
Get:3 http://security.ubuntu.com trusty-security/main amd64 Packages [45
```

Update System Packages

10. Upgrade System

This helps you to install new versions of all the packages on your system.

```
$ sudo apt upgrade
```

```
tecmint@tecmint ~ $ sudo apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following package was automatically installed and is no longer required
  libvncclient0
Use 'apt-get autoremove' to remove it.
The following packages have been kept back:
  apache2 apache2-bin apache2-data libdrm-dev libdrm-intel libdrm-intel-
  libdrm-nouveau libdrm-nouveau:i386 libdrm-radeon1 libdrm-radeon1:i386
  libdrm2 libdrm2:i386
The following packages will be upgraded:
  apache2-utils apt-transport-https apt-utils atril atril-common base-files
  bind9-host ca-certificates caja caja-common coreutils cpio cpp-4.8 curl
  dnsutils ecryptfs-utils eom eom-common ffmpeg firefox firefox-locale-en
  flashplugin-installer g++-4.8 gcc-4.8 gcc-4.8-base gcc-4.8-base:i386
  gcc-4.9-base gcc-4.9-base:i386 gir1.2-caja gir1.2-gtk-2.0 gir1.2-ibus-
  gir1.2-javascriptcoregtk-3.0 gir1.2-mate-panel gir1.2-webkit-3.0 git-manual
  glib-networking glib-networking:i386 glib-networking-common
  glib-networking-services google-chrome-stable gtk2-engines-pixbuf
  gtk2-engines-pixbuf:i386 hexchat hexchat-common ibus-gtk:i386 ifupdown
```

Upgrade System

11. Remove Unused Packages

When you install a new package on your system, its dependencies are also installed and they use some system libraries with other packages. After removing that particular package, its dependencies will remain on the system, therefore to remove them use autoremove as follows:

```
$ sudo apt autoremove
tecmint@tecmint ~ $ sudo apt autoremove
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 229 not upgraded.
tecmint@tecmint ~ $ |
```

Remove Unwanted Packages

12. Clean Old Repository of Downloaded Packages

The option ‘clean’ or ‘autoclean’ remove all old local repository of downloaded package files.

```
$ sudo apt autoclean
or
```

```
$ sudo apt clean  
tecmint@tecmint ~ $ sudo apt autoclean  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
tecmint@tecmint ~ $ sudo apt clean  
tecmint@tecmint ~ $ |
```

Clean Package Repository

13. Remove Packages with its Configuration Files

When you run **apt** with **remove**, it only removes the package files but configuration files remain on the system. Therefore to remove a package and its configuration files, you will have to use **purge**.

```
$ sudo apt purge glances  
tecmint@tecmint ~ $ sudo apt purge glances  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following packages will be REMOVED:  
  glances*  
0 upgraded, 0 newly installed, 1 to remove and 229 not upgraded.  
After this operation, 1,040 kB disk space will be freed.  
Do you want to continue? [Y/n] y  
(Reading database ... 234394 files and directories currently installed.)  
Removing glances (1.7.3-2ubuntu1) ...  
 * Stopping Glances server glances  
 * PID file not found  
Purging configuration files for glances (1.7.3-2ubuntu1) ...  
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...  
tecmint@tecmint ~ $ |
```

Remove Package Configuration Files

14. Install .Deb Package

To install a **.deb** file, run the command below with the filename as an argument as follows:

```
$ sudo apt deb atom-amd64.deb
```

```
tecmint@tecmint ~ $ sudo apt deb atom-amd64.deb
Selecting previously unselected package atom.
(Reading database ... 234336 files and directories currently installed.)
Preparing to unpack atom-amd64.deb ...
Unpacking atom (1.6.2) ...
Setting up atom (1.6.2) ...
Processing triggers for mime-support (3.54ubuntu1.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu1) ...
tecmint@tecmint ~ $ |
```

Install Deb Package

15. Find Help While Using APT

The following command will list you all the options with it's description on how to use **APT** on your system.

```
$ apt help
```

```
tecmint@tecmint ~ $ apt help
apt
Usage: apt command [options]
       apt help command [options]

Commands:
add-repository      - Add entries to apt sources.list
autoclean           - Erase old downloaded archive files
autoremove          - Remove automatically all unused packages
build               - Build binary or source packages from sources
build-dep           - Configure build-dependencies for source packages
changelog           - View a package's changelog
check               - Verify that there are no broken dependencies
clean               - Erase downloaded archive files
contains            - List packages containing a file
content             - List files contained in a package
deb                 - Install a .deb package
depends             - Show raw dependency information for a package
dist-upgrade        - Perform an upgrade, possibly installing and removing p
download            - Download the .deb file for a package
dselect-upgrade     - Follow dselect selections
held                - List all held packages
help                - Show help for a command
hold                - Hold a package
install             - Install/upgrade packages
policy              - Show policy settings
purge               - Remove packages and their configuration files
recommends          - List missing recommended packages for a particular pac
rdepends            - Show reverse dependency information for a package
reinstall            - Download and (possibly) reinstall a currently installed
remove              - Remove packages
search              - Search for a package by name and/or expression
show                - Display detailed information about a package
source              - Download source archives
sources             - Edit /etc/apt/sources.list with nano
unhold              - Unhold a package
update              - Download lists of new/upgradable packages
upgrade             - Perform a safe upgrade
version             - Show the installed version of a package
This apt has Super Cow Powers
tecmint@tecmint ~ $ |
```

APT Command Help

Summary

Always remember that good [Linux package management](#), can help you avoid breaking your system. There are so many other [package management tools](#) that you can use in Linux.

You can share with us what you use and your experience with it. I hope the article is helpful and for any additional information, leave a comment in the comment section.

7 Ways to Determine the File System Type in Linux (Ext2, Ext3 or Ext4)

by [Aaron Kili](#) | Published: March 4, 2017 | Last Updated: March 4, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

A file system is the way in which files are named, stored, retrieved as well as updated on a storage disk or partition; the way files are organized on the disk.

A file system is divided in two segments called: **User Data** and **Metadata** (file name, time it was created, modified time, it's size and location in the directory hierarchy etc).

In this guide, we will explain seven ways to identify your Linux file system type such as Ext2, Ext3, Ext4, Btrfs, GlusterFS plus many more.

1. Using df Command

df command reports file system disk space usage, to include the file system type on a particular disk partition, use the **-T** flag as below:

```
$ df -Th  
OR  
$ df -Th | grep '^/dev'  
tecmint@TecMint ~ $ df -Th  


| Filesystem | Type     | Size | Used | Avail | Use% | Mounted on        |
|------------|----------|------|------|-------|------|-------------------|
| udev       | devtmpfs | 3.9G | 0    | 3.9G  | 0%   | /dev              |
| tmpfs      | tmpfs    | 788M | 9.6M | 779M  | 2%   | /run              |
| /dev/sda10 | ext4     | 324G | 202G | 106G  | 66%  | /                 |
| tmpfs      | tmpfs    | 3.9G | 118M | 3.8G  | 3%   | /dev/shm          |
| tmpfs      | tmpfs    | 5.0M | 4.0K | 5.0M  | 1%   | /run/lock         |
| tmpfs      | tmpfs    | 3.9G | 0    | 3.9G  | 0%   | /sys/fs/cgroup    |
| cgmfs      | tmpfs    | 100K | 0    | 100K  | 0%   | /run/cgmanager/fs |
| tmpfs      | tmpfs    | 788M | 36K  | 788M  | 1%   | /run/user/1000    |

tecmint@TecMint ~ $  
tecmint@TecMint ~ $ df -Th | grep '^/dev'  
/dev/sda10 ext4 324G 202G 106G 66% /  
tecmint@TecMint ~ $ █
```

df Command – Find Filesystem Type

For a comprehensive guide for **df command** usage go through our articles:

1. [12 Useful “df” Commands to Check Disk Space in Linux](#)
2. [Pydf – An Alternative ‘df’ Command That Shows Disk Usage in Colours](#)

2. Using fsck Command

fsck is used to check and optionally [repair Linux file systems](#), it can also print the [file system type on specified disk partitions](#).

The flag **-N** disables checking of file system for errors, it just shows what would be done (but all we need is the file system type):

```
$ fsck -N /dev/sda3
$ fsck -N /dev/sdb1
tecmint@TecMint ~ $ fsck -N /dev/sda3
fsck from util-linux 2.27.1
[/sbin/fsck.ext2 (1) -- /dev/sda3] fsck.ext2 /dev/sda3
tecmint@TecMint ~ $
tecmint@TecMint ~ $ fsck -N /dev/sdb1
fsck from util-linux 2.27.1
[/sbin/fsck.ext2 (1) -- /dev/sdb1] fsck.ext2 /dev/sdb1
tecmint@TecMint ~ $
```

fsck – Print Linux Filesystem Type

3. Using lsblk Command

lsblk displays block devices, when used with the **-f** option, it prints file system type on partitions as well:

```
$ lsblk -f
tecmint@TecMint ~ $ lsblk -f
NAME   FSTYPE LABEL      UUID                                     MOUNTPOINT
sda
├─sda1  ntfs   WINRE_DRV  D4A45AAAA45A8EBC
├─sda2  vfat    SYSTEM_DRV 185C-DA5B
├─sda3  vfat    LRS_ESP    0E60-2E0E
├─sda4
├─sda5  ntfs   Windows8_OS 18D0632AD0630CF6
├─sda6  ntfs   LENOVO     9286FFD986FFBC33
├─sda7  ntfs   PBR_DRV    ECD06683D066543C
├─sda8
├─sda9  swap    e040de62-c837-453e-88ee-bd9000387083 [SWAP]
└─sda10 ext4    bb29ddaa3-bdaa-4b39-86cf-4a6dc9634a1b /
sr0
tecmint@TecMint ~ $
```

`lsblk` – Shows Linux Filesystem Type

4. Using mount Command

mount command is used to [mount a file system in Linux](#), it can also be used to [mount an ISO image](#), [mount remote Linux filesystem](#) and so much more.

When run without any arguments, it prints [info about disk partitions](#) including the file system type as below:

```
$ mount | grep '^/dev'  
tecmint@TecMint ~ $  
tecmint@TecMint ~ $ mount | grep '^/dev'  
/dev/sda10 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)  
tecmint@TecMint ~ $  
tecmint@TecMint ~ $
```

`Mount` – Show Filesystem Type in Linux

5. Using blkid Command

blkid command is used to [find or print block device properties](#), simply specify the disk partition as an argument like so:

```
$ blkid /dev/sda3  
tecmint@TecMint ~ $ blkid /dev/sda3  
/dev/sda3: LABEL="LRS_ESP" UUID="0E60-2E0E" TYPE="vfat" PARTLABEL="Basic data partition" PARTUUID="d464feab-0791-4866-a36b-90dbe6d6a437"  
tecmint@TecMint ~ $ blkid /dev/sda10  
/dev/sda10: UUID="bb29dda3-bdaa-4b39-86cf-4a6dc9634a1b" TYPE="ext4" PARTUUID="26b60905-1c39-4fd4-bdce-95c517c781fa"  
tecmint@TecMint ~ $  
tecmint@TecMint ~ $
```

`blkid` – Find Filesystem Type

6. Using file Command

file command identifies file type, the `-s` flag enables reading of block or character files and `-L` enables following of symlinks:

```
$ sudo file -sL /dev/sda3
```

```
tecmint@TecMint ~ $ sudo file -sL /dev/sda3
/dev/sda3: DOS/MBR boot sector, code offset 0x58+2, OEM-ID "MSDOS5.0", sectors/c
reserved sectors 4206, Media descriptor 0xf8, sectors/track 63, heads 255, hidden
2582528, sectors 2048000 (volumes > 32 MB) , FAT (32 bit), sectors/FAT 1993, ser
0xe602e0e, unlabeled
tecmint@TecMint ~ $
tecmint@TecMint ~ $ sudo file -sL /dev/sda10
/dev/sda10: Linux rev 1.0 ext4 filesystem data, UUID=bb29dda3-bdaa-4b39-86cf-4a6
(needs journal recovery) (extents) (large files) (huge files)
tecmint@TecMint ~ $
tecmint@TecMint ~ $
tecmint@TecMint ~ $
```

file – Identifies Filesystem Type

7. Using fstab File

The **/etc/fstab** is a static file system info (such as mount point, file system type, mount options etc) file:

```
$ cat /etc/fstab
tecmint@TecMint ~ $ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>           <dump> <pass>
# / was on /dev/sda10 during installation
UUID=bb29dda3-bdaa-4b39-86cf-4a6dc9634a1b /           ext4   errors=remount
    1
# swap was on /dev/sda9 during installation
UUID=e040de62-c837-453e-88ee-bd9000387083 none        swap   sw
0
```

Fstab – Shows Linux Filesystem Type

That's it! In this guide, we explained seven ways to identify your Linux file system type. Do you know of any method not mentioned here? Share it with us in the comments.

12 Useful “df” Commands to Check Disk Space in Linux

by [Ravi Saive](#) | Published: January 15, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

On the internet you will find plenty of tools for checking disk space utilization in Linux. However, Linux has a strong built in utility called ‘**df**’. The ‘**df**’ command stand for “**disk filesystem**”, it is used to get full summary of available and used disk space usage of file system on Linux system.

Using ‘**-h**’ parameter with (**df -h**) will shows the file system disk space statistics in “**human readable**” format, means it gives the details in bytes, mega bytes and gigabyte.



Useful df Command Examples

This article explain a way to get the full information of Linux disk space usage with the help of ‘**df**’ command with their practical examples. So, you could better understand the usage of **df** command in Linux.

1. Check File System Disk Space Usage

The “**df**” command displays the information of device name, total blocks, total disk space, used disk space, available disk space and mount points on a file system.

```
[root@tecmint ~]# df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/cciss/c0d0p2   78361192  23185840  51130588  32% /
/dev/cciss/c0d0p5   24797380  22273432  1243972  95% /home
/dev/cciss/c0d0p3   29753588  25503792  2713984  91% /data
/dev/cciss/c0d0p1   295561      21531   258770   8% /boot
```

```
tmpfs          257476        0     257476    0% /dev/shm
```

2. Display Information of all File System Disk Space Usage

The same as above, but it also displays information of dummy file systems along with all the file system disk usage and their memory utilization.

```
[root@tecmint ~]# df -a
Filesystem      1K-blocks   Used   Available  Use% Mounted on
/dev/cciss/c0d0p2    78361192 23186116 51130312  32% /
proc                  0       0       0       -  /proc
sysfs                 0       0       0       -  /sys
devpts                 0       0       0       -  /dev/pts
/dev/cciss/c0d0p5    24797380 22273432 1243972  95% /home
/dev/cciss/c0d0p3    29753588 25503792 2713984  91% /data
/dev/cciss/c0d0p1    295561    21531   258770   8% /boot
tmpfs                  257476     0     257476    0% /dev/shm
none                   0       0       0       - 
/proc/sys/fs/binfmt_misc
sunrpc                  0       0       0       - 
/var/lib/nfs/rpc_pipefs
```

3. Show Disk Space Usage in Human Readable Format

Have you noticed that above commands displays information in bytes, which is not readable yet all, because we are in a habit of reading the sizes in megabytes, gigabytes etc. as it makes very easy to understand and remember.

The **df** command provides an option to display sizes in **Human Readable** formats by using ‘-h’ (prints the results in human readable format (e.g., 1K 2M 3G)).

```
[root@tecmint ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/cciss/c0d0p2    75G  23G  49G  32% /
/dev/cciss/c0d0p5    24G  22G  1.2G  95% /home
/dev/cciss/c0d0p3    29G  25G  2.6G  91% /data
/dev/cciss/c0d0p1   289M  22M  253M   8% /boot
tmpfs                  252M     0  252M   0% /dev/shm
```

4. Display Information of /home File System

To see the information of only device **/home** file system in human readable format use the following command.

```
[root@tecmint ~]# df -hT /home
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/cciss/c0d0p5  ext3  24G  22G  1.2G  95% /home
```

5. Display Information of File System in Bytes

To display all file system information and usage in **1024-byte** blocks, use the option ‘**-k**’ (e.g. –block-size=1K) as follows.

```
[root@tecmint ~]# df -k
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/cciss/c0d0p2    78361192 23187212 51129216 32% /
/dev/cciss/c0d0p5    24797380 22273432 1243972 95% /home
/dev/cciss/c0d0p3    29753588 25503792 2713984 91% /data
/dev/cciss/c0d0p1     295561   21531   258770   8% /boot
tmpfs                  257476       0   257476   0% /dev/shm
```

6. Display Information of File System in MB

To display information of all file system usage in **MB (Mega Byte)** use the option as ‘**-m**’.

```
[root@tecmint ~]# df -m
Filesystem      1M-blocks   Used Available Use% Mounted on
/dev/cciss/c0d0p2    76525   22644   49931 32% /
/dev/cciss/c0d0p5    24217   21752   1215 95% /home
/dev/cciss/c0d0p3    29057   24907   2651 91% /data
/dev/cciss/c0d0p1     289     22     253   8% /boot
tmpfs                  252       0     252   0% /dev/shm
```

7. Display Information of File System in GB

To display information of all file system statistics in **GB (Gigabyte)** use the option as ‘**df -h**’.

```
[root@tecmint ~]# df -h
Filesystem      Size   Used Avail Use% Mounted on
/dev/cciss/c0d0p2    75G   23G   49G 32% /
/dev/cciss/c0d0p5    24G   22G   1.2G 95% /home
/dev/cciss/c0d0p3    29G   25G   2.6G 91% /data
/dev/cciss/c0d0p1   289M   22M   253M 8% /boot
tmpfs                  252M       0   252M 0% /dev/shm
```

8. Display File System Inodes

Using ‘**-i**’ switch will display the information of number of used inodes and their percentage for the file system.

```
[root@tecmint ~]# df -i
Filesystem      Inodes   IUsed   IFree  IUse% Mounted on
/dev/cciss/c0d0p2  20230848 133143 20097705   1% /
/dev/cciss/c0d0p5  6403712  798613 5605099  13% /home
/dev/cciss/c0d0p3  7685440 1388241 6297199  19% /data
/dev/cciss/c0d0p1   76304      40   76264   1% /boot
tmpfs                  64369       1   64368   1% /dev/shm
```

9. Display File System Type

If you notice all the above commands output, you will see there is no file system type mentioned in the results. To check the file system type of your system use the option ‘**T**’. It will display file system type along with other information.

```
[root@tecmint ~]# df -T
Filesystem      Type  1K-blocks  Used   Available Use% Mounted on
/dev/cciss/c0d0p2  ext3    78361192 23188812 51127616  32%   /
/dev/cciss/c0d0p5  ext3    24797380 22273432 1243972  95%   /home
/dev/cciss/c0d0p3  ext3    29753588 25503792 2713984  91%   /data
/dev/cciss/c0d0p1  ext3    295561     21531   258770   8%    /boot
tmpfs            tmpfs   257476      0    257476   0%    /dev/shm
```

10. Include Certain File System Type

If you want to display certain file system type use the ‘**-t**’ option. For example, the following command will only display **ext3** file system.

```
[root@tecmint ~]# df -t ext3
Filesystem      1K-blocks  Used   Available Use% Mounted on
/dev/cciss/c0d0p2  78361192 23190072 51126356  32%   /
/dev/cciss/c0d0p5  24797380 22273432 1243972  95%   /home
/dev/cciss/c0d0p3  29753588 25503792 2713984  91%   /data
/dev/cciss/c0d0p1  295561     21531   258770   8%    /boot
```

11. Exclude Certain File System Type

If you want to display file system type that doesn’t belongs to **ext3** type use the option as ‘**-x**’. For example, the following command will only display other file systems types other than **ext3**.

```
[root@tecmint ~]# df -x ext3
Filesystem      1K-blocks  Used   Available Use% Mounted on
tmpfs           257476      0    257476   0%    /dev/shm
```

12. Display Information of df Command.

Using ‘**–help**’ switch will display a list of available option that are used with **df** command.

```
[root@tecmint ~]# df --help
Usage: df [OPTION]... [FILE]...
Show information about the file system on which each FILE resides,
or all file systems by default.
Mandatory arguments to long options are mandatory for short options too.
-a, --all          include dummy file systems
-B, --block-size=SIZE use SIZE-byte blocks
-h, --human-readable print sizes in human readable format (e.g., 1K 234M 2G)
-H, --si           likewise, but use powers of 1000 not 1024
-i, --inodes       list inode information instead of block usage
-k                like --block-size=1K
-l, --local        limit listing to local file systems
--no-sync         do not invoke sync before getting usage info (default)
-P, --portability  use the POSIX output format
```

```
--sync           invoke sync before getting usage info
-t, --type=TYPE    limit listing to file systems of type TYPE
-T, --print-type     print file system type
-x, --exclude-type=TYPE  limit listing to file systems not of type TYPE
-v                  (ignored)
--help      display this help and exit
--version   output version information and exit
SIZE may be (or may be an integer optionally followed by) one of following:
kB 1000, K 1024, MB 1000*1000, M 1024*1024, and so on for G, T, P, E, Z, Y.
Report bugs to <bug-coreutils@gnu.org>.
```

10 fdisk Commands to Manage Linux Disk Partitions

by [Ravi Saive](#) | Published: December 16, 2015 | Last Updated: December 16, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

fdisk stands (for “**fixed disk** or **format disk**“) is an most commonly used command-line based disk manipulation utility for a **Linux/Unix** systems. With the help of fdisk command you can view, create, resize, delete, change, copy and move partitions on a hard drive using its own user friendly text based menu driven interface.

This tool is very useful in terms of creating space for new partitions, organising space for new drives, re-organising an old drives and copying or moving data to new disks. It allows you to create a maximum of four new **primary** partition and number of logical (**extended**) partitions, based on size of the hard disk you have in your system.



fdisk command to manage disk partition

This article explains 10 basic **fdisk commands** to manage a partition table in Linux based systems. You must be **root** user to run fdisk command, otherwise you will get a “**command not found**” error.

Caution – Don't Create, Delete or Modify Partitions. Unless you know what you are doing!

1. View all Disk Partitions in Linux

The following basic command list all existing disk partition on your system. The ‘-l’ argument stand for (listing all partitions) is used with fdisk command to view all available partitions on Linux. The partitions are displayed by their device’s names. For example: **/dev/sda**, **/dev/sdb** or **/dev/sdc**.

```
[root@tecmint.com ~]# fdisk -l
Disk /dev/sda: 637.8 GB, 637802643456 bytes
255 heads, 63 sectors/track, 77541 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/sda1 * 1 13 104391 83 Linux
/dev/sda2 14 2624 20972857+ 83 Linux
/dev/sda3 2625 4582 15727635 83 Linux
/dev/sda4 4583 77541 586043167+ 5 Extended
/dev/sda5 4583 5887 10482381 83 Linux
/dev/sda6 5888 7192 10482381 83 Linux
/dev/sda7 7193 7845 5245191 83 Linux
/dev/sda8 7846 8367 4192933+ 82 Linux swap / Solaris
/dev/sda9 8368 77541 555640123+ 8e Linux LVM
```

2. View Specific Disk Partition in Linux

To view all partitions of specific hard disk use the option ‘-l’ with device name. For example, the following command will display all disk partitions of device **/dev/sda**. If you’ve different device names, simple write device name as **/dev/sdb** or **/dev/sdc**.

```
[root@tecmint.com ~]# fdisk -l /dev/sda
Disk /dev/sda: 637.8 GB, 637802643456 bytes
255 heads, 63 sectors/track, 77541 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/sda1 * 1 13 104391 83 Linux
/dev/sda2 14 2624 20972857+ 83 Linux
/dev/sda3 2625 4582 15727635 83 Linux
/dev/sda4 4583 77541 586043167+ 5 Extended
/dev/sda5 4583 5887 10482381 83 Linux
/dev/sda6 5888 7192 10482381 83 Linux
/dev/sda7 7193 7845 5245191 83 Linux
/dev/sda8 7846 8367 4192933+ 82 Linux swap / Solaris
/dev/sda9 8368 77541 555640123+ 8e Linux LVM
```

3. Check all Available fdisk Commands

If you would like to view all commands which are available for fdisk. Simply use the following command by mentioning the hard disk name such as **/dev/sda** as shown below. The following command will give you output similar to below.

```
[root@tecmint ~]# fdisk /dev/sda
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
Command (m for help):
```

Type '**m**' to see the list of all available commands of fdisk which can be operated on **/dev/sda** hard disk. After, I enter '**m**' on the screen, you will see the all available options for fdisk that you can be used on the **/dev/sda** device.

```
[root@tecmint ~]# fdisk /dev/sda
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
Command (m for help): m
Command action
a    toggle a bootable flag
b    edit bsd disklabel
c    toggle the dos compatibility flag
d    delete a partition
l    list known partition types
m    print this menu
n    add a new partition
o    create a new empty DOS partition table
p    print the partition table
q    quit without saving changes
s    create a new empty Sun disklabel
t    change a partition's system id
u    change display/entry units
v    verify the partition table
w    write table to disk and exit
x    extra functionality (experts only)
Command (m for help):
```

4. Print all Partition Table in Linux

To print all partition table of hard disk, you must be on command mode of specific hard disk say **/dev/sda**.

```
[root@tecmint ~]# fdisk /dev/sda
```

From the command mode, enter '**p**' instead of '**m**' as we did earlier. As I enter '**p**', it will print the specific **/dev/sda** partition table.

```
Command (m for help): p
Disk /dev/sda: 637.8 GB, 637802643456 bytes
255 heads, 63 sectors/track, 77541 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

```

Device Boot      Start         End      Blocks   Id  System
/dev/sda1    *          1         13     104391   83  Linux
/dev/sda2            14        2624    20972857+  83  Linux
/dev/sda3        2625        4582    15727635   83  Linux
/dev/sda4        4583       77541    586043167+   5  Extended
/dev/sda5        4583        5887    10482381   83  Linux
/dev/sda6        5888        7192    10482381   83  Linux
/dev/sda7        7193        7845     5245191   83  Linux
/dev/sda8        7846        8367    4192933+   82  Linux swap / Solaris
/dev/sda9        8368       77541    555640123+  8e  Linux LVM
Command (m for help):

```

5. How to Delete a Partition in Linux

If you would like to delete a specific partition (i.e **/dev/sda9**) from the specific hard disk such as **/dev/sda**. You must be in fdisk command mode to do this.

```
[root@tecmint ~]# fdisk /dev/sda
```

Next, enter '**d**' to delete any given partition name from the system. As I enter '**d**', it will prompt me to enter partition number that I want to delete from **/dev/sda** hard disk. Suppose I enter number '**4**' here, then it will delete partition number '**4**' (i.e. **/dev/sda4**) disk and shows free space in partition table. Enter '**w**' to write table to disk and exit after making new alterations to partition table. The new changes would only take place after next reboot of system. This can be easily understood from the below output.

```

[root@tecmint ~]# fdisk /dev/sda
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
Command (m for help): d
Partition number (1-4): 4
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
WARNING: Re-reading the partition table failed with error 16: Device or
resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
You have new mail in /var/spool/mail/root

```

Warning : Be careful, while performing this step, because using option '**d**' will completely delete partition from system and may lost all data in partition.

6. How to Create a New Partition in Linux

If you've free space left on one of your device say **/dev/sda** and would like to create a new partition under it. Then you must be in fdisk command mode of **/dev/sda**. Type the following command to enter into command mode of specific hard disk.

```
[root@tecmint ~]# fdisk /dev/sda
```

After entering in command mode, now press “**n**” command to create a new partition under **/dev/sda** with specific size. This can be demonstrated with the help of following given output.

```
[root@tecmint ~]# fdisk /dev/sda
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
e
```

While creating a new partition, it will ask you two options ‘**extended**‘ or ‘**primary**‘ partition creation. Press ‘**e**‘ for extended partition and ‘**p**‘ for primary partition. Then it will ask you to enter following two inputs.

1. First cylinder number of the partition to be create.
2. Last cylinder number of the partition to be created (Last cylinder, +cylinders or +size).

You can enter the size of cylinder by adding “**+5000M**” in last cylinder. Here, ‘+‘ means addition and **5000M** means size of new partition (i.e **5000MB**). Please keep in mind that after creating a new partition, you should run ‘**w**‘ command to alter and save new changes to partition table and finally reboot your system to verify newly created partition.

```
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
WARNING: Re-reading the partition table failed with error 16: Device or
resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

7. How to Format a Partition in Linux

After the new partition is created, don’t skip to format the newly created partition using ‘**mkfs**‘ command. Type the following command in the terminal to format a partition. Here **/dev/sda4** is my newly created partition.

```
[root@tecmint ~]# mkfs.ext4 /dev/sda4
```

8. How to Check Size of a Partition in Linux

After formatting new partition, check the size of that partition using flag ‘**s**‘ (displays size in blocks) with fdisk command. This way you can check size of any specific device.

```
[root@tecmint ~]# fdisk -s /dev/sda2
```

9. How to Fix Partition Table Order

If you've deleted a logical partition and again recreated it, you might notice '**partition out of order**' problem or error message like '**Partition table entries are not in disk order**'.

For example, when three logical partitions such as (**sda4**, **sda5** and **sda6**) are deleted, and new partition created, you might expect the new partition name would be **sda4**. But, the system would create it as **sda5**. This happens because of, after the partition are deleted, **sda7** partition had been moved as **sda4** and free space shift to the end.

To fix such partition order problems, and assign **sda4** to the newly created partition, issue the '**x**' to enter an extra functionality section and then enter '**f**' expert command to fix the order of partition table as shown below.

```
[root@tecmint ~]# fdisk /dev/sda
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
Command (m for help): x
Expert command (m for help): f
Done.
Expert command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
WARNING: Re-reading the partition table failed with error 16: Device or
resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

After, running '**f**' command, don't forget to run '**w**' command to save and exit from fdisk command mode. Once it fixed partition table order, you will no longer get error messages.

10. How to Disable Boot Flag (*) of a Partition

By default, fdisk command shows the boot flag (i.e. '*' symbol on each partition. If you want to enable or disable boot flag on a specific partition, do the following steps.

```
[root@tecmint ~]# fdisk /dev/sda
```

Press '**p**' command to view the current partition table, you see there is a boot flag (asterisk (*)) symbol in orange color) on **/dev/sda1** disk as shown below.

```
[root@tecmint ~]# fdisk /dev/sda
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
Command (m for help): p
```

```

Disk /dev/sda: 637.8 GB, 637802643456 bytes
255 heads, 63 sectors/track, 77541 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot      Start         End      Blocks   Id  System
/dev/sda1    *           1          13     104391   83  Linux
/dev/sda2            14         2624    20972857+   83  Linux
/dev/sda3            2625        4582    15727635   83  Linux
/dev/sda4            4583        77541    586043167+    5  Extended
/dev/sda5            4583        5887    10482381   83  Linux
/dev/sda6            5888        7192    10482381   83  Linux
/dev/sda7            7193        7845    5245191    83  Linux
/dev/sda8            7846        8367    4192933+   82  Linux swap / Solaris
/dev/sda9            8368        77541    555640123+  8e  Linux LVM

```

Next enter command ‘**a**‘ to disable boot flag, then enter partition number ‘**1**‘ as (i.e. **/dev/sda1**) in my case. This will disable boot flag on the partition **/dev/sda1**. This will remove the asterisk (*) flag.

```

Command (m for help): a
Partition number (1-9): 1
Command (m for help): p
Disk /dev/sda: 637.8 GB, 637802643456 bytes
255 heads, 63 sectors/track, 77541 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot      Start         End      Blocks   Id  System
/dev/sda1            1          13     104391   83  Linux
/dev/sda2            14         2624    20972857+   83  Linux
/dev/sda3            2625        4582    15727635   83  Linux
/dev/sda4            4583        77541    586043167+    5  Extended
/dev/sda5            4583        5887    10482381   83  Linux
/dev/sda6            5888        7192    10482381   83  Linux
/dev/sda7            7193        7845    5245191    83  Linux
/dev/sda8            7846        8367    4192933+   82  Linux swap / Solaris
/dev/sda9            8368        77541    555640123+  8e  Linux LVM
Command (m for help):

```

I’ve tried my best to include almost all basic commands of fdisk commands, but still fdisk contains a variety of other expert commands you can use them by entering ‘**x**‘. For more detailed information, check out ‘**man fdisk**‘ command from the terminal. If I’ve missed any important command, please do share with me via comment section.

10 Useful du (Disk Usage) Commands to Find Disk Usage of Files and Directories

by [Ravi Saive](#) | Published: January 21, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

The Linux “du” (**Disk Usage**) is a standard **Unix/Linux** command, used to check the information of disk usage of files and directories on a machine. The **du** command has many parameter options that can be used to get the results in many formats. The **du** command also displays the files and directory sizes in a recursively manner.



Check Disk Usage of Files and Folders In Linux

This article explains **10 useful “du” commands** with their examples, that might helps you to find out the sizes of files and directories in Linux. The information provided in this article are taken from the man pages of **du** command.

Read Also:

1. [12 “df” Command to Check Linux System Disk Space](#)

1. To find out the disk usage summary of a **/home/tecmint** directory tree and each of its sub directories. Enter the command as:

```
[root@tecmint]# du /home/tecmint
40      /home/tecmint/downloads
4       /home/tecmint/.mozilla/plugins
4       /home/tecmint/.mozilla/extensions
12      /home/tecmint/.mozilla
12      /home/tecmint/.ssh
689112  /home/tecmint/Ubuntu-12.10
```

```
689360 /home/tecmint
```

The output of the above command displays the number of disk blocks in the **/home/tecmint** directory along with its sub-directories.

2. Using “**-h**” option with “**du**” command provides results in “**Human Readable Format**”. Means you can see sizes in **Bytes**, **Kilobytes**, **Megabytes**, **Gigabytes** etc.

```
[root@tecmint]# du -h /home/tecmint
40K      /home/tecmint/downloads
4.0K     /home/tecmint/.mozilla/plugins
4.0K     /home/tecmint/.mozilla/extensions
12K      /home/tecmint/.mozilla
12K      /home/tecmint/.ssh
673M    /home/tecmint/Ubuntu-12.10
674M    /home/tecmint
```

3. To get the summary of a grand total disk usage size of an directory use the option “**-s**” as follows.

```
[root@tecmint]# du -sh /home/tecmint
674M    /home/tecmint
```

4. Using “**-a**” flag with “**du**” command displays the disk usage of all the files and directories.

```
[root@tecmint]# du -a /home/tecmint
4      /home/tecmint/.bash_logout
12     /home/tecmint/downloads/uploadprogress-1.0.3.1.tgz
24     /home/tecmint/downloads/Phpfiles-org.tar.bz2
40     /home/tecmint/downloads
12     /home/tecmint/uploadprogress-1.0.3.1.tgz
4      /home/tecmint/.mozilla/plugins
4      /home/tecmint/.mozilla/extensions
12     /home/tecmint/.mozilla
4      /home/tecmint/.bashrc
689108 /home/tecmint/Ubuntu-12.10/ubuntu-12.10-server-i386.iso
689112 /home/tecmint/Ubuntu-12.10
689360 /home/tecmint
```

5. Using “**-a**” flag along with “**-h**” displays disk usage of all files and folders in human readable format. The below output is more easy to understand as it shows the files in **Kilobytes**, **Megabytes** etc.

```
[root@tecmint]# du -ah /home/tecmint
4.0K    /home/tecmint/.bash_logout
12K    /home/tecmint/downloads/uploadprogress-1.0.3.1.tgz
24K    /home/tecmint/downloads/Phpfiles-org.tar.bz2
40K    /home/tecmint/downloads
12K    /home/tecmint/uploadprogress-1.0.3.1.tgz
4.0K    /home/tecmint/.mozilla/plugins
4.0K    /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
```

```
4.0K /home/tecmint/.bashrc
673M /home/tecmint/Ubuntu-12.10/ubuntu-12.10-server-i386.iso
673M /home/tecmint/Ubuntu-12.10
674M /home/tecmint
```

6. Find out the disk usage of a directory tree with its subtress in **Kilobyte** blocks. Use the “**-k**” (displays size in **1024** bytes units).

```
[root@tecmint]# du -k /home/tecmint
40      /home/tecmint/downloads
4       /home/tecmint/.mozilla/plugins
4       /home/tecmint/.mozilla/extensions
12      /home/tecmint/.mozilla
12      /home/tecmint/.ssh
689112  /home/tecmint/Ubuntu-12.10
689360  /home/tecmint
```

7. To get the summary of disk usage of directory tree along with its subtrees in **Megabytes (MB)** only. Use the option “**-mh**” as follows. The “**-m**” flag counts the blocks in **MB** units and “**-h**” stands for human readable format.

```
[root@tecmint]# du -mh /home/tecmint
40K     /home/tecmint/downloads
4.0K    /home/tecmint/.mozilla/plugins
4.0K    /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
12K    /home/tecmint/.ssh
673M   /home/tecmint/Ubuntu-12.10
674M   /home/tecmint
```

8. The “**-c**” flag provides a grand total usage disk space at the last line. If your directory taken **674MB** space, then the last two line of the output would be.

```
[root@tecmint]# du -ch /home/tecmint
40K     /home/tecmint/downloads
4.0K    /home/tecmint/.mozilla/plugins
4.0K    /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
12K    /home/tecmint/.ssh
673M   /home/tecmint/Ubuntu-12.10
674M   /home/tecmint
674M   total
```

9. The below command calculates and displays the disk usage of all files and directories, but excludes the files that matches given pattern. The below command excludes the “**.txt**” files while calculating the total size of directory. So, this way you can exclude any file formats by using flag “**--exclude**”. See the output there is no **txt** files entry.

```
[root@tecmint]# du -ah --exclude="*.txt" /home/tecmint
4.0K   /home/tecmint/.bash_logout
12K   /home/tecmint/downloads/uploadprogress-1.0.3.1.tgz
24K   /home/tecmint/downloads/Phpfiles-org.tar.bz2
```

```

40K    /home/tecmint/downloads
12K    /home/tecmint/uploadprogress-1.0.3.1.tgz
4.0K   /home/tecmint/.bash_history
4.0K   /home/tecmint/.bash_profile
4.0K   /home/tecmint/.mozilla/plugins
4.0K   /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
4.0K   /home/tecmint/.bashrc
24K    /home/tecmint/Phpfiles-org.tar.bz2
4.0K   /home/tecmint/geoipupdate.sh
4.0K   /home/tecmint/.zshrc
120K   /home/tecmint/goaccess-0.4.2.tar.gz.1
673M   /home/tecmint/Ubuntu-12.10/ubuntu-12.10-server-i386.iso
673M   /home/tecmint/Ubuntu-12.10
674M   /home/tecmint

```

10. Display the disk usage based on modification of time, use the flag “**–time**” as shown below.

```

[root@tecmint]# du -ha --time /home/tecmint
4.0K   2012-10-12 22:32      /home/tecmint/.bash_logout
12K    2013-01-19 18:48      /home/tecmint/downloads/uploadprogress-
1.0.3.1.tgz
24K    2013-01-19 18:48      /home/tecmint/downloads/Phpfiles-org.tar.bz2
40K    2013-01-19 18:48      /home/tecmint/downloads
12K    2013-01-19 18:32      /home/tecmint/uploadprogress-1.0.3.1.tgz
4.0K   2012-10-13 00:11      /home/tecmint/.bash_history
4.0K   2012-10-12 22:32      /home/tecmint/.bash_profile
0      2013-01-19 18:32      /home/tecmint/xyz.txt
0      2013-01-19 18:32      /home/tecmint/abc.txt
4.0K   2012-10-12 22:32      /home/tecmint/.mozilla/plugins
4.0K   2012-10-12 22:32      /home/tecmint/.mozilla/extensions
12K    2012-10-12 22:32      /home/tecmint/.mozilla
4.0K   2012-10-12 22:32      /home/tecmint/.bashrc
24K    2013-01-19 18:32      /home/tecmint/Phpfiles-org.tar.bz2
4.0K   2013-01-19 18:32      /home/tecmint/geoipupdate.sh
4.0K   2012-10-12 22:32      /home/tecmint/.zshrc
120K   2013-01-19 18:32      /home/tecmint/goaccess-0.4.2.tar.gz.1
673M   2013-01-19 18:51      /home/tecmint/Ubuntu-12.10/ubuntu-12.10-
server-i386.iso
673M   2013-01-19 18:51      /home/tecmint/Ubuntu-12.10
674M   2013-01-19 18:52      /home/tecmint

```

How to Set Static IP Address and Configure Network in Linux

by [Marin Todorov](#) | Published: April 13, 2016 | Last Updated: April 14, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

If you are a Linux system administrator, time will come when you will need to configure networking on your system. Unlike desktop machines where you can use dynamic IP addresses, on a server infrastructure, you will need to setup a static IP address (at least in most cases).

Read Also: [How to Set or Change System Hostname in Linux](#)

IP address: 192.168.0.100 **Netmask:** 255.255.255.0 **Hostname:** node01.tecmint.com **Domain name:** tecmint.com **Gateway:** 192.168.0.1 **DNS Server 1:** 8.8.8.8 **DNS Server 2:** 4.4.4.4

Configure Static IP Address in RHEL/CentOS/Fedora:

To configure static IP address in **RHEL / CentOS / Fedora**, you will need to edit:

```
/etc/sysconfig/network  
/etc/sysconfig/network-scripts/ifcfg-eth0
```

Where in the above "ifcfg-eth0" answers to your network interface eth0. If your interface is named "eth1" then the file that you will need to edit is "ifcfg-eth1".

Let's start with the first file:

```
# vi /etc/sysconfig/network
```

Open that file and set:

```
NETWORKING=yes  
HOSTNAME=node01.tecmint.com  
GATEWAY=192.168.0.1  
NETWORKING_IPV6=no  
IPV6INIT=no
```

Next open:

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

Note: Make sure to open the file corresponding to your network interface. You can find your network interface name with [ifconfig -a command](#).

In that file make the following changes:

```
DEVICE="eth0"
BOOTPROTO="static"
DNS1="8.8.8.8"
DNS2="4.4.4.4"
GATEWAY="192.168.0.1"
HOSTNAME="node01.tecmint.com"
HWADDR="00:19:99:A4:46:AB"
IPADDR="192.68.0.100"
NETMASK="255.255.255.0"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="8105c095-799b-4f5a-a445-c6d7c3681f07"
```

You will only need to edit the settings for:

1. DNS1 and DNS2
2. GATEWAY
3. HOSTNAME
4. NETMASK
5. IPADDR

Other settings should have already been predefined.

Next edit `resolve.conf` file by opening it with a text editor such as [nano or vi](#):

```
# vi /etc/resolv.conf

nameserver 8.8.8.8 # Replace with your nameserver ip
nameserver 4.4.4.4 # Replace with your nameserver ip
```

Once you have made your changes restart the networking with:

```
# /etc/init.d/network restart [On SysVinit]
# systemctl restart network [On SystemD]
```

Set Static IP Address in Debian / Ubuntu

To setup static IP address in **Debian/ Ubuntu**, open the following file:

```
# nano /etc/network/interfaces
```

You may see a line looking like this:

```
auto eth0
iface eth0 inet dhcp
```

Change it so it looks like this:

```
auto eth0
iface eth0 inet static
address 192.168.0.100
netmask 255.255.255.0
gateway 192.168.0.1
dns-nameservers 4.4.4.4
dns-nameservers 8.8.8.8
```

Save the file and then edit /etc/resolv.conf like this:

```
# nano /etc/resolv.conf
nameserver 8.8.8.8 # Replace with your nameserver ip
nameserver 4.4.4.4 # Replace with your nameserver ip
```

Restart the networking on your system with:

```
# /etc/init.d/network restart [On SysVinit]
# systemctl restart network [On SystemD]
```

Your static IP address has been configured.

Conclusion:

You now know how to configure a static IP address on a Linux distro. If you have any questions or comments, please do not hesitate to submit them in the comment section below.

How to Auto Execute Commands/Scripts During Reboot or Startup

by [Gabriel Cánepa](#) | Published: February 13, 2017 | Last Updated: February 13, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

I am always fascinated by the things going on behind the scenes when I [boot a Linux system and log on](#). By pressing the power button on a bare metal or starting a virtual machine, you put in motion a series of events that lead to a fully-functional system – sometimes in less than a minute. The same is true when you log off and / or shutdown the system.

What makes this more interesting and fun is the fact that you can have the operating system execute certain actions when it boots and when you logon or logout.

In this distro-agnostic article we will discuss the traditional methods for accomplishing these goals in Linux.

Note: We will assume the use of **Bash** as main shell for logon and logout events. If you happen to use a different one, some of these methods may or may not work. If in doubt, refer to the documentation of your shell.

Executing Linux Scripts During Reboot or Startup

There are two traditional methods to execute a command or run scripts during startup:

Method #1 – Use a cron Job

Besides the usual format (minute / hour / day of month / month / day of week) that is widely used to indicate a schedule, [cron scheduler](#) also allows the use of @reboot. This directive, followed by the absolute path to the script, will cause it to run when the machine boots.

However, there are two caveats to this approach:

1. **a)** the cron daemon must be running (which is the case under normal circumstances), and
2. **b)** the script or the crontab file must include the environment variables (if any) that will be needed (refer to this StackOverflow thread for more details).

Method #2 – Use /etc/rc.d/rc.local

This method is valid even for systemd-based distributions. In order for this method to work, you must grant execute permissions to `/etc/rc.d/rc.local` as follows:

```
# chmod +x /etc/rc.d/rc.local
```

and add your script at the bottom of the file.

The following image shows how to run two sample scripts (`/home/gacanepa/script1.sh` and `/home/gacanepa/script2.sh`) using a **cron** job and **rc.local**, respectively, and their respective results.

`script1.sh:`

```
#!/bin/bash
DATE=$(date '+%F %H:%M:%S')
DIR=/home/gacanepa
echo "Current date and time: $DATE" > $DIR/file1.txt
script2.sh:
#!/bin/bash
SITE="Tecmint.com"
DIR=/home/gacanepa
echo "$SITE rocks... add us to your bookmarks." > $DIR/file2.txt
```

```
[gacanepa@server ~]# crontab -l
@reboot /home/gacanepa/script1.sh
[gacanepa@server ~]# tail -n 5 /etc/rc.d/rc.local
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
# that this script will be executed during boot.

touch /var/lock/subsys/local
/home/gacanepa/script2.sh
[gacanepa@server ~]# _
```



```
[gacanepa@server ~]# ls
file1.txt file2.txt script1.sh script2.sh
[gacanepa@server ~]# cat file1.txt file2.txt
Current date and time: 2017-02-11 17:28:24
Tecmint.com rocks... add us to your bookmarks.
[gacanepa@server ~]# _
```

Run Linux Scripts at Startup

Keep in mind that both scripts must be granted execute permissions previously:

```
$ chmod +x /home/gacanepa/script1.sh
$ chmod +x /home/gacanepa/script2.sh
```

Executing Linux Scripts at Logon and Logout

To execute a script at logon or logout, use `~.bash_profile` and `~.bash_logout`, respectively. Most likely, you will need to create the latter file manually. Just drop a line invoking your script at the bottom of each file in the same fashion as before and you are ready to go.

Summary

In this article we have explained how to run script at reboot, logon, and logout. If you can think of other methods we could have included here, feel free to use the comment form below to point them out. We look forward to hearing from you!

How to Find Number of Files in a Directory and Subdirectories

by [Aaron Kili](#) | Published: January 17, 2017 | Last Updated: January 17, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this guide, we will cover how to display the total number of files in the current working directory or any other directory and its subdirectories on a Linux system.

We will use the [find command](#) which is used to search for files in a directory hierarchy together with [wc command](#) which prints newline, word, and byte counts for each file, alternatively data read from standard input.

Following are the options that we can use with [find command](#) as follows:

1. `-type f` – specifies the file type to search for, in the case above, the `f` means find all regular files.
2. `-print` – an action to print the absolute path of a file.
3. `-l` – this option prints the total number of newlines, which is equals to the total number of absolute file paths output by [find command](#).

The general syntax of find command.

```
# find . -type f -print | wc -l
$ sudo find . -type f -print | wc -l
```

Important: Use [sudo command](#) to read all files in the specified directory including those in the subdirectories with superuser privileges, in order to avoid “**Permission denied**” errors as in the screen shot below:

```
aaronkilik@tecmint ~ $ find . -type f -print | wc -l
find: './.config/etcher': Permission denied
find: './.pki': Permission denied
find: './.cache/dconf': Permission denied
58043
aaronkilik@tecmint ~ $ sudo find . -type f -print | wc -l
58061
aaronkilik@tecmint ~ $
```

Find Number of Files in Linux

You can see that in the first command above, not all files in the current working directory are read by **find** command.

The following are extra examples to show total number of regular files in `/var/log` and `/etc` directories respectively:

```
$ sudo find /var/log/ -type f -print | wc -l  
$ sudo find /etc/ -type f -print | wc -l
```

6 WC Command Examples to Count Number of Lines, Words, Characters in Linux

by [Ravi Saive](#) | Published: February 25, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

The **wc** (word count) command in Unix/Linux operating systems is used to find out number of **newline count, word count, byte and characters** count in a files specified by the file arguments. The syntax of **wc** command as shown below.

```
# wc [options] filenames
```

The following are the options and usage provided by the command.

```
wc -l : Prints the number of lines in a file.  
wc -w : prints the number of words in a file.  
wc -c : Displays the count of bytes in a file.  
wc -m : prints the count of characters from a file.  
wc -L : prints only the length of the longest line in a file.
```

So, let's see how we can use the '**wc**' command with their few available arguments and examples in this article. We have used the '**tecmint.txt**' file for testing the commands. Let's find out the output of the file using [cat command](#) as shown below.

```
[root@tecmint ~]# cat tecmint.txt  
Red Hat  
CentOS  
Fedora  
Debian  
Scientific Linux  
OpenSuse  
Ubuntu  
Xubuntu  
Linux Mint  
Pearl Linux  
Slackware  
Mandriva
```

1. A Basic Example of WC Command

The '**wc**' command without passing any parameter will display a basic result of "**tecmint.txt**" file. The three numbers shown below are **12 (number of lines)**, **16 (number of words)** and **112 (number of bytes)** of the file.

```
[root@tecmint ~]# wc tecmint.txt  
12 16 112 tecmint.txt
```

2. Count Number of Lines

To count number of newlines in a file use the option ‘-l‘, which prints the number of lines from a given file. Say, the following command will display the count of newlines in a file. In the output the first field assigned as count and second field is the name of file.

```
[root@tecmint ~]# wc -l tecmint.txt  
12 tecmint.txt
```

3. Display Number of Words

Using ‘-w‘ argument with ‘wc‘ command prints the number of words in a file. Type the following command to count the words in a file.

```
[root@tecmint ~]# wc -w tecmint.txt  
16 tecmint.txt
```

4. Count Number of Bytes and Characters

When using options ‘-c‘ and ‘-m‘ with ‘wc‘ command will print the total **number of bytes** and **characters** respectively in a file.

```
[root@tecmint ~]# wc -c tecmint.txt  
112 tecmint.txt  
[root@tecmint ~]# wc -m tecmint.txt  
112 tecmint.txt
```

5. Display Length of Longest Line

The ‘wc‘ command allow an argument ‘-L‘, it can be used to print out the length of longest (**number of characters**) line in a file. So, we have the longest character line (‘**Scientific Linux**’) in a file.

```
[root@tecmint ~]# wc -L tecmint.txt  
16 tecmint.txt
```

6. Check More WC Options

For more information and help on the **wc** command, simple run the ‘**wc --help**‘ or ‘**man wc**‘ from the command line.

```
[root@tecmint ~]# wc --help  
Usage: wc [OPTION]... [FILE]...  
or:  wc [OPTION]... --files0-from=F  
Print newline, word, and byte counts for each FILE, and a total line if  
more than one FILE is specified. With no FILE, or when FILE is -,  
read standard input.  
-c, --bytes          print the byte counts  
-m, --chars         print the character counts
```

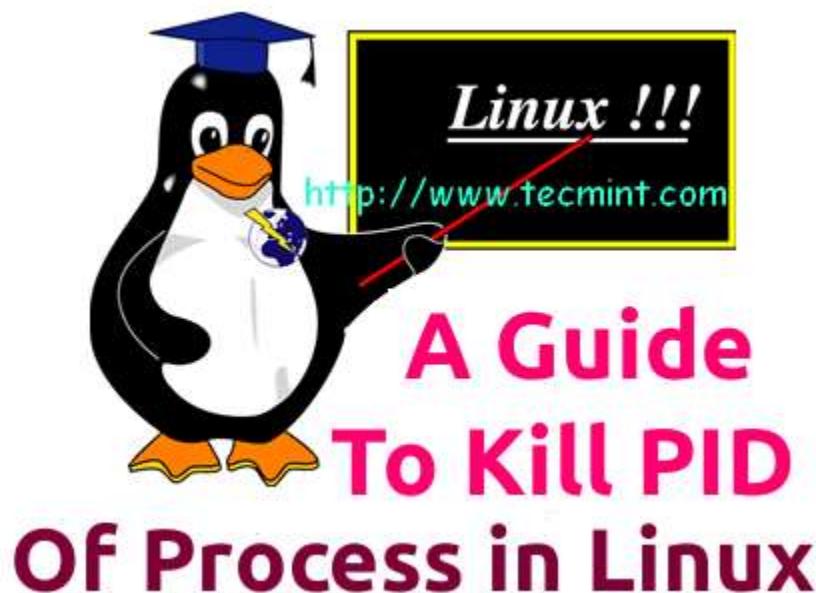
```
-l, --lines          print the newline counts
-L, --max-line-length print the length of the longest line
-w, --words          print the word counts
--help               display this help and exit
--version            output version information and exit
Report wc bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'wc invocation'
```

A Guide to Kill, Pkill and Killall Commands to Terminate a Process in Linux

by [Editor](#) | Published: September 19, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Linux Operating System comes with Kill command to terminate a process. The command makes it possible to continue running the server without the need of reboot after a major change/update. Here comes the great power of Linux and this is one of the reasons, why Linux is running on **90%** of servers, on the planet.



Kill, Pkill and Killall Commands Examples

Kill command send a signal, a specified signal to be more perfect to a process. The kill command can be executed in a number of ways, directly or from a shell script.

Using **kill** command from **/usr/bin** provide you some extra feature to kill a process by process name using **pkill**. The common syntax for kill command is:

```
# kill [signal or option] PID(s)
```

For a kill command a Signal Name could be:

Signal Name	Signal Value	Behaviour
-------------	--------------	-----------

SIGHUP	1	Hangup
SIGKILL	9	Kill Signal
SIGTERM	15	Terminate

Clearly from the behaviour above **SIGTERM** is the default and safest way to kill a process. **SIGHUP** is less secure way of killing a process as **SIGTERM**. **SIGKILL** is the most unsafe way among the above three, to kill a process which terminates a process without saving.

In order to kill a process, we need to know the **Process ID** of a process. A **Process** is an instance of a program. Every-time a program starts, automatically an unique **PID** is generated for that process. Every Process in **Linux**, have a **pid**. The first process that starts when Linux System is booted is – **init process**, hence it is assigned a value of ‘1‘ in most of the cases.

Init is the master process and can not be killed this way, which insures that the master process don’t gets killed accidentally. **Init** decides and allows itself to be killed, where kill is merely a request for a shutdown.

To know all the processes and correspondingly their assigned **pid**, run.

```
# ps -A
```

Sample Output

PID	TTY	TIME	CMD
1	?	00:00:01	init
2	?	00:00:00	kthreadd
3	?	00:00:00	migration/0
4	?	00:00:00	ksoftirqd/0
5	?	00:00:00	migration/0
6	?	00:00:00	watchdog/0
7	?	00:00:01	events/0
8	?	00:00:00	cgroup
9	?	00:00:00	khelper
10	?	00:00:00	netns
11	?	00:00:00	async/mgr
12	?	00:00:00	pm
13	?	00:00:00	sync_supers
14	?	00:00:00	bdi-default
15	?	00:00:00	kintegrityd/0
16	?	00:00:00	kbroadcastd/0
17	?	00:00:00	kacpid
18	?	00:00:00	kacpi_notify
19	?	00:00:00	kacpi_hotplug
20	?	00:00:00	ata/0
21	?	00:00:00	ata_aux
22	?	00:00:00	ksuspend_usbd

How about Customising the above output using syntax as ‘**pidof process**‘.

```
# pidof mysqld
```

Sample Output

Another way to achieve the above goal is to follow the below syntax.

```
# ps aux | grep mysqld
```

Sample Output

```
root      1582  0.0  0.0  5116  1408 ?          S      09:49   0:00 /bin/sh
/usr/bin/mysqld_safe --datadir=/var/lib/mysql --
socket=/var/lib/mysql/mysql.sock --pid-file=/var/run/mysqld/mysqld.pid --
basedir=/usr --user=mysql
mysql     1684  0.1  0.5 136884 21844 ?          Sl      09:49   1:09
/usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql --
log-error=/var/log/mysqld.log --pid-file=/var/run/mysqld/mysqld.pid --
socket=/var/lib/mysql/mysql.sock
root      20844  0.0  0.0   4356    740 pts/0      S+     21:39   0:00 grep mysqld
```

Before we step ahead and execute a kill command, some important points to be noted:

1. A user can kill all his process.
2. A user can not kill another user's process.
3. A user can not kill processes System is using.
4. A root user can kill System-level-process and the process of any user.

Another way to perform the same function is to execute ‘**pgrep**’ command.

```
# pgrep mysq
```

Sample Output

3139

To kill the above process **PID**, use the kill command as shown.

```
kill -9 3139
```

The above command will kill the process having **pid=3139**, where **PID** is a **Numerical Value** of process.

Another way to perform the same function, can be rewritten as.

```
# kill -SIGTERM 3139
```

Similarly ‘**kill -9 PID**’ is similar to ‘**kill -SIGKILL PID**’ and vice-versa.

How about killing a process using process name

You must be aware of process name, before killing and entering a wrong process name may screw you.

```
# pkill mysqld
```

Kill more than one process at a time.

```
# kill PID1 PID2 PID3  
or  
# kill -9 PID1 PID2 PID3  
or  
# kill -SIGKILL PID1 PID2 PID3
```

What if a process have too many instances and a number of child processes, we have a command '**killall**'. This is the only command of this family, which takes process name as argument instead of process number.

Syntax:

```
# killall [signal or option] Process Name
```

To kill all **mysql instances** along with child processes, use the command as follow.

```
# killall mysqld
```

You can always verify the status of the process if it is running or not, using any of the below command.

```
# service mysql status  
# pgrep mysql  
# ps -aux | grep mysql
```

14 Useful Examples of Linux ‘sort’ Command – Part 1

by [Editor](#) | Published: April 15, 2015 | Last Updated: April 18, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Sort is a Linux program used for printing lines of input text files and concatenation of all files in sorted order. Sort command takes blank space as field separator and entire Input file as sort key. It is important to notice that sort command don't actually sort the files but only print the sorted output, until your redirect the output.

This article aims at deep insight of Linux ‘**sort**’ command with 14 useful practical examples that will show you how to use sort command in Linux.

1. First we will be creating a text file (**tecmint.txt**) to execute ‘**sort**’ command examples. Our working directory is ‘**/home/\$USER/Desktop/tecmint**’.

The option ‘-e’ in the below command enables interpretation of backslash and /n tells **echo** to write each string to a new line.

```
$ echo -e "computer\nmouse\nLAPTOP\nadata\nRedHat\nlaptop\ndebian\nlaptop" > tecmint.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ e
```

2. Before we start with ‘**sort**‘ lets have a look at the contents of the file and the way it look.

```
$ cat tecmint.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop-tecmint$ cat
```

3. Now sort the content of the file using following command.

```
$ sort tecmint.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$
```

Note: The above command don't actually sort the contents of text file but only show the sorted output on terminal.

4. Sort the contents of the file ‘**tecmint.txt**‘ and write it to a file called (**sorted.txt**) and verify the content by using [cat command](#).

```
$ sort tecmint.txt > sorted.txt  
$ cat sorted.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

5. Now sort the contents of text file ‘**tecmint.txt**’ in reverse order by using ‘**-r**’ switch and redirect output to a file ‘**reversesorted.txt**’. Also check the content listing of the newly created file.

```
$ sort -r tecmint.txt > reversesorted.txt  
$ cat reversesorted.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

6. We are going to create a new file (**lsl.txt**) at the same location for detailed examples and populate it using the output of ‘**ls -l**‘ for your home directory.

```
$ ls -l /home/$USER > /home/$USER/Desktop/tecmint/lsl.txt  
$ cat lsl.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

Now will see examples to sort the contents on the basis of other field and not the default initial characters.

7. Sort the contents of file ‘**lsl.txt**‘ on the basis of **2nd column** (which represents number of symbolic links).

```
$ sort -nk2 lsl.txt
```

Note: The ‘**-n**‘ option in the above example sort the contents numerically. Option ‘**-n**‘ must be used when we wanted to sort a file on the basis of a column which contains numerical values.

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$
```

8. Sort the contents of file ‘**lsl.txt**‘ on the basis of **9th column** (which is the name of the files and folders and is non-numeric).

```
$ sort -k9 lsl.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ sort -k
```

- 9.** It is not always essential to run sort command on a file. We can pipeline it directly on the terminal with actual command.

```
$ ls -l /home/$USER | sort -nk5
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

- 10.** Sort and remove duplicates from the text file **tecmint.txt**. Check if the duplicate has been removed or not.

```
$ cat tecmint.txt  
$ sort -u tecmint.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$
```

Rules so far (what we have observed):

1. Lines starting with numbers are preferred in the list and lies at the top until otherwise specified (**-r**).
2. Lines starting with lowercase letters are preferred in the list and lies at the top until otherwise specified (**-r**).
3. Contents are listed on the basis of occurrence of alphabets in dictionary until otherwise specified (**-r**).
4. Sort command by default treat each line as string and then sort it depending upon dictionary occurrence of alphabets (Numeric preferred; see rule – 1) until otherwise specified.

11. Create a third file ‘**Isla.txt**‘ at the current location and populate it with the output of ‘**ls -lA**‘ command.

```
$ ls -lA /home/$USER > /home/$USER/Desktop/tecmint/Isla.txt  
$ cat Isla.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$
```

Those having understanding of ‘ls‘ command knows that ‘ls -lA’=‘ls -l‘ + **Hidden** files. So most of the contents on these two files would be same.

12. Sort the contents of two files on standard output in one go.

```
$ sort lsl.txt lsla.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop-tecmint$ s
```

Notice the repetition of files and folders.

13. Now we can see how to sort, merge and remove duplicates from these two files.

```
$ sort -u lsl.txt lsla.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$
```

Notice that duplicates has been omitted from the output. Also, you can write the output to a new file by redirecting the output to a file.

14. We may also sort the contents of a file or the output based upon more than one column. Sort the output of ‘ls -l‘ command on the basis of field 2,5 (Numeric) and 9 (Non-Numeric).

```
$ ls -l /home/$USER | sort -t "," -nk2,5 -k9
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

That's all for now. In the next article we will cover a few more examples of ‘**sort**’ command in detail for you. Till then stay tuned and connected to Tecmint. Keep sharing. Keep commenting. Like and share us and help us get spread.

7 Interesting Linux ‘sort’ Command Examples – Part 2

by [Editor](#) | Published: April 18, 2015 | Last Updated: April 18, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In our last article we have covered various examples on **sort** command, if you've missed, you can go through it using below link. In continuation to the last post this post aims at covering remaining of sort command so that both the article together serves as complete guide to Linux ‘**sort**’ command.

1. [14 ‘sort’ Command Examples in Linux](#)

Before we continue further, create a text file ‘**month.txt**’ and populate it with the data as given below.

```
$ echo -e "mar\ndec\noct\nsep\nfeb\naug" > month.txt
$ cat month.txt
```

15. Sort the file ‘**month.txt**’ on the basis of month order by using switch ‘**M**’ (**-month-sort**).

```
$ sort -M month.txt
```

Important: Note that ‘**sort**’ command needs at least 3 characters to consider month name.

16. Sort the data that is in human readable format say 1K, 2M, 3G, 2T, where K,M,G,T represents Kilo, Mega, Giga, Tera.

```
$ ls -l /home/$USER | sort -h -k5
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$
```

- 17.** In the last article we have created a file ‘sorted.txt‘ in example **number 4** and another text file ‘lsl.txt‘ in example **number 6**. We know ‘sorted.txt‘ is already sorted while ‘lsl.txt‘ is not. Lets check both the files are sorted or not using sort command.

```
$ sort -c sorted.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

If it returns **0**, means that the file is sorted and there is no conflict.

```
$ sort -c lsl.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

Reports Disorder. Conflict..

18. If the delimiter (separator) between words are space, sort command automatically interpret anything after horizontal space as new word. What if the delimiter is not space?

Consider a text file, the contents of which are separated by anything other than space such as ‘|’ or ‘\’ or ‘+’ or ‘.’ or....

Create a text file where contents are separated by +. Use ‘cat‘ to check the contents of file.

```
$ echo -e  
"21+linux+server+production\n11+debian+RedHat+CentOS\n131+Apache+Mysql+PHP\n7  
+Shell Scripting+python+perl\n111+postfix+exim+sendmail" > delimiter.txt  
$ cat delimiter.txt
```

```
File Edit View Search Terminal Help
```

```
avi@tecmint:~/Desktop/tecmint$ echo -e "21+linux+server+production\n11+devel+CentOS\n131+Apache+Mysql+PHP\n7+Shell Scripting+python+perl\n111+postfix+sendmail" > delimiter.txt
```

Now sort this file on the basis of **1st** field which is numerical.

```
$ sort -t '+' -nk1 delimiter.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

And second on the basis of **4th** field which is non numeric.

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

If the delimiter is **Tab** you may use `$'\t'` in place of ‘+’, as shown in the above example.

19. Sort the contents of ‘`ls -l`’ command for your home directory on the basis of **5th column** which represents the ‘**amount of data**‘ in Random order.

```
$ ls -l /home/avi/ | sort -k5 -R
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$
```

Every time you run the above piece of script you are likely to get a different result since the result is generated randomly.

As clear from the **Rule number – 2** from the last article, **sort** command prefer line starting with lowercase characters over uppercase characters. Also check **example 3** in last article, where string ‘**laptop**’ appears before string ‘**LAPTOP**’.

20. How to override the default sorting preference? before we are able to override the default sorting preference we need to export the environment variable **LC_ALL to c**. To do this run the below code on your Command Line Prompt.

```
$ export LC_ALL=C
```

And then sort the text file ‘**tecmint.txt**‘ overriding the default sort preference.

```
$ sort tecmint.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

Override Sorting Preferences

Don't forget to compare the output with the one you achieved in **example 3** and also you can use option '**-f**' aka '**-ignore-case**' to get much organized output.

```
$ sort -f tecmint.txt
```

```
File Edit View Search Terminal Help  
avi@tecmint:~/Desktop/tecmint$ █
```

21. How about running ‘sort‘ on two input files and join them in one go!

Lets create two text file namely ‘**file1.txt**‘ and ‘**file2.txt**‘ and populate it with some data. Here we are populating ‘**file1.txt**‘ with numbers as below. Also used ‘**cat**‘ command to check the content of file.

```
$ echo -e "5 Reliable\n2 Fast\n3 Secure\n1 open-source\n4 customizable" >  
file1.txt  
$ cat file1.txt
```

```
File Edit View Search Terminal Help
```

```
avi@tecmint:~$ echo -e "5 Reliable\n2 Fast\n3 Secure\n1 open-source\n4 c  
le" > file1.txt
```

And populate second file ‘**file2.txt**‘ with some data as.

```
$ echo -e "3 RedHat\n1 Debian\n5 Ubuntu\n2 Kali\n4 Fedora" > file2.txt  
$ cat file2.txt
```

File Edit View Search Terminal Help

```
avi@tecmint:~$ echo -e "3 RedHat\n1 Debian\n5 Ubuntu\n2 Kali\n4 Fedora"  
xt
```

Now sort and join the output of both the files.

```
$ join <(sort -n file1.txt) <(sort file2.txt)
```

File Edit View Search Terminal Help

avi@tecmint:~\$ █

That's all for now. Keep Connected. Keep to Tecmint. Please Provide us with your valuable feedback in the comments below. Like and share us and help us get spread

13 Linux Network Configuration and Troubleshooting Commands

by [Ravi Saive](#) | Published: September 24, 2012 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Computers are connected in a network to exchange information or resources each other. Two or more computer connected through network media called **computer network**. There are number of network devices or media are involved to form computer network. Computer loaded with **Linux Operating System** can also be a part of network whether it is small or large network by its **multitasking and multiuser** natures. Maintaining of system and network up and running is a task of **System / Network Administrator's** job. In this article we are going to review frequently used network configuration and troubleshoot commands in Linux.



Linux Network Configuration and Troubleshooting Commands

1. ifconfig

ifconfig (interface configurator) command is use to initialize an interface, assign **IP Address** to interface and **enable** or **disable** interface on demand. With this command you can view **IP Address** and **Hardware / MAC address** assign to interface and also **MTU (Maximum transmission unit)** size.

```
# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0C:29:28:FD:4C
inet addr:192.168.50.2 Bcast:192.168.50.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe28:fd4c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:6093 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4824 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

```
RX bytes:6125302 (5.8 MiB) TX bytes:536966 (524.3 KiB)
Interrupt:18 Base address:0x2000
lo      Link encap:Local Loopback
inet  addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:480 (480.0 b) TX bytes:480 (480.0 b)
```

ifconfig with interface (**eth0**) command only shows specific interface details like **IP Address**, **MAC Address** etc. with **-a** options will display all available interface details if it is disable also.

```
# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:0C:29:28:FD:4C
inet  addr:192.168.50.2 Bcast:192.168.50.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe28:fd4c/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:6119 errors:0 dropped:0 overruns:0 frame:0
TX packets:4841 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6127464 (5.8 MiB) TX bytes:539648 (527.0 KiB)
Interrupt:18 Base address:0x2000
```

Assigning IP Address and Gateway

Assigning an **IP Address** and **Gateway** to interface on the fly. The setting will be removed in case of system reboot.

```
# ifconfig eth0 192.168.50.5 netmask 255.255.255.0
```

Enable or Disable Specific Interface

To **enable** or **disable** specific Interface, we use example command as follows.

Enable eth0

```
# ifup eth0
```

Disable eth0

```
# ifdown eth0
```

Setting MTU Size

By default **MTU** size is **1500**. We can set required **MTU** size with below command. Replace **XXXX** with size.

```
# ifconfig eth0 mtu XXXX
```

Set Interface in Promiscuous mode

Network interface only received packets belongs to that particular **NIC**. If you put interface in **promiscuous** mode it will receive all the packets. This is very useful to capture packets and analyze later. For this you may require superuser access.

```
# ifconfig eth0 - promisc
```

2. PING Command

PING (Packet INternet Groper) command is the best way to test connectivity between **two nodes**. Whether it is **Local Area Network (LAN)** or **Wide Area Network (WAN)**. Ping use **ICMP (Internet Control Message Protocol)** to communicate to other devices. You can ping host name of **ip address** using below command.

```
# ping 4.2.2.2
PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.
64 bytes from 4.2.2.2: icmp_seq=1 ttl=44 time=203 ms
64 bytes from 4.2.2.2: icmp_seq=2 ttl=44 time=201 ms
64 bytes from 4.2.2.2: icmp_seq=3 ttl=44 time=201 ms
OR
# ping www.tecmint.com
PING tecmint.com (50.116.66.136) 56(84) bytes of data.
64 bytes from 50.116.66.136: icmp_seq=1 ttl=47 time=284 ms
64 bytes from 50.116.66.136: icmp_seq=2 ttl=47 time=287 ms
64 bytes from 50.116.66.136: icmp_seq=3 ttl=47 time=285 ms
```

In **Linux** ping command keep executing until you interrupt. Ping with **-c** option exit after **N** number of request (success or error respond).

```
# ping -c 5 www.tecmint.com
PING tecmint.com (50.116.66.136) 56(84) bytes of data.
64 bytes from 50.116.66.136: icmp_seq=1 ttl=47 time=285 ms
64 bytes from 50.116.66.136: icmp_seq=2 ttl=47 time=285 ms
64 bytes from 50.116.66.136: icmp_seq=3 ttl=47 time=285 ms
64 bytes from 50.116.66.136: icmp_seq=4 ttl=47 time=285 ms
64 bytes from 50.116.66.136: icmp_seq=5 ttl=47 time=285 ms
--- tecmint.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4295ms
rtt min/avg/max/mdev = 285.062/285.324/285.406/0.599 ms
```

3. TRACEROUTE Command

traceroute is a network troubleshooting utility which shows number of hops taken to reach destination also determine packets traveling path. Below we are tracing route to global **DNS server IP Address** and able to reach destination also shows path of that packet is traveling.

```
# traceroute 4.2.2.2
traceroute to 4.2.2.2 (4.2.2.2), 30 hops max, 60 byte packets
1  192.168.50.1 (192.168.50.1)  0.217 ms  0.624 ms  0.133 ms
2  227.18.106.27.mysipl.com (27.106.18.227)  2.343 ms  1.910 ms  1.799 ms
```

```

3 221-231-119-111.mysipl.com (111.119.231.221) 4.334 ms 4.001 ms 5.619 ms
4 10.0.0.5 (10.0.0.5) 5.386 ms 6.490 ms 6.224 ms
5 gio-0-0.dgw1.bom2.pacific.net.in (203.123.129.25) 7.798 ms 7.614 ms
7.378 ms
6 115.113.165.49.static-mumbai.vsnl.net.in (115.113.165.49) 10.852 ms
5.389 ms 4.322 ms
7 ix-0-100.tcore1.MLV-Mumbai.as6453.net (180.87.38.5) 5.836 ms 5.590 ms
5.503 ms
8 if-9-5.tcore1.WYN-Marseille.as6453.net (80.231.217.17) 216.909 ms
198.864 ms 201.737 ms
9 if-2-2.tcore2.WYN-Marseille.as6453.net (80.231.217.2) 203.305 ms 203.141
ms 202.888 ms
10 if-5-2.tcore1.WV6-Madrid.as6453.net (80.231.200.6) 200.552 ms 202.463
ms 202.222 ms
11 if-8-2.tcore2.SV8-Highbridge.as6453.net (80.231.91.26) 205.446 ms
215.885 ms 202.867 ms
12 if-2-2.tcore1.SV8-Highbridge.as6453.net (80.231.139.2) 202.675 ms
201.540 ms 203.972 ms
13 if-6-2.tcore1.NJY-Newark.as6453.net (80.231.138.18) 203.732 ms 203.496
ms 202.951 ms
14 if-2-2.tcore2.NJY-Newark.as6453.net (66.198.70.2) 203.858 ms 203.373 ms
203.208 ms
15 66.198.111.26 (66.198.111.26) 201.093 ms 63.243.128.25 (63.243.128.25)
206.597 ms 66.198.111.26 (66.198.111.26) 204.178 ms
16 ae9.edge1.NewYork.Level3.net (4.68.62.185) 205.960 ms 205.740 ms
205.487 ms
17 vlan51.ebr1.NewYork2.Level3.net (4.69.138.222) 203.867 ms
vlan52.ebr2.NewYork2.Level3.net (4.69.138.254) 202.850 ms
vlan51.ebr1.NewYork2.Level3.net (4.69.138.222) 202.351 ms
18 ae-6-6.ebr2.NewYork1.Level3.net (4.69.141.21) 201.771 ms 201.185 ms
201.120 ms
19 ae-81-81.csv3.NewYork1.Level3.net (4.69.134.74) 202.407 ms 201.479 ms
ae-92-92.csv4.NewYork1.Level3.net (4.69.148.46) 208.145 ms
20 ae-2-70.edge2.NewYork1.Level3.net (4.69.155.80) 200.572 ms ae-4-
90.edge2.NewYork1.Level3.net (4.69.155.208) 200.402 ms ae-1-
60.edge2.NewYork1.Level3.net (4.69.155.16) 203.573 ms
21 b.resolvers.Level3.net (4.2.2.2) 199.725 ms 199.190 ms 202.488 ms

```

4. NETSTAT Command

Netstat (Network Statistic) command display connection info, routing table information etc. To displays routing table information use option as **-r**.

```

# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask        Flags   MSS Window irtt
Iface
192.168.50.0    *           255.255.255.0    U        0 0          0 eth0
link-local      *           255.255.0.0     U        0 0          0 eth0
default         192.168.50.1  0.0.0.0      UG       0 0          0 eth0

```

For more examples of **Netstat Command**, please read our earlier article on [20 Netstat Command Examples in Linux](#).

5. DIG Command

Dig (domain information groper) query **DNS** related information like **A Record**, **CNAME**, **MX Record** etc. This command mainly use to troubleshoot **DNS** related query.

```
# dig www.tecmint.com; <>>> DiG 9.8.2rc1-RedHat-9.8.2-0.10.rc1.el6 <>>>
www.tecmint.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<
```

For more examples of **Dig Command**, please read the article on [10 Linux Dig Commands to Query DNS](#).

6. NSLOOKUP Command

nslookup command also use to find out **DNS** related query. The following examples shows **A Record (IP Address)** of **tecmint.com**.

```
# nslookup www.tecmint.com
Server:        4.2.2.2
Address:       4.2.2.2#53
Non-authoritative answer:
www.tecmint.com canonical name = tecmint.com.
Name: tecmint.com
Address: 50.116.66.136
```

For more **NSLOOKUP Command**, read the article on [8 Linux Nslookup Command Examples](#).

7. ROUTE Command

route command also shows and manipulate **ip** routing table. To see default routing table in **Linux**, type the following command.

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.50.0   *               255.255.255.0   U      0      0        0 eth0
link-local      *               255.255.0.0    U      1002   0        0 eth0
default         192.168.50.1   0.0.0.0        UG     0      0        0 eth0
```

Adding, deleting routes and default Gateway with following commands.

Route Adding

```
# route add -net 10.10.10.0/24 gw 192.168.0.1
```

Route Deleting

```
# route del -net 10.10.10.0/24 gw 192.168.0.1
```

Adding default Gateway

```
# route add default gw 192.168.0.1
```

8. HOST Command

host command to find name to **IP** or **IP** to name in **IPv4** or **IPv6** and also query **DNS** records.

```
# host www.google.com
www.google.com has address 173.194.38.180
www.google.com has address 173.194.38.176
www.google.com has address 173.194.38.177
www.google.com has address 173.194.38.178
www.google.com has address 173.194.38.179
www.google.com has IPv6 address 2404:6800:4003:802::1014
```

Using **-t** option we can find out DNS Resource Records like **CNAME**, **NS**, **MX**, **SOA** etc.

```
# host -t CNAME www.redhat.com
www.redhat.com is an alias for wildcard.redhat.com.edgekey.net.
```

9. ARP Command

ARP (Address Resolution Protocol) is useful to **view** / **add** the contents of the kernel's **ARP tables**. To see default table use the command as.

```
# arp -e
Address           HWtype  HWaddress          Flags Mask
Iface
192.168.50.1     ether    00:50:56:c0:00:08  C
eth0
```

10. ETHTOOL Command

ethtool is a replacement of **mii-tool**. It is to view, setting speed and duplex of your **Network Interface Card (NIC)**. You can set duplex permanently in **/etc/sysconfig/network-scripts/ifcfg-eth0** with **ETHTOOL_OPTS** variable.

```
# ethtool eth0
Settings for eth0:
Current message level: 0x00000007 (7)
Link detected: yes
```

11. IWCONFIG Command

iwconfig command in **Linux** is use to configure a **wireless network interface**. You can see and set the basic **Wi-Fi** details like **SSID** channel and encryption. You can refer man page of **iwconfig** to know more.

```
# iwconfig [interface]
```

12. HOSTNAME Command

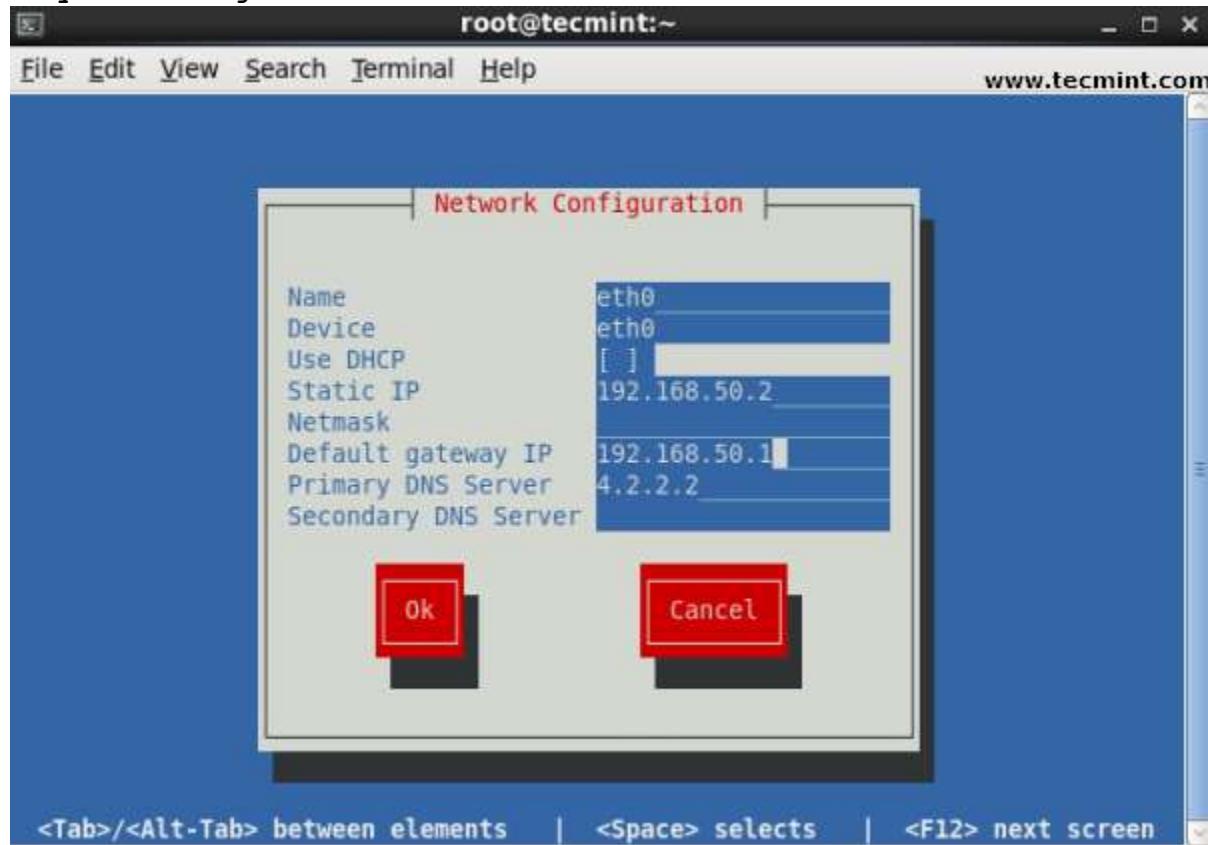
hostname is to identify in a network. Execute **hostname** command to see the hostname of your box. You can set hostname permanently in **/etc/sysconfig/network**. Need to reboot box once set a proper hostname.

```
# hostname  
tecmint.com
```

13. GUI tool system-config-network

Type **system-config-network** in command prompt to configure network setting and you will get nice **Graphical User Interface (GUI)** which may also use to configure **IP Address, Gateway, DNS** etc. as shown below image.

```
# system-config-network
```



Linux GUI Network Configuration Tool

This article can be useful for day to day use of **Linux Network administrator** in **Linux / Unix-like operating system**. Kindly share through our comment box if we missed out.

Learn The Basics of How Linux I/O (Input/Output) Redirection Works

by [Aaron Kili](#) | Published: January 13, 2017 | Last Updated: January 13, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

One of the most important and [interesting topics under Linux administration](#) is I/O redirection. This feature of the command line enables you to redirect the input and/or output of commands from and/or to files, or join multiple commands together using pipes to form what is known as a “**command pipeline**”.

All the commands that we run fundamentally produce two kinds of output:

1. the command result – data the program is designed to produce, and
2. the program status and error messages that informs a user of the program execution details.

In Linux and other Unix-like systems, there are three default files named below which are also identified by the shell using file descriptor numbers:

1. **stdin or 0** – it’s connected to the keyboard, most programs read input from this file.
2. **stdout or 1** – it’s attached to the screen, and all programs send their results to this file and
3. **stderr or 2** – programs send status/error messages to this file which is also attached to the screen.

Therefore, I/O redirection allows you to alter the input source of a command as well as where its output and error messages are sent to. And this is made possible by the “<” and “>” redirection operators.

How To Redirect Standard Output to File in Linux

You can redirect standard output as in the example below, here, we want to store the output of the [top command](#) for later inspection:

```
$ top -bn 5 >top.log
```

Where the flags:

1. **-b** – enables **top** to run in batch mode, so that you can redirect its output to a file or another command.
2. **-n** – specifies the number of iterations before the command terminates.

You can view the contents of `top.log` file using [cat command](#) as follows:

```
$ cat top.log
```

To append the output of a command, use the “`>>`” operator.

For instance to append the output of [top command](#) above in the `top.log` file especially within a script (or on the command line), enter the line below:

```
$ top -bn 5 >>top.log
```

Note: Using the file descriptor number, the output redirect command above is the same as:

```
$ top -bn 5 1>top.log
```

How To Redirect Standard Error to File in Linux

To redirect standard error of a command, you need to explicitly specify the file descriptor number, `2` for the shell to understand what you are trying to do.

For example the [ls command](#) below will produce an error when executed by a normal system user without root privileges:

```
$ ls -l /root/
```

You can redirect the standard error to a file as below:

```
$ ls -l /root/ 2>ls-error.log
$ cat ls-error.log
aaronkilik@tecmint ~ $ ls -l /root/
ls: cannot open directory '/root/': Permission denied
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $ ls -l /root/ 2>ls-error.log
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $ cat ls-error.log
ls: cannot open directory '/root/': Permission denied
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $
```

Redirect Standard Error to File

In order to append the standard error, use the command below:

```
$ ls -l /root/ 2>>ls-error.log
```

How To Redirect Standard Output/ Error To One File

It is also possible to capture all the output of a command (both standard output and standard error) into a single file. This can be done in two possible ways by specifying the file descriptor numbers:

1. The first is a relatively old method which works as follows:

```
$ ls -l /root/ >ls-error.log 2>&1
```

The command above means the shell will first send the output of the [ls command](#) to the file **ls-error.log** (using `>ls-error.log`), and then writes all error messages to the file descriptor **2** (standard output) which has been redirected to the file **ls-error.log** (using `2>&1`). Implying that standard error is also sent to the same file as standard output.

2. The second and direct method is:

```
$ ls -l /root/ &>ls-error.log
```

You can as well append standard output and standard error to a single file like so:

```
$ ls -l /root/ &>>ls-error.log
```

How To Redirect Standard Input to File

Most if not all commands get their input from standard input, and by default standard input is attached to the keyboard.

To redirect standard input from a file other than the keyboard, use the “`<`” operator as below:

```
$ cat <domains.list
aaronkilik@tecmint ~ $ cat <domains.list
tecmint.com
news.tecmint.com
linuxsay.com
windowsmint.com
aaronkilik@tecmint ~ $
```

Redirect Standard Input to File

How To Redirect Standard Input/Output to File

You can perform standard input, standard output redirection at the same time using [sort command](#) as below:

```
$ sort <domains.list >sort.output
```

How to Use I/O Redirection Using Pipes

To redirect the output of one command as input of another, you can use pipes, this is a powerful means of building useful command lines for complex operations.

For example, the command below will [list the top five recently modified files](#).

```
$ ls -lt | head -n 5
```

Here, the options:

1. `-l` – enables long listing format
2. `-t` – [sort by modification time with the newest files](#) are shown first
3. `-n` – specifies the number of header lines to show

Important Commands for Building Pipelines

Here, we will briefly review two important commands for building command pipelines and they are:

xargs which is used to build and execute command lines from standard input. Below is an example of a pipeline which uses **xargs**, this command is used to [copy a file into multiple directories in Linux](#):

```
$ echo /home/aaronkilik/test/ /home/aaronkilik/tmp | xargs -n 1 cp -v  
/home/aaronkilik/bin/sys_info.sh  
aaronkilik@tecmint ~ $ echo /home/aaronkilik/test/ /home/aaronkilik/tmp  
xargs -n 1 cp -v /home/aaronkilik/bin/sys_info.sh  
'/home/aaronkilik/bin/sys_info.sh' -> '/home/aaronkilik/test/sys_info.sh'  
'/home/aaronkilik/bin/sys_info.sh' -> '/home/aaronkilik/tmp/sys_info.sh'  
aaronkilik@tecmint ~ $ █
```

Copy Files to Multiple Directories

And the options:

1. `-n 1` – instructs xargs to use at most one argument per command line and send to the [cp command](#)
2. `cp` – copies the file
3. `-v` – [displays progress of copy command](#).

For more usage options and info, read through the **xargs** man page:

```
$ man xargs
```

A **tee** command reads from standard input and writes to standard output and files. We can demonstrate how **tee** works as follows:

```
$ echo "Testing how tee command works" | tee file1
```

```
aaronkilik@tecmint ~ $ echo "Testing how tee command works" | tee file1
Testing how tee command works
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $ cat file1
Testing how tee command works
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $ █
```

tee Command Example

10 Useful Sudoers Configurations for Setting ‘sudo’ in Linux

by [Aaron Kili](#) | Published: January 6, 2017 | Last Updated: January 12, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In Linux and other Unix-like operating systems, only the **root** user can run all commands and perform certain critical operations on the system such as install and update, remove packages, [create users and groups](#), modify important system configuration files and so on.

However, a system administrator who assumes the role of the root user can permit other normal system users with the help of [sudo command](#) and a few configurations to run some commands as well as carry out a number of vital system operations including the ones mentioned above.

Alternatively, the system administrator can share the root user password (which is not a recommended method) so that normal system users have access to the root user account via **su** command.

sudo allows a permitted user to execute a command as root (or another user), as specified by the security policy:

1. It reads and parses **/etc/sudoers**, looks up the invoking user and its permissions,
2. then prompts the invoking user for a password (normally the user’s password, but it can as well be the target user’s password. Or it can be skipped with NOPASSWD tag),
3. after that, sudo creates a child process in which it calls **setuid()** to switch to the target user
4. next, it executes a shell or the command given as arguments in the child process above.

Below are ten **/etc/sudoers** file configurations to modify the behavior of **sudo** command using **Defaults** entries.

```
$ sudo cat /etc/sudoers
/etc/sudoers File
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
```

```

Defaults          secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
Defaults          logfile="/var/log/sudo.log"
Defaults          lecture="always"
Defaults          badpass_message="Password is wrong, please try again"
Defaults          passwd_tries=5
Defaults          insults
Defaults          log_input,log_output

```

Types of Defaults Entries

Defaults	parameter,	parameter_list	#affect all users on any host
Defaults@Host_List	parameter,	parameter_list	#affects all users on a specific host
Defaults:User_List	parameter,	parameter_list	#affects a specific user
Defaults!Cmnd_List	parameter,	parameter_list	#affects a specific command
Defaults>Runas_List	parameter,	parameter_list	#affects commands being run as a specific user

For the scope of this guide, we will zero down to the first type of **Defaults** in the forms below. Parameters may be flags, integer values, strings, or lists.

You should note that flags are implicitly boolean and can be turned off using the '!' operator, and lists have two additional assignment operators, += (add to list) and -= (remove from list).

```

Defaults      parameter
OR
Defaults      parameter=value
OR
Defaults      parameter -=value
Defaults      parameter +=value
OR
Defaults      !parameter

```

1. Set a Secure PATH

This is the path used for every command run with sudo, it has two importances:

1. Used when a system administrator does not trust sudo users to have a secure PATH environment variable
2. To separate “root path” and “user path”, only users defined by **exempt_group** are not affected by this setting.

To set it, add the line:

```

Defaults
secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

```

2. Enable sudo on TTY User Login Session

To enable sudo to be invoked from a real **tty** but not through methods such as **cron** or **cgi-bin** scripts, add the line:

```
Defaults    requiretty
```

3. Run Sudo Command Using a pty

A few times, attackers can run a malicious program (such as a virus or malware) using sudo, which would again fork a background process that remains on the user's terminal device even when the main program has finished executing.

To avoid such a scenario, you can configure sudo to run other commands only from a **psuedo-pty** using the **use_pty** parameter, whether I/O logging is turned on or not as follows:

```
Defaults    use_pty
```

4. Create a Sudo Log File

By default, sudo logs through **syslog(3)**. However, to specify a custom log file, use the **logfile** parameter like so:

```
Defaults    logfile="/var/log/sudo.log"
```

To log hostname and the four-digit year in the custom log file, use **log_host** and **log_year** parameters respectively as follows:

```
Defaults    log_host, log_year, logfile="/var/log/sudo.log"
```

Below is an example of a custom sudo log file:

```
aaronkilik@tecmint ~ $ cat /var/log/sudo.log
Dec 28 11:55:04 : aaronkilik : TTY=pts/0 ; PWD=/home/aaronkilik ; USER=r
    COMMAND=/usr/sbin/visudo
Dec 28 12:03:19 : aaronkilik : TTY=pts/0 ; PWD=/home/aaronkilik ; USER=r
    COMMAND=/usr/sbin/visudo
Dec 28 12:03:44 : aaronkilik : TTY=pts/0 ; PWD=/home/aaronkilik ; USER=r
    COMMAND=/usr/sbin/visudo
Dec 28 12:04:01 : aaronkilik : TTY=pts/0 ; PWD=/home/aaronkilik ; USER=r
    COMMAND=/usr/sbin/visudo
Dec 28 12:04:39 : aaronkilik : TTY=pts/1 ; PWD=/home/aaronkilik ; USER=r
    COMMAND=/usr/sbin/visudo
Dec 28 12:05:58 : aaronkilik : 2 incorrect password attempts ; TTY=pts/0
    PWD=/home/aaronkilik ; USER=root ; COMMAND=/usr/sbin/visudo
Dec 28 12:06:58 : aaronkilik : 3 incorrect password attempts ; TTY=pts/0
    PWD=/home/aaronkilik ; USER=root ; COMMAND=/usr/sbin/visudo
Dec 28 12:21:02 : aaronkilik : 1 incorrect password attempt ; TTY=pts/0
    PWD=/home/aaronkilik ; USER=root ; COMMAND=/usr/sbin/visudo
Dec 28 12:46:54 : aaronkilik : TTY=pts/1 ; PWD=/home/aaronkilik ; USER=r
```

Create Custom Sudo Log File

5. Log Sudo Command Input/Output

The **log_input** and **log_output** parameters enable sudo to run a command in pseudo-tty and log all user input and all output sent to the screen respectively.

The default I/O log directory is **/var/log/sudo-io**, and if there is a session sequence number, it is stored in this directory. You can specify a custom directory through the **ilog_dir** parameter.

```
Defaults  log_input, log_output
```

There are some escape sequences supported such as `%{seq}` which expands to a monotonically increasing base-36 sequence number, such as 000001, where every two digits are used to form a new directory, e.g. **00/00/01** as in the example below:

```
$ cd /var/log/sudo-io/
$ ls
$ cd 00/00/01
$ ls
$ cat log
```

```
tecmint aaronkilik # cd /var/log/sudo-io/
tecmint sudo-io # ls
00 seq
tecmint sudo-io # cd 00/00/01/
tecmint 01 # ls
log stderr stdin stdout timing ttyin ttyout
tecmint 01 # cat log
1483434005:aaronkilik:root:::dev/pts/2:17:67
/home/aaronkilik
/usr/bin/apt-get update
tecmint 01 #
```

Log sudo Input Output

You can view the rest of the files in that directory using the [cat command](#).

6. Lecture Sudo Users

To lecture sudo users about password usage on the system, use the **lecture** parameter as below.

It has 3 possible values:

1. always – always lecture a user.
2. once – only lecture a user the first time they execute sudo command (this is used when no value is specified)
3. never – never lecture the user.

```
Defaults lecture="always"
```

Additionally, you can set a custom lecture file with the **lecture_file** parameter, type the appropriate message in the file:

```
Defaults lecture_file="/path/to/file"
aaronkilik@tecmint ~ $ sudo apt-get update

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for aaronkilik:
```

Lecture Sudo Users

7. Show Custom Message When You Enter Wrong sudo Password

When a user enters a wrong password, a certain message is displayed on the command line. The default message is “**sorry, try again**”, you can modify the message using the **badpass_message** parameter as follows:

```
Defaults badpass_message="Password is wrong, please try again"
```

8. Increase sudo Password Tries Limit

The parameter **passwd_tries** is used to specify the number of times a user can try to enter a password.

The default value is 3:

```
Defaults passwd_tries=5
aaronkilik@tecmint ~ $ sudo apt-get update

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for aaronkilik:
Password is wrong, please try again
[sudo] password for aaronkilik:
Password is wrong, please try again
[sudo] password for aaronkilik:
Password is wrong, please try again
[sudo] password for aaronkilik:
Password is wrong, please try again
[sudo] password for aaronkilik:
Password is wrong, please try again
[sudo] password for aaronkilik:
sudo: 5 incorrect password attempts
aaronkilik@tecmint ~ $
```

Increase Sudo Password Attempts

To set a password timeout (default is 5 minutes) using **passwd_timeout** parameter, add the line below:

```
Defaults passwd_timeout=2
```

9. Let Sudo Insult You When You Enter Wrong Password

In case a user types a wrong password, sudo will display insults on the terminal with the insults parameter. This will automatically turn off the **badpass_message** parameter.

```
Defaults insults
aaronkilik@tecmint ~ $ sudo apt-get update

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for aaronkilik:
It can only be attributed to human error.
[sudo] password for aaronkilik:
Take a stress pill and think things over.
[sudo] password for aaronkilik:
... and it used to be so popular...
[sudo] password for aaronkilik: █
```

Let's Sudo Insult You When Enter Wrong Password

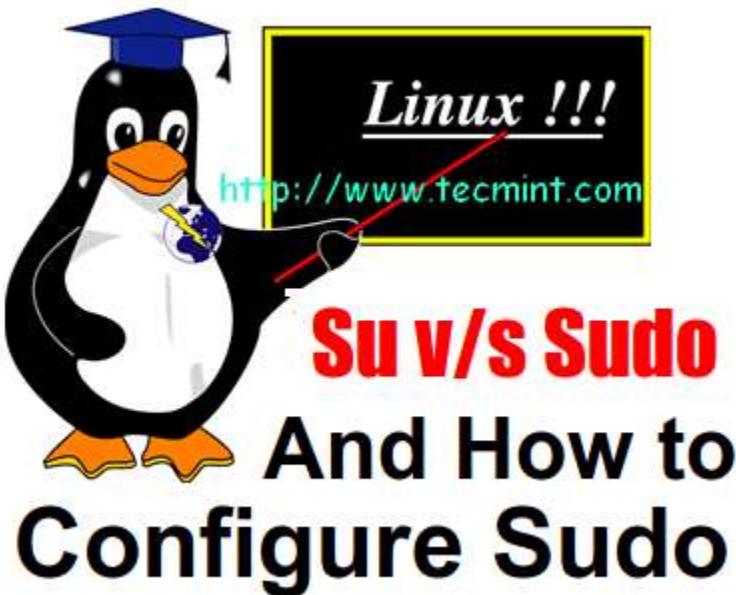
Difference Between su and sudo and How to Configure sudo in Linux

by [Editor](#) | Published: March 19, 2014 | Last Updated: January 12, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Linux System is much secured than any of its counterpart. One of the way to [implement security in Linux](#) is the user management policy and user permission and normal users are not authorized to perform any system operations.

If a normal user needs to perform any system wide changes he needs to use either ‘**su**’ or ‘**sudo**’ command.



Linux: su v/s sudo

NOTE – This article is more applicable to **Ubuntu** based distributions, but also applicable to most of the popular **Linux** distributions.

‘su’ Vs ‘sudo’

‘**su**’ forces you to share your **root password** to other users whereas ‘**sudo**’ makes it possible to execute system commands without **root** password. ‘**sudo**’ lets you use your own password to execute system commands i.e., delegates system responsibility without **root** password.

What is ‘sudo’?

‘**sudo**’ is a root binary *setuid*, which executes root commands on behalf of authorized users and the users need to enter their own password to execute system command followed by ‘**sudo**’.

Who can execute ‘sudo’?

We can run ‘**/usr/sbin/visudo**’ to add/remove the list of users who can execute ‘**sudo**’.

```
$ sudo /usr/sbin/visudo
```

A screen shot of ‘**/usr/sbin/visudo**’ file, looks something like this:



The **sudo** list looks like the below string, by default:

```
root ALL=(ALL) ALL
```

Note: You must be *root* to edit **/usr/sbin/visudo** file.

Granting sudo Access

In many situation, **System Administrator**, specially new to the field finds the string “**root ALL=(ALL) ALL**” as a template and grants unrestricted access to others which may be potentially very harmful.

Editing ‘**/usr/sbin/visudo**’ file to something like the below pattern may really be very dangerous, unless you believe all the listed users completely.

```
root ALL=(ALL) ALL
adam ALL=(ALL) ALL
tom ALL=(ALL) ALL
mark ALL=(ALL) ALL
```

Parameters of sudo

A properly configured ‘**sudo**‘ is very flexible and number of commands that needs to be run may be precisely configured.

The Syntax of configured ‘**sudo**‘ line is:

```
User_name Machine_name=(Effective_user) command
```

The above Syntax can be divided into four parts:

1. **User_name**: This is the name of ‘**sudo**‘ user.
2. **Machine_name**: This is the host name, in which ‘**sudo**‘ command is valid. Useful when you have lots of host machines.
3. **(Effective_user)**: The ‘Effective user’ that are allowed to execute the commands. This column lets you allows users to execute System Commands.
4. **Command**: command or a set of commands which user may run.

Suggested Read: 10 Useful Sudoers Configurations for Setting ‘sudo’ in Linux

Some of the Situations, and their corresponding ‘**sudo**‘ line:

Q1. You have a user **mark** which is a Database Administrator. You are supposed to provide him all the access on Database Server (**beta.database_server.com**) only, and not on any host.

For the above situation the ‘**sudo**‘ line can be written as:

```
mark beta.database_server.com=(ALL) ALL
```

Q2. You have a user ‘**tom**‘ which is supposed to execute system command as user other than root on the same Database Server, above Explained.

For the above situation the ‘**sudo**‘ line can be written as:

```
mark beta.database_server.com=(tom) ALL
```

Q3. You have a sudo user ‘**cat**‘ which is supposed to run command ‘**dog**‘ only.

To implement the above situation, we can write ‘**sudo**‘ as:

```
mark beta.database_server.com=(cat) dog
```

Q4. What if the user needs to be granted several commands?

If the number of commands, user is supposed to run is under **10**, we can place all the commands alongside, with white space in between them, as shown below:

```
mark beta.database_server.com=(cat) /usr/bin/command1 /usr/sbin/command2  
/usr/sbin/command3 ...
```

If this list of command varies to the range, where it is literally not possible to type each command manually we need to use **aliases**. Aliases! Yeah the Linux utility where a long-lengthy command or a list of command can be referred as a small and easy keyword.

A few **alias** Examples, which can be used in place of entry in ‘**sudo**‘ configuration file.

```
User_Alias ADMINS=tom,jerry,adam  
user_Alias WEBMASTER=henry,mark  
WEBMASTERS WEBSERVERS=(www) APACHE  
Cmnd_Alias PROC=/bin/kill,/bin/killall, /usr/bin/top
```

It is possible to specify a **System Groups**, in place of users, that belongs to that group just sufficing ‘%’ as below:

```
%apacheadmin WEBSERVERS=(www) APACHE
```

Q5. How about executing a ‘**sudo**‘ command without entering password?

We can execute a ‘**sudo**‘ command without entering password by using ‘**NOPASSWD**‘ flag.

```
adam ALL=(ALL) NOPASSWD: PROCS
```

Here the user ‘**adam**‘ can execute all the commands **aliased** under “**PROCS**”, without entering password.

Suggested Read: [Let Sudo Insult You When You Enter Incorrect Password](#)

“**sudo**” provides you a robust and safe environment with loads of flexibility as compared to ‘**su**‘. Moreover “**sudo**” configuration is easy. Some Linux distributions have “**sudo**” enabled by default while most of the distros of today needs you to enable it as a **Security Measure**.

To add an user (bob) to sudo just run the below command as root.

```
adduser bob sudo
```

That’s all for now. I’ll be here again with another Interesting article. Till then stay tuned and connected to Tecmint. Don’t forget to provide us with your valuable feedback in our comment section.

Set Date and Time for Each Command You Execute in Bash History

by [Ravi Saive](#) | Published: January 14, 2017 | January 14, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

By default, all commands executed by **Bash** on the command line are stored in history buffer or recorded in a file called **~/.bash_history**. This means that a system administrator can view a list of commands executed by users on the system or a user can view his/her command history using the [history command](#) like so.

```
$ history
aaronkilik@tecmint ~ $ history
 610 sudo systemctl status iptables
 611 sudo systemctl start iptables
 612 service firewalld start
 613 service firewalld status
 614 ping 4.2.2.2
 615 find / -name my.conf
 616 sudo find / -name my.conf
 617 find / -name my.conf
 618 find / -name "my.conf"
 619 echo $MYSQL_HOME
 620 mysql --help
 621 mysqld --help
 622 mysqld --help --verbose
 623 locate my.conf | less
 624 locate my.conf
 625 locate "my.conf"
 626 strace mysql ";" 2>&1 | grep cnf
 627 strace mysql ";" 2>&1 | grep cnf | grep mysql
 628 man strace
 629 strace mysql 2>&1 | grep cnf | grep mysql
 630 strace mysql ";" 2>&1 | grep cnf | grep mysql
```

Linux History Command

From the output of the [history command](#) above, the **date** and **time** when a command was executed is not shown. This is the default setting on most if not all Linux distributions.

In this article, we will explain how you can configure time stamp information when each command in the Bash history was executed to be displayed.

The **date** and **time** associated with each history entry can be written to the history file, marked with the history comment character by setting the **HISTTIMEFORMAT** variable.

There are two possible ways of doing this: one does it temporarily while the other makes it permanent.

To set **HISTTIMEFORMAT** variable temporarily, export it as below on the command line:

```
$ export HISTTIMEFORMAT='%F %T'
```

In the export command above, the time stamp format:

1. **%F** – expands to full date same, as **%Y-%m-%d** (year-month-date).
2. **%T** – expands to time; same as **%H:%M:%S** (hour:minute:seconds).

Read through the [date command](#) man page for additional usage information:

```
$ man date
```

Then check your command history as follows:

```
$ history
1593 2017-01-11 01:37:08date
1594 2017-01-11 01:37:09history
1595 2017-01-11 01:41:09man bash | grep history
1596 2017-01-11 01:42:07man date
1597 2017-01-11 02:07:26unset HISTTIMEFORMAT
1598 2017-01-11 02:07:30history
1599 2017-01-11 02:07:35export HISTTIMEFORMAT='%F %T'
1600 2017-01-11 02:07:38unset HISTTIMEFORMAT
1601 2017-01-11 02:07:41export HISTTIMEFORMAT='%F %T'
1602 2017-01-11 02:07:44history
1603 2017-01-11 02:07:54unset HISTTIMEFORMAT
1604 2017-01-11 02:07:56history
1605 2017-01-11 02:09:07df
1606 2017-01-11 02:09:11date
1607 2017-01-11 02:09:16du
1608 2017-01-11 02:09:26ping localhost
1609 2017-01-11 02:09:34history
1610 2017-01-11 02:09:44history | head
1611 2017-01-11 02:11:47export HISTTIMEFORMAT='%F %T'
1612 2017-01-11 02:11:54history
aaronkilik@tecmint ~ $
```

Display Linux Command History with Date and Time

However, if you want to configure this variable permanently, open the file `~/.bashrc` with your favorite editor:

```
$ vi ~/.bashrc
```

And add the line below in it (you mark it with a comment as your own configuration):

```
#my config  
export HISTTIMEFORMAT='%F %T'
```

Save the file and exit, afterwards, run the command below to effect the changes made to the file:

```
$ source ~/.bashrc
```

That's all! Do share with us any interesting history command tips and tricks or your thoughts about this guide via the comment section below.

How to Compress and Decompress a .bz2 File in Linux

by [Aaron Kili](#) | Published: November 3, 2016 | Last Updated: July 17, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

To compress a file(s), is to significantly decrease the size of the file(s) by encoding data in the file(s) using less bits, and it is normally a useful practice [during backup and transfer of a file\(s\)](#) over a network. On the other hand, decompressing a file(s) means restoring data in the file(s) to its original state.

Suggested Read: [Learn Linux ‘tar’ Command with This 18 Examples](#)

There are several [file compression and decompression tools](#) available in Linux such as **gzip**, **7-zip**, **Lrzip**, **PeaZip** and many more.

In this tutorial, we will look at how to compress and decompress .bz2 files using the bzip2 tool in Linux.

Bzip2 is a well known compression tool and it's available on most if not all the major Linux distributions, you can use the appropriate command for your distribution to install it.

```
$ sudo apt install bzip2      [On Debian/Ubuntu]  
$ sudo yum install bzip2      [On CentOS/RHEL]  
$ sudo dnf install bzip2      [On Fedora 22+]
```

The conventional syntax of using **bzip2** is:

```
$ bzip2 option(s) filenames
```

How to Use “bzip2” to Compress Files in Linux

You can compress a file as below, where the flag **-z** enables file compression:

```
$ bzip2 filename  
OR  
$ bzip2 -z filename
```

To compress a .tar file, use the command format:

```
$ bzip2 -z backup.tar
```

Important: By default, **bzip2** deletes the input files during compression or decompression, to keep the input files, use the **-k** or **--keep** option.

In addition, the **-f** or **--force** flag will force **bzip2** to overwrite an existing output file.

```
----- To keep input file -----  
$ bzip2 -zk filename  
$ bzip2 -zk backup.tar
```

You can as well set the block size to **100k** upto **900k**, using **-1** or **--fast** to **-9** or **-best** as shown in the below examples:

```
$ bzip2 -k1 Etcher-linux-x64.AppImage  
$ ls -lh Etcher-linux-x64.AppImage.bz2  
$ bzip2 -k9 Etcher-linux-x64.AppImage  
$ bzip2 -kf9 Etcher-linux-x64.AppImage  
$ ls -lh Etcher-linux-x64.AppImage.bz2
```

The screenshot below shows how to use options to keep the input file, force **bzip2** to overwrite an output file and set the block size during compression.

```
aaronkilik@tecmint ~ $ bzip2 -k1 Etcher-linux-x64.AppImage  
aaronkilik@tecmint ~ $  
aaronkilik@tecmint ~ $ ls -lh Etcher-linux-x64.AppImage.bz2  
-rwxr-xr-x 1 aaronkilik aaronkilik 73M Jul 26 01:34 Etcher-linux-x64.AppImage.bz2  
aaronkilik@tecmint ~ $  
aaronkilik@tecmint ~ $ bzip2 -k9 Etcher-linux-x64.AppImage  
bzip2: Output file Etcher-linux-x64.AppImage.bz2 already exists.  
aaronkilik@tecmint ~ $  
aaronkilik@tecmint ~ $ bzip2 -kf9 Etcher-linux-x64.AppImage  
aaronkilik@tecmint ~ $  
aaronkilik@tecmint ~ $ ls -lh Etcher-linux-x64.AppImage.bz2  
-rwxr-xr-x 1 aaronkilik aaronkilik 73M Jul 26 01:34 Etcher-linux-x64.AppImage.bz2  
aaronkilik@tecmint ~ $  
aaronkilik@tecmint ~ $ █
```

Compress Files Using bzip2 in Linux

How to Use “bzip2” to Decompress Files in Linux

To decompress a **.bz2** file, make use of the **-d** or **--decompress** option like so:

```
$ bzip2 -d filename.bz2
```

Note: The file must end with a **.bz2** extension for the command above to work.

```
$ bzip2 -vd Etcher-linux-x64.AppImage.bz2
```

```
$ bzip2 -vfd Etcher-linux-x64.AppImage.bz2
$ ls -l Etcher-linux-x64.AppImage
aaronkilik@tecmint ~ $ bzip2 -vd Etcher-linux-x64.AppImage.bz2
bzip2: Output file Etcher-linux-x64.AppImage already exists.
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $ bzip2 -vfd Etcher-linux-x64.AppImage.bz2
Etcher-linux-x64.AppImage.bz2: done
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $ ls -l Etcher-linux-x64.AppImage
-rwxr-xr-x 1 aaronkilik aaronkilik 76087296 Jul 26 01:34 Etcher-linux-x64.AppImage
aaronkilik@tecmint ~ $ █
```

Decompress bzip2 File in Linux

To view the **bzip2** help page and **man page**, type the command below:

```
$ bzip2 -h
$ man bzip2
```

Lastly, with the simple elaborations above, I believe you are now capable of compressing and decompressing .bz2 files using the **bzip2** tool in Linux. However, for any questions or feedback, reach us using the comment section below.

How to Find a Process Name Using PID Number in Linux

by [Aaron Kili](#) | Published: November 1, 2016 | Last Updated: October 31, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this article, we will look at [how to find a process name](#) by its process identification number (PID). Before we dive into the actual solution, let us briefly talk about how processes are created and identified by Linux.

Every time a user or the system (Linux) launches a program, the kernel will create a process. A process holds execution details of the program in memory such as its input and output data, variables and so on.

Importantly, since Linux is a multitasking operating system, it executes several programs simultaneously, and this means each process must be identified specifically.

The kernel identifies each process using a process **ID (PID)**, a every instance of process must have a unique **PID** from other processes which is assigned when the process is invoked, to avoid any execution errors.

The [/proc file system](#) stores information about [currently running processes on your system](#), it contains directories for each process.

Use the [ls command](#) to list its contents, however, the list may be long, so employ a pipeline and the [less utility to view the /proc contents](#) in a more convenient way as below:

```
$ ls /proc
OR
$ ls /proc | less
```

List /proc File System

1	168	2230	25	329	584	7386	83	driver	schedstat			
10	169	2234	2503	33	603	74	830	execdomains	scsi			
1070	17	2247	2507	34	610	7411	833	fb	self			
1081	1702	2256	2523	349	611	7423	836	filesystems	slabinfo			
109	1714	2258	253	35	612	745	839	fs	softirqs			
11	173	2266	2551	36	613	746	84	interrupts	stat			
110	1760	2273	26	362	62	75	844	iomem	swaps			
1188	1763	2278	2688	3642	63	7533	85	ioports	sys			
12	1769	2282	2694	3643	64	7589	86	irq	sysrq-			
		trigger										
1204	177	2283	2695	37	6436	76	860	kallsyms	sysvipc			
1209	1773	2285	2698	38	65	7619	87	kcore	thread-			
		self										
1254	18	2287	2699	39	66	7689	9	keys	timer_list			

13	1847	2295	27	3974	67	7690	94		key-users
timer_stats									
15	1914	23	2702	3976	68	77	977	kmsg	tty
152	1917	2308	28	4273	6897	7725	981	kpagecgrou	uptime
153	1918	2309	280	4374	69	7729	987	kpagecount	version
154	1938	2310	2815	4392	6969	7733	997	kpageflags	
version_signature									
155	1956	2311	2817	44	6980	78	acpi	loadavg	
vmallocinfo									
156	1981	2315	282	45	7	79	asound	locks	vmstat
1565	1986	2316	283	4543	70	790	buddyinfo	mdstat	zoneinfo
1567	1988	2317	29	46	71	8	bus	meminfo	
157	2	2324	2935	461	7102	80	cgroups	misc	
1579	20	2347	2944	4686	72	808	cmdline	modules	
158	2010	2354	3	47	73	81	consoles	mounts	
1584	2043	2436	30	4700	7304	810	cpuinfo	mtrr	
159	2044	2437	3016	5	7311	815	crypto	net	
1590	21	2442	31	515	7322	82	devices	pagetypeinfo	
16	2167	2443	318	5273	7347	820	diskstats	partitions	
160	22	2492	32	5274	7367	823	dma	sched_debug	

From the screenshot above, the numbered directories store information files about the processes in execution, where each number corresponds to a **PID**.

Below is the list of files for **systemd** process with **PID 1**:

```
$ ls /proc/1
Show SystemD Process PID
ls: cannot read symbolic link '/proc/1/cwd': Permission denied
ls: cannot read symbolic link '/proc/1/root': Permission denied
ls: cannot read symbolic link '/proc/1/exe': Permission denied
attr      coredump_filter  gid_map    mountinfo   oom_score   schedstat
status
autogroup cpuset          io         mounts     oom_score_adj sessionid
syscall
auxv      cwd             limits     mountstats pagemap     setgroups
task
cgroup    environ         loginuid   net        personality smaps
timers
clear_refs exe            map_files  ns         projid_map  stack
uid_map
cmdline   fd              maps       numa_maps  root        stat
wchan
comm      fdinfo          mem       oom_adj    sched      statm
```

You can [monitor processes and their PIDs](#) using traditional Linux commands such as [ps](#), [top](#) and relatively new [glances](#) command plus many more as in the examples below:

```
$ ps aux
Show Running Processes with PID
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1  0.0  0.0 185728  6268 ?        Ss   10:15  0:01 /sbin/init
splash
root      2  0.0  0.0      0      0 ?        S    10:15  0:00 [kthreadd]
```

```

root      3 0.0 0.0    0 0 ?      S 10:15 0:00
[ksoftirqd/0]
root      5 0.0 0.0    0 0 ?      S< 10:15 0:00
[kworker/0:0H]
root      7 0.0 0.0    0 0 ?      S 10:15 0:09 [rcu_sched]
root      8 0.0 0.0    0 0 ?      S 10:15 0:00 [rcu_bh]
root      9 0.0 0.0    0 0 ?      S 10:15 0:00
[migration/0]
root     10 0.0 0.0    0 0 ?      S 10:15 0:00 [watchdog/0]
root     11 0.0 0.0    0 0 ?      S 10:15 0:00 [watchdog/1]
root     12 0.0 0.0    0 0 ?      S 10:15 0:00
[migration/1]
root     13 0.0 0.0    0 0 ?      S 10:15 0:00
[ksoftirqd/1]
root     15 0.0 0.0    0 0 ?      S< 10:15 0:00
[kworker/1:0H]
root     16 0.0 0.0    0 0 ?      S 10:15 0:00 [watchdog/2]
root     17 0.0 0.0    0 0 ?      S 10:15 0:00
[migration/2]
root     18 0.0 0.0    0 0 ?      S 10:15 0:00
[ksoftirqd/2]
root     20 0.0 0.0    0 0 ?      S< 10:15 0:00
[kworker/2:0H]
root     21 0.0 0.0    0 0 ?      S 10:15 0:00 [watchdog/3]
root     22 0.0 0.0    0 0 ?      S 10:15 0:00
[migration/3]
root     23 0.0 0.0    0 0 ?      S 10:15 0:00
[ksoftirqd/3]
root     25 0.0 0.0    0 0 ?      S< 10:15 0:00
[kworker/3:0H]
root     26 0.0 0.0    0 0 ?      S 10:15 0:00 [kdevtmpfs]
root     27 0.0 0.0    0 0 ?      S< 10:15 0:00 [netns]
root     28 0.0 0.0    0 0 ?      S< 10:15 0:00 [perf]
.....

```

Monitor Linux processes using traditional [top command](#).

```
$ top
```

tecmint@TecMint ~/Test												
top - 13:25:10 up 3:09, 1 user, load average: 0.49, 0.50, 0.64												
Tasks: 217 total, 1 running, 216 sleeping, 0 stopped, 0 zombie												
%Cpu(s): 7.1 us, 0.5 sy, 0.0 ni, 91.9 id, 0.5 wa, 0.0 hi, 0.0 si,												
KiB Mem : 8069036 total, 2613988 free, 3085424 used, 2369624 buff/cache												
KiB Swap: 3906556 total, 3906556 free, 0 used. 4204324 avail Mem												
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMM	
2523	tecmint	20	0	2671944	1.329g	177208	S	10.3	17.3	65:28.10	fire	
4273	tecmint	20	0	1704436	240248	66812	S	9.6	3.0	1:19.85	chrom	
2295	tecmint	20	0	1845984	254472	63248	S	6.6	3.2	5:05.86	cinn	
1584	root	20	0	776368	254896	239328	S	1.0	3.2	5:27.42	Xorg	
7367	tecmint	20	0	1176040	155100	73600	S	1.0	1.9	0:18.82	chrom	
2688	tecmint	20	0	1158872	215184	105932	S	0.7	2.7	2:35.14	chrom	
7	root	20	0	0	0	0	S	0.3	0.0	0:10.04	rcu_	
1590	root	20	0	168444	13472	8064	S	0.3	0.2	0:46.16	team	
2503	tecmint	20	0	506948	48464	31944	S	0.3	0.6	0:36.87	gnome	
7769	root	20	0	0	0	0	S	0.3	0.0	0:00.02	kwork	
7777	tecmint	20	0	41948	3780	3108	R	0.3	0.0	0:00.16	top	
1	root	20	0	185728	6268	3920	S	0.0	0.1	0:01.84	system	
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthr	
3	root	20	0	0	0	0	S	0.0	0.0	0:00.09	ksoft	
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kwor	
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_	
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migr	
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	watc	
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.03	watc	
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migr	
13	root	20	0	0	0	0	S	0.0	0.0	0:00.08	ksoft	
15	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kwor	
16	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	watc	
17	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migr	
18	root	20	0	0	0	0	S	0.0	0.0	0:00.10	ksoft	
20	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kwor	
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.04	watc	
22	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migr	
23	root	20	0	0	0	0	S	0.0	0.0	0:00.08	ksoft	

Monitor Linux Processes with top Command

Monitor Linux processes using **glances**, a new real-time process monitoring tool for Linux.

```
$ glances
```

tecmint@TecMint ~/Test									
TecMint (LinuxMint 18 64bit / Linux 4.4.0-21-generic)									
CPU	29.6%	nice:	0.0%	LOAD	4-core	MEM	39.4%	SWAP	
user:	18.5%	irq:	0.0%	1 min:	0.96	total:	7.70G	total:	
system:	11.1%	iowait:	0.0%	5 min:	0.78	used:	3.03G	used:	
idle:	70.4%	steal:	0.0%	15 min:	0.72	free:	4.66G	free:	
NETWORK	Rx/s	Tx/s	TASKS	220 (626 thr), 1 run, 219 slp, 0 oth sor					
enp1s0	0b	0b							
lo	2Kb	2Kb	CPU%	MEM%	PID	USER			
wlp2s0	0b	0b	0.0	0.0	44	root			
			0.0	0.0	21	root			
DISK I/O	R/s	W/s							
ram0	0	0	0.0	0.0	3642	root			
ram1	0	0	0.0	2.6	2944	tecmint			
ram10	0	0	0.0	0.0	168	root			
ram11	0	0	0.0	0.0	810	root			
ram12	0	0	0.0	0.0	8	root			
ram13	0	0	0.0	0.1	1769	tecmint			
ram14	0	0	0.0	0.0	839	syslog			
ram15	0	0	0.0	0.0	25	root			
ram2	0	0	0.0	0.0	31	root			
ram3	0	0	0.0	0.8	2935	tecmint			
ram4	0	0	0.0	0.0	109	root			
ram5	0	0	0.0	0.0	7783	root			
ram6	0	0	0.0	0.1	2699	tecmint			
ram7	0	0	0.0	0.1	3974	tecmint			
ram8	0	0	0.0	0.0	160	root			
ram9	0	0	0.0	0.3	1847	tecmint			
sda1	0	0	0.0	0.1	1981	tecmint			
sda10	0	0	0.0	0.2	1590	root			
sda2	0	0	0.0	0.0	7725	root			
sda3	0	0	0.0	0.0	2258	root			
sda4	0	0	0.0	0.0	23	root			
sda5	0	0							
2016-10-27 13:28:46	No warning or critical alert detected								

Glances – Real Time Linux Processes Monitoring

Learn more about [how to install Glances in Linux systems](#).

Find Out Process PID Number

To find out the **PID** of a process, you can use `pidof`, a simple command to print out the **PID** of a process:

```
$ pidof firefox
$ pidof python
$ pidof cinnamon
```



```
tecmint@TecMint ~/Test
tecmint@TecMint ~/Test $ pidof firefox
2523
tecmint@TecMint ~/Test $ pidof python
2308
tecmint@TecMint ~/Test $ pidof cinnamon
2295
tecmint@TecMint ~/Test $
```

Find Linux Process PID

Coming back to our point of focus, assuming you already know the **PID** of a process, you can print its name using the command form below:

```
$ ps -p PID -o format
```

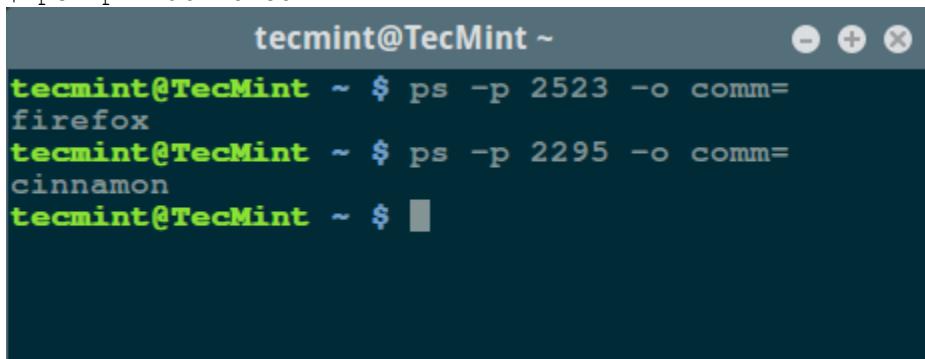
Where:

1. -p specifies the PID
2. -o format enables a user-defined format

Find Out Process Name Using PID Number

In this section, we will see how to find out a process name using its PID number with the help of user defined format i.e `comm=` which means command name, same as the process name.

```
$ ps -p 2523 -o comm=
$ ps -p 2295 -o comm=
```



```
tecmint@TecMint ~
tecmint@TecMint ~ $ ps -p 2523 -o comm=
firefox
tecmint@TecMint ~ $ ps -p 2295 -o comm=
cinnamon
tecmint@TecMint ~ $
```

Find Linux Process Name

For additional usage information and options, look through the **ps man** page.

```
$ man ps
```

If you want to kill a process using its PID number, I suggest you to read [Find and Kill Linux Processes Using its PID](#).

Thats it for the moment, if you know any other better way to find out a process name using **PID**, do share with us via our comment section below

12 Useful Commands For Filtering Text for Effective File Operations in Linux

by [Aaron Kili](#) | Published: January 6, 2017 | January 6, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this article, we will review a number of command line tools that act as filters in Linux. A **filter** is a program that reads standard input, performs an operation upon it and writes the results to standard output.

For this reason, it can be used to process information in powerful ways such as restructuring output to generate useful reports, modifying text in files and many other system administration tasks.

With that said, below are some of the useful file or text filters in Linux.

1. Awk Command

Awk is a remarkable pattern scanning and processing language, it can be used to build useful filters in Linux. You can start using it by reading through our [Awk series Part 1 to Part 13](#).

Additionally, also read through the **awk** man page for more info and usage options:

```
$ man awk
```

2. Sed Command

sed is a powerful stream editor for filtering and transforming text. We've already written a two useful articles on sed, that you can go through it here:

1. [How to use GNU ‘sed’ Command to Create, Edit, and Manipulate files in Linux](#)
2. [15 Useful ‘sed’ Command Tips and Tricks for Daily Linux System Administration Tasks](#)

The sed man page has added control options and instructions:

```
$ man sed
```

3. Grep, Egrep, Fgrep, Rgrep Commands

These filters output lines matching a given pattern. They read lines from a file or standard input, and print all matching lines by default to standard output.

Note: The main program is [grep](#), the variations are simply the same as [using specific grep options](#) as below (and they are still being used for backward compatibility):

```
$ egrep = grep -E  
$ fgrep = grep -F  
$ rgrep = grep -r
```

Below are some basic grep commands:

```
tecmint@TecMint ~ $ grep "aaronkilik" /etc/passwd  
aaronkilik:x:1001:1001::/home/aaronkilik:  
tecmint@TecMint ~ $ cat /etc/passwd | grep "aronkilik"  
aronkilik:x:1001:1001::/home/aaronkilik:
```

You can read more about [What's Difference Between Grep, Egrep and Fgrep in Linux?](#).

4. head Command

head is used to display the first parts of a file, it outputs the first **10** lines by default. You can use the **-n num** flag to specify the number of lines to be displayed:

```
tecmint@TecMint ~ $ head /var/log/auth.log  
Jan  2 10:45:01 TecMint CRON[3383]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Jan  2 10:45:01 TecMint CRON[3383]: pam_unix(cron:session): session closed  
for user root  
Jan  2 10:51:34 TecMint sudo: tecmint : TTY=unknown ; PWD=/home/tecmint ;  
USER=root ; COMMAND=/usr/lib/linuxmint/mintUpdate/checkAPT.py  
Jan  2 10:51:34 TecMint sudo: pam_unix(sudo:session): session opened for user  
root by (uid=0)  
Jan  2 10:51:39 TecMint sudo: pam_unix(sudo:session): session closed for user  
root  
Jan  2 10:55:01 TecMint CRON[4099]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Jan  2 10:55:01 TecMint CRON[4099]: pam_unix(cron:session): session closed  
for user root  
Jan  2 11:05:01 TecMint CRON[4138]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Jan  2 11:05:01 TecMint CRON[4138]: pam_unix(cron:session): session closed  
for user root  
Jan  2 11:09:01 TecMint CRON[4146]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
tecmint@TecMint ~ $ head -n 5 /var/log/auth.log  
Jan  2 10:45:01 TecMint CRON[3383]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Jan  2 10:45:01 TecMint CRON[3383]: pam_unix(cron:session): session closed  
for user root  
Jan  2 10:51:34 TecMint sudo: tecmint : TTY=unknown ; PWD=/home/tecmint ;  
USER=root ; COMMAND=/usr/lib/linuxmint/mintUpdate/checkAPT.py  
Jan  2 10:51:34 TecMint sudo: pam_unix(sudo:session): session opened for user  
root by (uid=0)  
Jan  2 10:51:39 TecMint sudo: pam_unix(sudo:session): session closed for user  
root
```

Learn how to use [head command with tail and cat commands](#) for effective usage in Linux.

5. tail Command

tail outputs the last parts (**10** lines by default) of a file. Use the **-n num** switch to specify the number of lines to be displayed.

The command below will output the last **5** lines of the specified file:

```
tecmint@TecMint ~ $ tail -n 5 /var/log/auth.log
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on 0.0.0.0 port 22.
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on :: port 22.
Jan  6 13:01:27 TecMint sshd[1269]: Received SIGHUP; restarting.
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on 0.0.0.0 port 22.
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on :: port 22.
```

Additionally, **tail** has a special option **-f** for [watching changes in a file in real-time](#) (especially log files).

The following command will enable you monitor changes in the specified file:

```
tecmint@TecMint ~ $ tail -f /var/log/auth.log
Jan  6 12:58:01 TecMint sshd[1269]: Server listening on :: port 22.
Jan  6 12:58:11 TecMint sshd[1269]: Received SIGHUP; restarting.
Jan  6 12:58:12 TecMint sshd[1269]: Server listening on 0.0.0.0 port 22.
Jan  6 12:58:12 TecMint sshd[1269]: Server listening on :: port 22.
Jan  6 13:01:27 TecMint sshd[1269]: Received SIGHUP; restarting.
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on 0.0.0.0 port 22.
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on :: port 22.
Jan  6 13:01:27 TecMint sshd[1269]: Received SIGHUP; restarting.
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on 0.0.0.0 port 22.
Jan  6 13:01:27 TecMint sshd[1269]: Server listening on :: port 22.
```

Read through the **tail** man page for a complete list of usage options and instructions:

```
$ man tail
```

6. sort Command

sort is used to sort lines of a text file or from standard input.

Below is the content of a file named **domains.list**:

```
tecmint@TecMint ~ $ cat domains.list
tecmint.com
tecmint.com
news.tecmint.com
news.tecmint.com
linuxsay.com
linuxsay.com
windowsmint.com
```

windowsmint.com

You can run a simple [sort command](#) to sort the file content like so:

```
tecmint@TecMint ~ $ sort domains.list
linuxsay.com
linuxsay.com
news.tecmint.com
news.tecmint.com
tecmint.com
tecmint.com
windowsmint.com
windowsmint.com
```

You can use **sort** command in many ways, go through some of the useful articles on sort command as follows:

1. [14 Useful Examples of Linux ‘sort’ Command – Part 1](#)
2. [7 Interesting Linux ‘sort’ Command Examples – Part 2](#)
3. [How to Find and Sort Files Based on Modification Date and Time](#)
4. <https://www.tecmint.com/sort-ls-output-by-last-modified-date-and-time/>

7. uniq Command

uniq command is used to report or omit repeated lines, it filters lines from standard input and writes the outcome to standard output.

After running **sort** on an input stream, you can remove repeated lines with **uniq** as in the example below.

To indicate the number of occurrences of a line, use the **-c** option and ignore differences in case while comparing by including the **-i** option:

```
tecmint@TecMint ~ $ cat domains.list
tecmint.com
tecmint.com
news.tecmint.com
news.tecmint.com
linuxsay.com
linuxsay.com
windowsmint.com
tecmint@TecMint ~ $ sort domains.list | uniq -c
2 linuxsay.com
2 news.tecmint.com
2 tecmint.com
1 windowsmint.com
```

Read through the **uniq** man page for further usage info and flags:

```
$ man uniq
```

8. fmt Command

fmt simple optimal text formatter, it reformats paragraphs in specified file and prints results to the standard output.

The following is the content extracted from the file **domain-list.txt**:

```
1.tecmint.com 2.news.tecmint.com 3.linuxsay.com 4.windowsmint.com
```

To reformat the above content to a standard list, run the following command with **-w** switch is used to define the maximum line width:

```
tecmint@TecMint ~ $ cat domain-list.txt
1.tecmint.com 2.news.tecmint.com 3.linuxsay.com 4.windowsmint.com
tecmint@TecMint ~ $ fmt -w 1 domain-list.txt
1.tecmint.com
2.news.tecmint.com
3.linuxsay.com
4.windowsmint.com
```

9. pr Command

pr command converts text files or standard input for printing. For instance on **Debian** systems, you can list all installed packages as follows:

```
$ dpkg -l
```

To organize the list in pages and columns ready for printing, issue the following command.

```
tecmint@TecMint ~ $ dpkg -l | pr --columns 3 -1 20
2017-01-06 13:19                                         Page 1
Desired=Unknown/Install ii  adduser                  ii  apg
| Status=Not/Inst/Conf- ii  adwaita-icon-theme      ii  app-install-data
| / Err?=(none)/Reinst-r ii  adwaita-icon-theme-    ii  apparmor
||/ Name                 ii  alsa-base              ii  apt
+---+====+====+====+====+====+====+====+====+====+
ii  accountsservice      ii  anacron                ii  apt-transport-https
ii  acl                   ii  apache2                ii  apt-utils
ii  acpi-support          ii  apache2-bin           ii  apt-xapian-index
ii  acpid                 ii  apache2-data          ii  aptdaemon
ii  add-apt-key           ii  apache2-utils         ii  aptdaemon-data
2017-01-06 13:19                                         Page 2
ii  aptitude               ii  avahi-daemon        ii  bind9-host
ii  aptitude-common        ii  avahi-utils          ii  binfmt-support
ii  apturl                 ii  aview                 ii  binutils
ii  apturl-common          ii  banshee              ii  bison
ii  archdetect-deb         ii  baobab               ii  blt
ii  aspell                 ii  base-files           ii  blueberry
ii  aspell-en              ii  base-passwd         ii  bluetooth
ii  at-spi2-core           ii  bash                 ii  bluez
ii  attr                  ii  bash-completion     ii  bluez-cups
ii  avahi-autoipd          ii  bc                  ii  bluez-obexd
```

.....

The flags used here are:

1. `--column` defines number of columns created in the output.
2. `-l` specifies page length (default is 66 lines).

10. tr Command

This tool translates or deletes characters from standard input and writes results to standard output.

The syntax for using **tr** is as follows:

```
$ tr options set1 set2
```

Take a look at the examples below, in the first command, `set1([:upper:])` represents the case of input characters (all upper case).

Then `set2([:lower:])` represents the case in which the resultant characters will be. It's same thing in the second example and the escape sequence `\n` means print output on a new line:

```
tecmint@TecMint ~ $ echo "WWW.TECMINT.COM" | tr [:upper:] [:lower:]
www.tecmint.com
tecmint@TecMint ~ $ echo "news.tecmint.com" | tr [:lower:] [:upper:]
NEWS.TECMINT.COM
```

11. more Command

more command is a useful file perusal filter created basically for certificate viewing. It shows file content in a page like format, where users can press **[Enter]** to view more information.

You can use it to view large files like so:

```
tecmint@TecMint ~ $ dmesg | more
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuart
[    0.000000] Linux version 4.4.0-21-generic (buildd@lgw01-21) (gcc version
5.3.1 20160413 (Ubuntu 5.3.1-14ubuntu2) ) #37-Ubuntu SMP Mon Apr 18 18:33:37
UTC 2016 (Ubuntu 4.4.0-21.37-generic
4.4.6)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.4.0-21-generic
root=UUID=bb29dda3-bdaa-4b39-86cf-4a6dc9634a1b ro quiet splash vt.handoff=7
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Centaur CentaurHauls
[    0.000000] x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
```

```

[    0.000000] x86/fpu: Supporting XSAVE feature 0x01: 'x87 floating point
registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x04: 'AVX registers'
[    0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832
bytes, using 'standard' format.
[    0.000000] x86/fpu: Using 'eager' FPU context switches.
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000009d3fff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009d400-0x000000000009ffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000000e0000-0x00000000000fffff]
reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x00000000a56affff] usable
[    0.000000] BIOS-e820: [mem 0x00000000a56b0000-0x00000000a5eaffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000a5eb0000-0x00000000aaabefff] usable
--More--

```

12. less Command

less is the opposite of **more** command above but it offers extra features and it's a little faster with large files.

Use it in the same way as more:

```

tecmint@TecMint ~ $ dmesg | less
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuart
[    0.000000] Linux version 4.4.0-21-generic (buildd@lgw01-21) (gcc version
5.3.1 20160413 (Ubuntu 5.3.1-14ubuntu2) ) #37-Ubuntu SMP Mon Apr 18 18:33:37
UTC 2016 (Ubuntu 4.4.0-21.37-generic
4.4.6)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.4.0-21-generic
root=UUID=bb29dda3-bdaa-4b39-86cf-4a6dc9634a1b ro quiet splash vt.handoff=7
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Centaur CentaurHauls
[    0.000000] x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
[    0.000000] x86/fpu: Supporting XSAVE feature 0x01: 'x87 floating point
registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x04: 'AVX registers'
[    0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832
bytes, using 'standard' format.
[    0.000000] x86/fpu: Using 'eager' FPU context switches.
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000009d3fff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009d400-0x000000000009ffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000000e0000-0x00000000000fffff]
reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x00000000a56affff] usable

```

```
[      0.000000] BIOS-e820: [mem 0x00000000a56b0000-0x00000000a5eaffff]  
reserved  
[      0.000000] BIOS-e820: [mem 0x00000000a5eb0000-0x00000000aaabefff] usable  
:  
:
```

12 Practical Examples of Linux grep Command

by [Rob Krul](#) | Published: November 1, 2013 | Last Updated: December 31, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Have you ever been confronted with the task of looking for a particular string or pattern in a file, yet have no idea where to start looking? Well then, here is **grep** to the rescue!



12 Grep Command Examples

grep is a powerful file pattern searcher that comes equipped on every distribution of **Linux**. If, for whatever reason, it is not installed on your system, you can easily install it via your package manager (**apt-get** on **Debian/Ubuntu** and **yum** on **RHEL/CentOS/Fedora**).

```
$ sudo apt-get install grep          #Debian/Ubuntu  
$ sudo yum install grep            #RHEL/CentOS/Fedora
```

I have found that the easiest way to get your feet wet with **grep** is to just dive right in and use some real world examples.

1. Search and Find Files

Let's say that you have just installed a fresh copy of the new **Ubuntu** on your machine, and that you are going to give **Python** scripting a shot. You have been scouring the web looking for tutorials, but you see that there are two different versions of **Python** in use, and you don't know

which one was installed on your system by the **Ubuntu** installer, or if it installed any modules. Simply run this command:

```
# dpkg -l | grep -i python
```

Sample Output

```
ii  python2.7          2.7.3-0ubuntu3.4
Interactive high-level object-oriented language (version 2.7)
ii  python2.7-minimal   2.7.3-0ubuntu3.4
Minimal subset of the Python language (version 2.7)
ii  python-openssl      0.12-1ubuntu2.1
Python wrapper around the OpenSSL library
ii  python-pam           0.4.2-12.2ubuntu4
Python interface to the PAM library
A
```

First, we ran **dpkg -l**, which lists installed ***.deb** packages on your system. Second, we piped that output to **grep -i** python, which simple states “go to grep and filter out and return everything with ‘python’ in it.” The **-i** option is there to ignore-case, as **grep** is case-sensitive. Using the **-i** option is a good habit of getting into, unless of course you are trying to nail down a more specific search.

2. Search and Filter Files

The **grep** can also be used to search and filter within individual files or multiple files. Lets take this scenario:

You are having some trouble with your **Apache Web Server**, and you have reached out to one of the many awesome forums on the net asking for some help. The kind soul who replies to you has asked you to post the contents of your **/etc/apache2/sites-available/default-ssl** file. Wouldn’t it be easier for you, the guy helping you, and everyone reading it, if you could remove all of the commented lines? Well you can! Just run this:

```
# grep -v "#" /etc/apache2/sites-available/default-ssl
```

The **-v** option tells **grep** to invert its output, meaning that instead of printing matching lines, do the opposite and print all of the lines that don’t match the expression, in this case, the # commented lines.

3. Find all .mp3 Files Only

The **grep** can be very useful for filtering from **stdout**. For example, let’s say that you have an entire folder full of music files in a bunch of different formats. You want to find all of the ***.mp3** files from the artist **JayZ**, but you don’t want any of the remixed tracks. Using a **find command** with a couple of **grep** pipes will do the trick:

```
# find . -name "*.mp3" | grep -i JayZ | grep -vi "remix"
```

In this example, we are using find to print all of the files with a ***.mp3 extension**, piping it to **grep -i** to filter out and prints all files with the name “**JayZ**” and then another pipe to **grep -vi** which filters out and does not print all filenames with the string (in any case) “**remix**”.

Suggested Read: [35 Practical Examples of Linux Find Command](#)

4. Display Number of Lines Before or After Search String

Another couple of options are the **-A** and **-B** switches, which displays the matched line and number of lines either that come before or after the search string. While the man page gives a more detailed explanation, I find it easiest to remember the options as **-A = after**, and **-B = before**:

```
# ifconfig | grep -A 4 eth0
# ifconfig | grep -B 2 UP
```

5. Prints Number of Lines Around Match

The grep’s **-C** option is similar, but instead of printing the lines that come either before or after the string, it prints the lines in either direction:

```
# ifconfig | grep -C 2 lo
```

6. Count Number of Matches

Similar to piping a **grep** string to word count (**wc** program) grep’s built-in option can perform the same for you:

```
# ifconfig | grep -c inet6
```

7. Search Files by Given String

The **-n** option for **grep** is very useful when debugging files during compile errors. It displays the line number in the file of the given search string:

```
# grep -n "main" setup..py
```

8. Search a string Recursively in all Directories

If you would like to search for a string in the current directory along with all of the subdirectories, you can specify the **-r** option to search recursively:

```
# grep -r "function" *
```

9. Searches for the entire pattern

Passing the **-w** option to grep searches for the entire pattern that is in the string. For example, using:

```
# ifconfig | grep -w "RUNNING"
```

Will print out the line containing the pattern in quotes. On the other hand, if you try:

```
# ifconfig | grep -w "RUN"
```

Nothing will be returned as we are not searching for a pattern, but an entire word.

10. Search a string in Gzipped Files

Deserving some mention are grep's derivatives. The first is **zgrep**, which, similar to **zcat**, is for use on **gzipped** files. It takes the same options as **grep** and is used in the same way:

```
# zgrep -i error /var/log/syslog.2.gz
```

11. Match Regular Expression in Files

The **egrep** is another derivative that stands for “**Extended Global Regular Expression**”. It recognizes additional expression meta-characters such **at + ? | ()**.

Suggested Read: What's Difference Between Grep, Egrep and Fgrep in Linux?

egrep is very useful for searching source files, and other pieces of code, should the need arise. It can be invoked from regular grep by specifying the **-E** option.

```
# grep -E
```

12. Search a Fixed Pattern String

The **fgrep** searches a file or list of files for a fixed pattern string. It is the same as **grep -F**. A common way of using **fgrep** is to pass a file of patterns to it:

```
# fgrep -f file_full_of_patterns.txt file_to_search.txt
```

This is just a starting point with **grep**, but as you are probably able to see, it is invaluable for a variety of purposes. Aside from the simple one line commands we have implemented, **grep** can be used to write powerful **cron** jobs, and robust **shell scripts**, for a start.

Suggested Read: 11 ‘Grep’ Commands on Character Classes and Bracket Expressions

Be creative, experiment with the options in the **man page**, and come up with **grep expressions** that serve your own purposes!

11 Advanced Linux ‘Grep’ Commands on Character Classes and Bracket Expressions

by [Jalpan Trivedi](#) | Published: August 30, 2016 | Last Updated: August 31, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Have you ever been into a situation where you need to [search for a string, word or pattern](#) inside a file? if yes, then the **grep** utility comes handy in such situation.

grep is a command line utility for searching plain-text data for lines which matching a regular expression. If you will divide the word **grep** like **g/re/p** then the meaning of **grep** is (globally search a regular expression and print) which search pattern from the file and print the line on the screen i.e. standard output.

Suggested Read: [12 Basic Practical Examples of Linux grep Command](#)

In this article I will be going to explain advanced commands on **grep** for the **Character Classes** in Linux and Unix like operating system.

Here I have considered **tecmint.txt** is the base file where we will search pattern with the help of **grep** command in this article for explanation.

1. Search Alphanumeric Characters

If you have thousands of lines in a file and wanted to search a line which will start from only **A-Z, a-z & 0-9 (Alphanumeric Characters)**.

```
$ grep "^[[:alnum:]]" tecmint.txt
tecmint@tecmint:~$ grep "^[[:alnum:]]" tecmint.txt
TecMint.com is a website that publishes practical
and useful out-of-the-box articles for aspirant like you and me.
And resources that the modern web professional will appreciate.
2012 August
fTecMint was started on 15th August 2012 by technical professionals and
all the articles and contents are written by talented professionals around the globe
keeping in high importance on quality, comprehensiveness, and
usefulness goes into each of the articles published.
gWe, as a team want to share our IT skills and experience through our website which
may assist to formulate a task easy.
too ambiguous or just imprecise. There are many excellent articles on specific topics,
2015
2016
tecmint@tecmint:~$ █
```

Grep – Search Alphanumeric Characters in File

2. Search Alpha Characters

Similar options like if you want to search line which will start from only [A-Z & a-z] i.e. Alpha Characters.

```
$ grep "^[[:alpha:]]" tecmint.txt
tecmint@tecmint:~$ grep "^[[:alpha:]]" tecmint.txt
TecMint.com is a website that publishes practical
and useful out-of-the-box articles for aspirant like you and me.
And resources that the modern web professional will appreciate.
fTecMint was started on 15th August 2012 by technical professionals and
all the articles and contents are written by talented professionals around the globe
keeping in high importance on quality, comprehensiveness, and
usefulness goes into each of the articles published.
gWe, as a team want to share our IT skills and experience through our website which
may assist to formulate a task easy.
too ambiguous or just imprecise. There are many excellent articles on specific topic
s,
```

Grep – Search Alpha Characters in File

3. Search Blank Characters

Another options like if you want to search line which will start from [Tab & Space] i.e. **Blank Characters**.

```
$ grep "^[[:blank:]]" tecmint.txt
tecmint@tecmint:~$ grep "^[[:blank:]]" tecmint.txt
      You are Welcome to join our community and can be part of our team.
      To send your articles for review,
```

Grep – Search for Spaces or Tabs in File

4. Search Digit Characters

The digit option for grep is also very useful to search line which will start from digit [0-9] i.e. **Digit Characters**.

```
$ grep "^[[:digit:]]" tecmint.txt
tecmint@tecmint:~$ grep "^[[:digit:]]" tecmint.txt
2012 August
2015
2016
```

Grep – Search Number Characters in File

5. Search Lower Letters

Another option for grep is to search line which will start from lower letters i.e [a-z] (**Lower Letters**).

```
$ grep "^[[:lower:]]" tecmint.txt
tecmint@tecmint:~$ grep "^[[:lower:]]" tecmint.txt
and useful out-of-the-box articles for aspirant like you and me.
fTecMint was started on 15th August 2012 by technical professionals and
all the articles and contents are written by talented professionals around the globe
,
keeping in high importance on quality, comprehensiveness, and
usefulness goes into each of the articles published.
gWe, as a team want to share our IT skills and experience through our website which
may assist to formulate a task easy.
too ambiguous or just imprecise. There are many excellent articles on specific topic
s,
```

Grep – Search Lower Letters or Words in File

6. Search Punctuation Characters

The Punctuation characters for grep is to search line which will start from [! " # \$ % & ‘ () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~ .] i.e. **Punctuation Characters**.

```
$ grep "^[[:punct:]]" tecmint.txt
tecmint@tecmint:~$ grep "^[[:punct:]]" tecmint.txt
#We seek to present exceptional, remarkable tips, tutorials,
#It has been seen generally we found a majority of Linux related resources on the we
b to be either too detailed,
^but they were usually part of a general interest publication, and information on r
elated topics on the same site is sometimes hard to find.
#contributing and submitting well written article on Linux.
^please contact us by email tecmint.com [at] gmail [dot] com.
```

Grep – Search Punctuation Characters in File

7. Search Graphical Characters

The grep is also used to search a line which will start from **Alphanumeric & Punctuation Characters** called as **Graphical Characters**.

```
$ grep "^[[:graph:]]" tecmint.txt
```

```
tecmint@tecmint:~$ grep "^\[:graph:\]" tecmint.txt
TecMint.com is a website that publishes practical
and useful out-of-the-box articles for aspirant like you and me.
%We seek to present exceptional, remarkable tips, tutorials,
And resources that the modern web professional will appreciate.
2012 August
fTecMint was started on 15th August 2012 by technical professionals and
all the articles and contents are written by talented professionals around the globe
, keeping in high importance on quality, comprehensiveness, and
usefulness goes into each of the articles published.
gWe, as a team want to share our IT skills and experience through our website which
may assist to formulate a task easy.
#It has been seen generally we found a majority of Linux related resources on the we
b to be either too detailed,
too ambiguous or just imprecise. There are many excellent articles on specific topic
s,
^Fbut they were usually part of a general interest publication, and information on r
elated topics on the same site is sometimes hard to find.
2015
2016
#contributing and submitting well written article on Linux.
^Bplease contact us by email tecmint.com [at] gmail [dot] com.
```

Grep – Search Graphical Characters in File

8. Search Printable Characters

Similarly like **Graphical Characters**, grep is useful to search a line which will start from Alphanumeric, Punctuation and space characters.

```
$ grep "^\[:print:\]" tecmint.txt
tecmint@tecmint:~$ grep "^\[:print:\]" tecmint.txt
TecMint.com is a website that publishes practical
and useful out-of-the-box articles for aspirant like you and me.
%We seek to present exceptional, remarkable tips, tutorials,
And resources that the modern web professional will appreciate.
2012 August
fTecMint was started on 15th August 2012 by technical professionals and
all the articles and contents are written by talented professionals around the globe
, keeping in high importance on quality, comprehensiveness, and
usefulness goes into each of the articles published.
gWe, as a team want to share our IT skills and experience through our website which
may assist to formulate a task easy.
#It has been seen generally we found a majority of Linux related resources on the we
b to be either too detailed,
too ambiguous or just imprecise. There are many excellent articles on specific topic
s,
^Fbut they were usually part of a general interest publication, and information on r
elated topics on the same site is sometimes hard to find.
2015
2016
#contributing and submitting well written article on Linux.
To send your articles for review,
^Bplease contact us by email tecmint.com [at] gmail [dot] com.
```

Grep – Search Printable Characters in File

9. Search Space Characters

The grep has also a functionality to search a line which will start from [**tab**, **newline**, **vertical tab**, **form feed**, **carriage return**, and **space**] i.e. **Space Characters**.

```
$ grep "^\[:space:]" tecmint.txt
tecmint@tecmint:~$ grep "^\[:space:]" tecmint.txt
      You are Welcome to join our community and can be part of our team.
      To send your articles for review,
```

Grep – Search Space Characters in File

10. Search Uppercase Letters

Another option in the grep is also used to search a line which will start from [A-Z] i.e **Uppercase Letters**.

```
$ grep "^\[:upper:]" tecmint.txt
tecmint@tecmint:~$ grep "^\[:upper:]" tecmint.txt
TecMint.com is a website that publishes practical
And resources that the modern web professional will appreciate.
```

Grep – Search Uppercase Letters in File

11. Search Hexadecimal Digits

The grep searches a line which will start from [0-9, A-F and a-f] i.e **Hexadecimal Digits**.

```
$ grep "^\[:xdigit:]" tecmint.txt
tecmint@tecmint:~$ grep "^\[:xdigit:]" tecmint.txt
and useful out-of-the-box articles for aspirant like you and me.
And resources that the modern web professional will appreciate.
2012 August
fTecMint was started on 15th August 2012 by technical professionals and
all the articles and contents are written by talented professionals around the globe
,
2015
2016
```

Grep – Search Hexadecimal Digits in File

I have explained the advanced functionality of **grep** which is very strong and powerful tool to search the pattern in a File. Grep is also an important tool for shell scripting and programmers to search the pattern in the programs. It is worth to be familiar with other options and syntax to save the time.

Suggested Read: What's Difference Between Grep, Egrep and Fgrep in Linux?

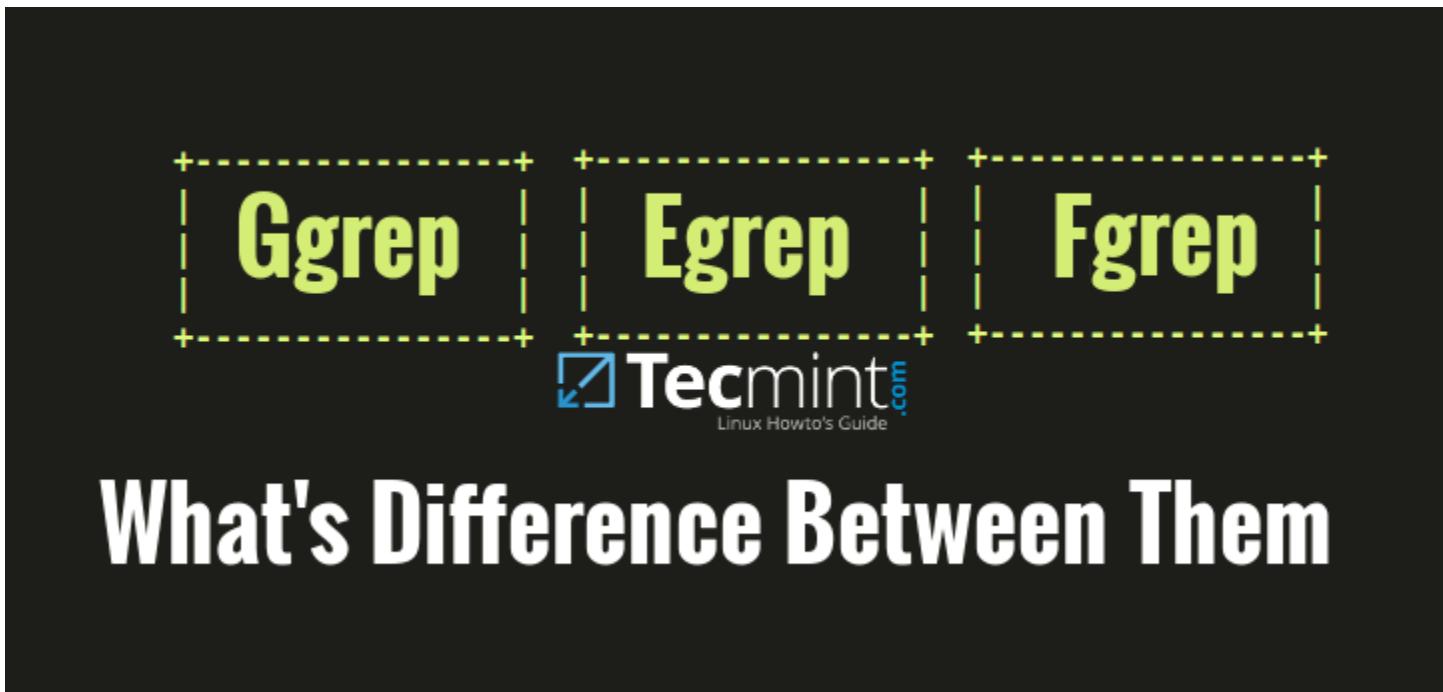
In case any issues on the commands which is explained in the article, you can post your comment in the comment section below.

What's Difference Between Grep, Egrep and Fgrep in Linux?

by [Gunjit Khera](#) | Published: February 15, 2016 | Last Updated: August 30, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

One of the renowned search tool on Unix-like systems which can be used to search for anything whether it be a file, or a line or multiple lines in file is grep utility. It is very vast in functionality which can be attributed to the large number of options it supports like: searching using string pattern, or reg-ex pattern or perl based reg-ex etc.



What's Difference Between Them

Difference Between grep, egrep and fgrep in Linux

Due its varying functionalities, it has many variants including **grep**, **egrep (Extended GREP)**, **fgrep (Fixed GREP)**, **pgrep (Process GREP)**, **rgrep (Recursive GREP)** etc. But these variants have minor differences to original **grep** which has made them popular and to be used by various Linux programmers for specific tasks.

Main thing that remains to be investigated is what are the differences between the three main variants i.e. '**grep**', '**egrep**' and '**fgrep**' of grep that makes Linux users choose one or the other version as per requirement.

Some Special Meta-Characters of grep

1. + – Equivalent to one or more occurrences of previous character.
2. ? – This denotes almost 1 repetition of previous character. Like: a? Would match ‘a’ or ‘aa’.
3. (– Start of alternation expression.
4.) – End of alternation expression.
5. | – Matching either of the expression separated by ‘|’. Like: “(a | b) cde” would match either ‘abcde’ or ‘bbcde’.
6. { – This meta-character indicates start of range specifier. Like: “a{2}” matches “aa” in file i.e. a 2 times.
7. } – This meta-character indicates end of range specifier.

Differences Between grep, egrep and fgrep

Some main differences between **grep**, **egrep** and **fgrep** can be highlighted as follows. For this set of examples we are assuming the file on which operation is being performed to be:

```
grep is a command that can be used on unix-like systems.  
it searches for any string in list of strings or file.  
It is very fast.  
(f|g)ile
```

```
□  
~  
~  
~  
~  
-- INSERT --
```

9,1

Linux grep Command

Grep Command

grep or **Global Regular Expression Print** is the main search program on Unix-like systems which can search for any type of string on any file or list of files or even output of any command.

Suggested Read: [12 Practical Examples of Linux grep Command](#)

It uses **Basic Regular Expressions** apart from normal strings as a search pattern. In Basic Regular Expressions (BRE), meta-characters like: '{', '}', '(', ')', '|', '+', '?' loose their meaning and are treated as normal characters of string and need to be escaped if they are to be treated as special characters.

Suggested Read: [11 Advance ‘Grep’ Commands on Character Classes and Bracket Expressions](#)

Also, grep uses Boyer-Moore algorithm for fast searching any string or regular expression.

```
$ grep -C 0 '(f|g)ile' check_file
$ grep -C 0 '\(f\|g\)\ile' check file
tecmint@tecmint ~ $ grep -C 0 '(f|g)ile' check_file
(f|g)ile
tecmint@tecmint ~ $ grep -C 0 '\(f\|g\)\ile' check_file
it searches for any string in list of strings or file.
tecmint@tecmint ~ $ 
```

Linux grep Command Example

Like here, when the command is run without escaping '()' and '|' then it searched for the complete string i.e. "(f|g)ile" in the file. But when the special characters were escaped, then instead of treating them as part of string, grep treated them as meta-characters and searched for words "file" or "gile" in the file.

Egrep Command

Egrep or **grep -E** is another version of grep or the Extended grep. This version of grep is efficient and fast when it comes to searching for a regular expression pattern as it treats meta-characters as is and doesn't substitute them as strings like in grep, and hence you are freed from the burden of escaping them as in grep. It uses ERE or the Extended Regular Expression set.

In case of egrep, even if you do not escape the meta-characters, it would treat them as special characters and substitute them for their special meaning instead of treating them as part of string.

```
$ egrep -C 0 '(f|g)ile' check_file
$ egrep -C 0 '\(f\|g\)\ile' check file
tecmint@tecmint ~ $ egrep -C 0 '(f|g)ile' check_file
it searches for any string in list of strings or file.
tecmint@tecmint ~ $ egrep -C 0 '\(f\|g\)\ile' check_file
(f|g)ile
tecmint@tecmint ~ $ 
```

Linux egrep Command Examples

Like here, **egrep** searched for “file” string when the meta-characters were not escaped as it would mean by the meaning of these characters. But, when these characters were escaped, then egrep treated them as part of string and searched for complete string “(f|g)ile” in the file.

fgrep Command

Fgrep or the **Fixed grep** or **grep -F** is yet another version of grep which is fast in searching when it comes to search for the entire string instead of regular expression as it doesn’t recognize the regular expressions, neither any meta-characters. For searching any direct string, this is the version of grep which should be selected.

Fgrep searches for complete string and doesn’t even recognize special characters as part of regular expression even if escaped or not escaped.

```
$ fgrep -C 0 '(f|g)ile' check_file
$ fgrep -C 0 '\(f\|g\)\ile' check_file
tecmint@tecmint ~ $ fgrep -C 0 '(f|g)ile' check_file
(f|g)ile
tecmint@tecmint ~ $ fgrep -C 0 '\(f\|g\)\ile' check_file
tecmint@tecmint ~ $ 
```

Linux fgrep Command Examples

Like, when meta-characters were not escaped, fgrep searched for the complete string “(f|g)ile” in the file, and when the meta-characters were escaped, then the fgrep command searched for “\ (f\|g\)\ile” all characters as is in the file.

We’ve already covered some practical examples of grep command you can read them here, if you want to get more out of grep command in Linux.

[Learn 12 Practical Examples of Linux grep Command](#)

Conclusion

Above highlighted are the differences between ‘grep’, ‘egrep’ and ‘fgrep’. Apart from difference in the set of regular expressions used, and speed of execution, rest command line parameters remain same for all the three versions of grep and even instead of “egrep” or “fgrep”, “grep -E” or “grep -F” are recommended to be used.

If you find any other differences between these three versions of grep, do mention them in your comments

Understanding Different Classifications of Shell Commands and Their Usage in Linux

by [Aaron Kili](#) | Published: August 26, 2016 | Last Updated: August 26, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

When it comes to gaining absolute control over your Linux system, then nothing comes close to the command line interface (CLI). In order to become a Linux power user, one must understand the [different types of shell](#) commands and the appropriate ways of using them from the terminal.

In Linux, there are several types of commands, and for a new Linux user, knowing the meaning of different commands enables for efficient and precise usage. Therefore, in this article, we shall walk through the various classifications of shell commands in Linux.

Suggested Read: [5 Interesting Command Line Tips and Tricks in Linux – Part 1](#)

One important thing to note is that the command line interface is different from the shell, it only provides a means for you to access the shell. The shell, which is also programmable then makes it possible to communicate with the kernel using commands.

Different classifications of Linux commands fall under the following classifications:

1. Program Executables (File System Commands)

When you run a command, Linux searches through the directories stored in the [\\$PATH environmental variable](#) from left to right for the executable of that specific command.

You can view the directories in the \$PATH as follows:

```
$ echo $PATH  
/home/aaronkilik/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:  
/bin:/usr/games:/usr/local/games
```

In the above order, the directory `/home/aaronkilik/bin` will be searched first followed by `/usr/local/sbin` and so on, the order is significant in the search process.

Examples of file system commands in `/usr/bin` directory:

```
$ ll /bin/
```

Sample Output

```
total 16284
```

```

drwxr-xr-x  2 root root    4096 Jul 31 16:30 .
drwxr-xr-x 23 root root    4096 Jul 31 16:29 ..
-rw xr-xr-x  1 root root    6456 Apr 14 18:53 archdetect*
-rw xr-xr-x  1 root root 1037440 May 17 16:15 bash*
-rw xr-xr-x  1 root root   520992 Jan 20 2016 btrfs*
-rw xr-xr-x  1 root root  249464 Jan 20 2016 btrfs-calc-size*
lrwxrwxrwx  1 root root      5 Jul 31 16:19 btrfsck -> btrfs*
-rw xr-xr-x  1 root root 278376 Jan 20 2016 btrfs-convert*
-rw xr-xr-x  1 root root  249464 Jan 20 2016 btrfs-debug-tree*
-rw xr-xr-x  1 root root  245368 Jan 20 2016 btrfs-find-root*
-rw xr-xr-x  1 root root  270136 Jan 20 2016 btrfs-image*
-rw xr-xr-x  1 root root  249464 Jan 20 2016 btrfs-map-logical*
-rw xr-xr-x  1 root root  245368 Jan 20 2016 btrfs-select-super*
-rw xr-xr-x  1 root root  253816 Jan 20 2016 btrfs-show-super*
-rw xr-xr-x  1 root root  249464 Jan 20 2016 btrfs-tune*
-rw xr-xr-x  1 root root  245368 Jan 20 2016 btrfs-zero-log*
-rw xr-xr-x  1 root root   31288 May 20 2015 bunzip2*
-rw xr-xr-x  1 root root 1964536 Aug 19 2015 busybox*
-rw xr-xr-x  1 root root   31288 May 20 2015 bzcat*
lrwxrwxrwx  1 root root      6 Jul 31 16:19 bzcmp -> bzipdiff*
-rw xr-xr-x  1 root root   2140 May 20 2015 bzipdiff*
lrwxrwxrwx  1 root root      6 Jul 31 16:19 bzgrep -> bzgrep*
-rw xr-xr-x  1 root root   4877 May 20 2015 bzexe*
lrwxrwxrwx  1 root root      6 Jul 31 16:19 bzfgrep -> bzgrep*
-rw xr-xr-x  1 root root   3642 May 20 2015 bzgrep*

```

2. Linux Aliases

These are user defined commands, they are created using the **alias** shell built-in command, and contain other shell commands with some options and arguments. The idea is to basically use new and short names for lengthy commands.

Suggested Read: [10 Amazing and Mysterious Uses of \(!\) Symbol or Operator in Linux Commands](#)

The syntax for creating an **alias** is as follows:

```
$ alias newcommand='command -options'
```

To list all **aliases** on your system, issue the command below:

```
$ alias -p
alias alert='notify-send --urgency=low -i "$( [ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\\s*[0-9]+\\\s*//;s/[;\\&]\\s*alert$//\\''")'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

To create a new alias in Linux, go through some below examples.

```
$ alias update='sudo apt update'  
$ alias upgrade='sudo apt dist-upgrade'  
$ alias -p | grep 'up'  
aaronkilik@tecmint ~/bin $ alias update='sudo apt update'  
aaronkilik@tecmint ~/bin $  
aaronkilik@tecmint ~/bin $ alias upgrade='sudo apt dist-upgrade'  
aaronkilik@tecmint ~/bin $  
aaronkilik@tecmint ~/bin $  
aaronkilik@tecmint ~/bin $ alias -p | grep 'up'  
alias update='sudo apt update'  
alias upgrade='sudo apt dist-upgrade'  
aaronkilik@tecmint ~/bin $
```

Create Aliases in Linux

However, the aliases we have created above only work temporarily, when the system is restarted, they will not work after the next boot. You can set permanent aliases in your `.bashrc` file as shown below.

```
# some more ls aliases  
alias ll='ls -alF'  
alias la='ls -A'  
alias l='ls -CF'  
  
# Add an "alert" alias for long running commands. Use like so:  
# sleep 10; alert  
alias alert='notify-send --urgency=low -i "$( [ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''\s/\^\\s*[0-9]\\+\s*//;s/[;&|]\\s*alert$///'\')"  
  
# Alias definitions.  
# You may want to put all your additions into a separate file like  
# ~/.bash_aliases, instead of adding them here directly.  
# See /usr/share/doc/bash-doc/examples in the bash-doc package.
```

105,1

83%

Set Aliases Permanent in Linux

After adding them, run the command below to active.

```
$ source ~/.bashrc
```

3. Linux Shell Reserved Words

In shell programming, words such as **if**, **then**, **fi**, **for**, **while**, **case**, **esac**, **else**, **until** and many others are shell reserved words. As the description implies, they have specialized meaning to the shell.

You can list out all Linux shell keywords using `type` command as shown:

```
$ type if then fi for while case esac else until
if is a shell keyword
then is a shell keyword
fi is a shell keyword
for is a shell keyword
while is a shell keyword
case is a shell keyword
esac is a shell keyword
else is a shell keyword
until is a shell keyword
```

Suggested Read: [10 Useful Linux Chaining Operators with Practical Examples](#)

4. Linux Shell Functions

A shell function is a group of commands that are executed collectively within the current shell. Functions help to carry out a specific task in a shell script. The conventional form of writing shell functions in a script is:

```
function_name() {
command1
command2
.....
}
```

Alternatively,

```
function function_name {
command1
command2
.....
}
```

Let's take a look at how to write shell functions in a script named `shell_functions.sh`.

```
#!/bin/bash
#write a shell function to update and upgrade installed packages
upgrade_system(){
sudo apt update;
sudo apt dist-upgrade;
}
#execute function
upgrade_system
```

Instead of executing the two commands: `sudo apt update` and `sudo apt dist-upgrade` from the command line, we have written a simple shell function to execute the two commands as a single command, `upgrade_system` within a script.

Suggested Read: [5 Shell Scripts for Linux Newbies to Learn Shell Programming](#)

Save the file and thereafter, make the script executable. Finally run it as below:

```
$ chmod +x shell_functions.sh
$ ./shell_functions.sh
aaronkilik@tecmint ~/bin $ chmod +x shell_functions.sh
aaronkilik@tecmint ~/bin $ ./shell_functions.sh
Hit:1 http://ppa.launchpad.net/maarten-baert/simplescreenrecorder/ubuntu xenial InRelease
Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:3 http://security.ubuntu.com/ubuntu xenial-security InRelease [94.5 kB]
Ign:4 http://packages.linuxmint.com sarah InRelease
Hit:5 http://archive.canonical.com/ubuntu xenial InRelease
Get:6 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [95.7 kB]
Hit:7 http://packages.linuxmint.com sarah Release
Hit:9 http://archive.ubuntu.com/ubuntu xenial-backports InRelease
Get:10 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [134 kB]
Get:11 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [376 kB]
Reading package lists... Done [37%] [10 Packages 97.6 kB/134 kB 73%] ^C
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded: →
  apparmor base-files fontconfig fontconfig-config kpartx kpartx-boot
  libapparmor-perl libapparmor1 libfontconfig1:i386 libfontconfig1
  mintbackup mintupload nvidia-prime-applet
13 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,245 kB of archives.
After this operation, 8,192 B of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Linux Shell Functions Script

5. Linux Shell Built-in Commands

These are Linux commands that built into the shell, thus you cannot find them within the file system. They include **pwd**, **cd**, **bg**, **alias**, **history**, **type**, **source**, **read**, **exit** and many others.

You can list or check Linux built-in commands using **type** command as shown:

```
$ type pwd
pwd is a shell builtin
$ type cd
cd is a shell builtin
$ type bg
bg is a shell builtin
$ type alias
alias is a shell builtin
$ type history
history is a shell builtin
```

Learn about some Linux built-in Commands usage:

1. [15 ‘pwd’ Command Examples in Linux](#)
2. [15 ‘cd’ Command Examples in Linux](#)
3. [Learn The Power of Linux ‘history’ Command](#)

Conclusion

As a Linux user, it is always important to know the type of command you are running. I believe, with the precise and simple-to-understand explanation above including a few relevant illustrations, you probably have a good understanding of the [various categories of Linux commands](#).

You can as well get in touch through the comment section below for any questions or supplementary ideas that you would like to offer us.

10 Amazing and Mysterious Uses of (!) Symbol or Operator in Linux Commands

by [Editor](#) | Published: May 15, 2015 | Last Updated: May 18, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

The '!' symbol or operator in Linux can be used as **Logical Negation** operator as well as to fetch commands from history with tweaks or to run previously run command with modification. All the commands below have been checked explicitly in bash Shell. Though I have not checked but a major of these won't run in other shell. Here we go into the amazing and mysterious uses of '!' symbol or operator in Linux commands.

1. Run a command from history by command number.

You might not be aware of the fact that you can run a command from your **history** command (already/earlier executed commands). To get started first find the command number by running '**history**' command.

```
$ history
```

File Edit View Search Terminal Help

avi@deb:~\$ █

Now run a command from **history** just by the number at which it appears, in the output of **history**. Say run a command that appears at number **1551** in the output of ‘**history**‘ command.

\$!1551

File Edit View Search Terminal Help

avi@deb:~\$ █

And, it runs the command ([top command](#) in the above case), that was listed at number **1551**. This way to retrieving already executed command is very helpful specially in case of those commands which are long. You just need to call it using **![Number at which it appears in the output of history command]**.

2. Run previously executed command as 2nd last command, 7th last command,etc.

You may run those commands which you have run previously by their running sequence being the last run command will be represented as **-1**, second last as **-2**, seventh last as **-7**,....

First run **history** command to get a list of last executed command. It is necessary to run **history** command, so that you can be sure that there is no command like `rm command > file` and others just to make sure you do not run any dangerous command accidentally. And then check Sixth last command, Eight last command and Tenth last command.

```
$ history
$ !-6
$ !-8
$ !-10
```

```
File Edit View Search Terminal Help
```

```
avi@deb:/home$
```

Run Last Executed Commands By Numbers

3. Pass arguments of last command that we run to the new command without retyping

I need to list the content of directory '**/home/\$USER/Binary/firefox**' so I fired.

```
$ ls /home/$USER/Binary/firefox
```

Then I realized that I should have fired '**ls -l**' to see which file is executable there? So should I type the whole command again! No I don't need. I just need to carry the last argument to this new command as:

```
$ ls -l !$
```

Here **!\$** will carry arguments passed in last command to this new command.

```
File Edit View Search Terminal Help
```

```
avi@deb:/home$ █
```

Pass Arguments of Last Executed Command to New

4. How to handle two or more arguments using (!)

Let's say I created a text file **1.txt** on the Desktop.

```
$ touch /home/avi/Desktop/1.txt
```

and then copy it to '**/home/avi/Downloads**' using complete path on either side with **cp** command.

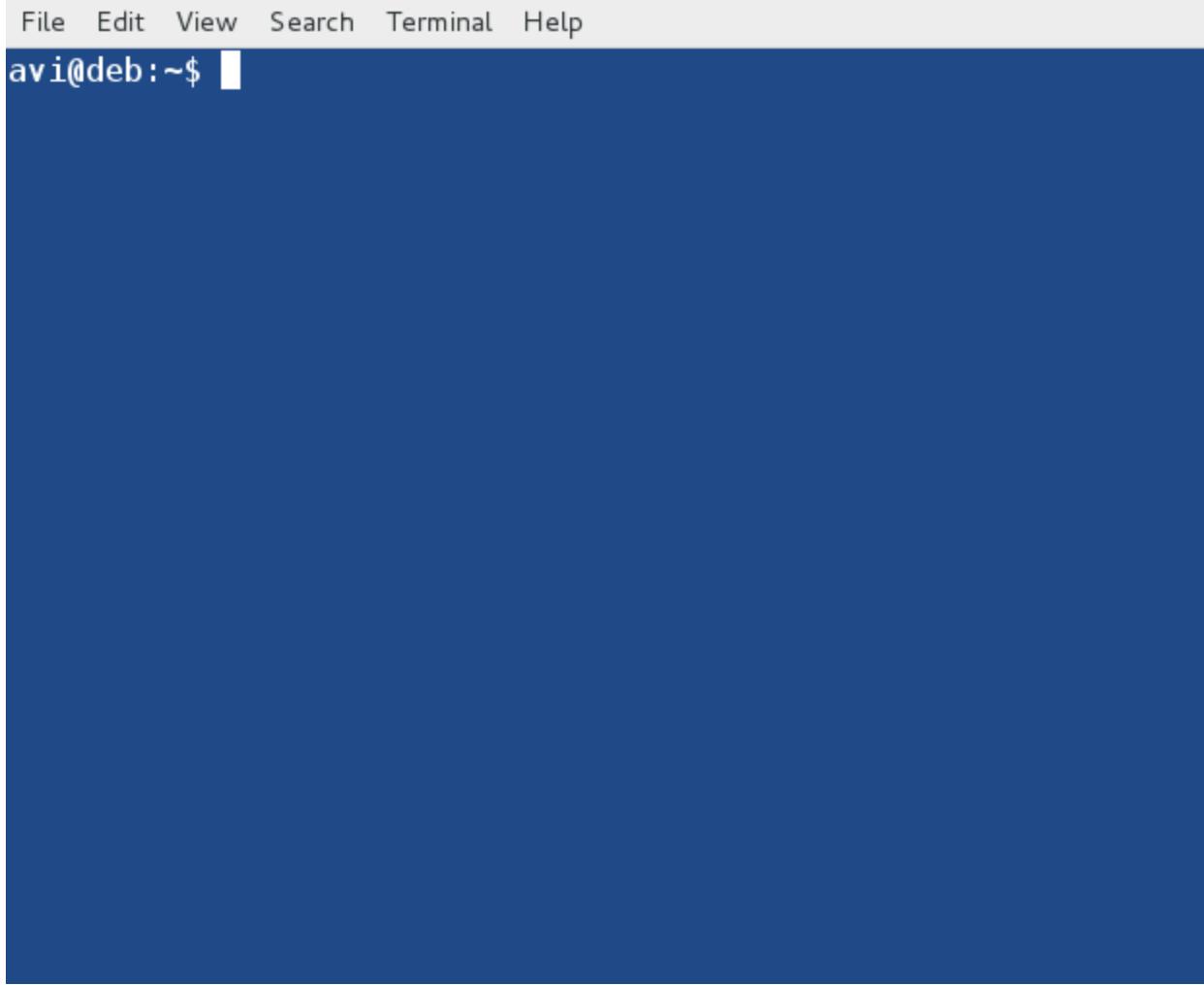
```
$ cp /home/avi/Desktop/1.txt /home/avi/downloads
```

Now we have passed two arguments with **cp** command. First is '**/home/avi/Desktop/1.txt**' and second is '**/home/avi/Downloads**', lets handle them differently, just execute `echo [arguments]` to print both arguments differently.

```
$ echo "1st Argument is : !^"  
$ echo "2nd Argument is : !cp:2"
```

Note 1st argument can be printed as “!^” and rest of the arguments can be printed by executing “! [Name_of_Command] : [Number_of_argument]”.

In the above example the first command was ‘cp’ and 2nd argument was needed to print. Hence “! cp:2”, if any command say xyz is run with 5 arguments and you need to get 4th argument, you may use “!xyz:4”, and use it as you like. All the arguments can be accessed by “!*”.



```
File Edit View Search Terminal Help
avi@deb:~$
```

Handle Two or More Arguments

5. Execute last command on the basis of keywords

We can execute the last executed command on the basis of keywords. We can understand it as follows:

```
$ ls /home > /dev/null [Command 1]
$ ls -l /home/avi/Desktop > /dev/null [Command 2]
$ ls -la /home/avi/Downloads > /dev/null [Command 3]
$ ls -lA /usr/bin > /dev/null [Command 4]
```

Here we have used same command (**ls**) but with different switches and for different folders. Moreover we have sent to output of each command to '**/dev/null**' as we are not going to deal with the output of the command also the console remains clean.

Now Execute last run command on the basis of keywords.

```
$ ! ls [Command 1]
$ ! ls -l [Command 2]
$ ! ls -la [Command 3]
$ ! ls -lA [Command 4]
```

Check the output and you will be astonished that you are running already executed commands just by **ls** keywords.



The screenshot shows a terminal window with a dark blue background and a light blue header bar. The header bar contains the following menu items: File, Edit, View, Search, Terminal, and Help. Below the header bar, the terminal prompt is visible: **avi@deb:~\$**. The main area of the terminal is mostly blank, indicating that no new commands have been entered since the history was captured.

Run Commands Based on Keywords

6. The power of !! Operator

You can run/alter your last run command using (!!). It will call the last run command with alter/tweak in the current command. Lets show you the scenario

Last day I run a one-liner script to get my private IP so I run,

```
$ ip addr show | grep inet | grep -v 'inet6' | grep -v '127.0.0.1' | awk  
'{print $2}' | cut -f1 -d/
```

Then suddenly I figured out that I need to redirect the output of the above script to a file **ip.txt**, so what should I do? Should I retype the whole command again and redirect the output to a file? Well an easy solution is to use **UP** navigation key and add '`> ip.txt`' to redirect the output to a file as.

```
$ ip addr show | grep inet | grep -v 'inet6' | grep -v '127.0.0.1' | awk  
'{print $2}' | cut -f1 -d/ > ip.txt
```

Thanks to the life Savior **UP** navigation key here. Now consider the below condition, the next time I run below one-liner script.

```
$ ifconfig | grep "inet addr:" | awk '{print $2}' | grep -v '127.0.0.1' | cut  
-f2 -d:
```

As soon as I run script, the bash prompt returned an error with the message "bash: ifconfig: command not found", It was not difficult for me to guess I run this command as user where it should be run as root.

So what's the solution? It is difficult to login to root and then type the whole command again! Also (**UP Navigation Key**) in last example didn't came to rescue here. So? We need to call "`!!`" without quotes, which will call the last command for that user.

```
$ su -c "!!" root
```

Here **su** is switch user which is root, `-c` is to run the specific command as the user and the most important part `!!` will be replaced by command and last run command will be substituted here. Yeah! You need to provide root password.

```
File Edit View Search Terminal Help  
avi@deb:~$ ifconfig | grep "inet addr:" | awk '{print $2}' | grep -v '127.0.0.1'| cut -f2 -d:■
```

The Power of !! Key

I make use of !! mostly in following scenarios,

1. When I run **apt-get** command as normal user, I usually get an error saying you don't have permission to execute.

```
$ apt-get upgrade && apt-get dist-upgrade
```

Opps error...don't worry execute below command to get it successful..

```
$ su -c !!
```

Same way I do for,

```
$ service apache2 start  
or  
$ /etc/init.d/apache2 start  
or
```

```
$ systemctl start apache2
```

OOPS User not authorized to carry such task, so I run..

```
$ su -c 'service apache2 start'  
or  
$ su -c '/etc/init.d/apache2 start'  
or  
$ su -c 'systemctl start apache2'
```

7. Run a command that affects all the file except ![FILE_NAME]

The **!** (**Logical NOT**) can be used to run the command on all the files/extension except that is behind **'!'**.

A. Remove all the files from a directory except the one the name of which is **2.txt**.

```
$ rm !(2.txt)
```

B. Remove all the file type from the folder except the one the extension of which is '**pdf**'.

```
$ $ rm !(*.pdf)
```

8. Check if a directory (say /home/avi/Tecmint)exist or not? Printf if the said directory exist or not.

Here we will use **'! -d'** to validate if the directory exist or not followed by Logical AND Operator **(&&)** to print that directory does not exist and Logical OR Operator **(||)** to print the directory is present.

Logic is, when the output of **[! -d /home/avi/Tecmint]** is **0**, it will execute what lies beyond Logical **AND** else it will go to Logical **OR** **(||)** and execute what lies beyond Logical **OR**.

```
$ [ ! -d /home/avi/Tecmint ] && printf '\nno such /home/avi/Tecmint directory  
exist\n' || printf '\n/home/avi/Tecmint directory exist\n'
```

9. Check if a directory exist or not? If not exit the command.

Similar to the above condition, but here if the desired directory doesn't exist it will exit the command.

```
$ [ ! -d /home/avi/Tecmint ] && exit
```

10. Create a directory (say test) in your home directory if it does not exist.

A general implementation in Scripting Language where if the desired directory does not exist, it will create one.

```
[ ! -d /home/avi/Tecmint ] && mkdir /home/avi/Tecmint
```

That's all for now. If you know or come across any other use of '!' which is worth knowing, you may like to provide us with your suggestion in the feedback. Keep connected!

15 ‘pwd’ (Print Working Directory) Command Examples in Linux

by [Editor](#) | Published: November 22, 2014 | Last Updated: January 27, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

For those working with Linux command Line, command ‘**pwd**’ is very helpful, which tells where you are – in which directory, starting from the root (/). Specially for Linux newbies, who may get lost amidst of directories in command Line Interface while navigation, command ‘**pwd**’ comes to rescue.



15 pwd Command Examples

What is pwd?

‘**pwd**’ stands for ‘**Print Working Directory**’. As the name states, command ‘**pwd**’ prints the current working directory or simply the directory user is, at present. It prints the current directory name with the complete path starting from root (/). This command is built in shell command and is available on most of the shell – bash, Bourne shell, ksh,zsh, etc.

Basic syntax of pwd:

```
# pwd [OPTION]
```

Options used with pwd

Options	Description
-L (logical)	Use PWD from environment, even if it contains symbolic links
-P (physical)	Avoid all symbolic links
-help	Display this help and exit
-version	Output version information and exit

If both ‘-L’ and ‘-P’ options are used, option ‘L’ is taken into priority. If no option is specified at the prompt, pwd will avoid all symlinks, i.e., take option ‘-P’ into account.

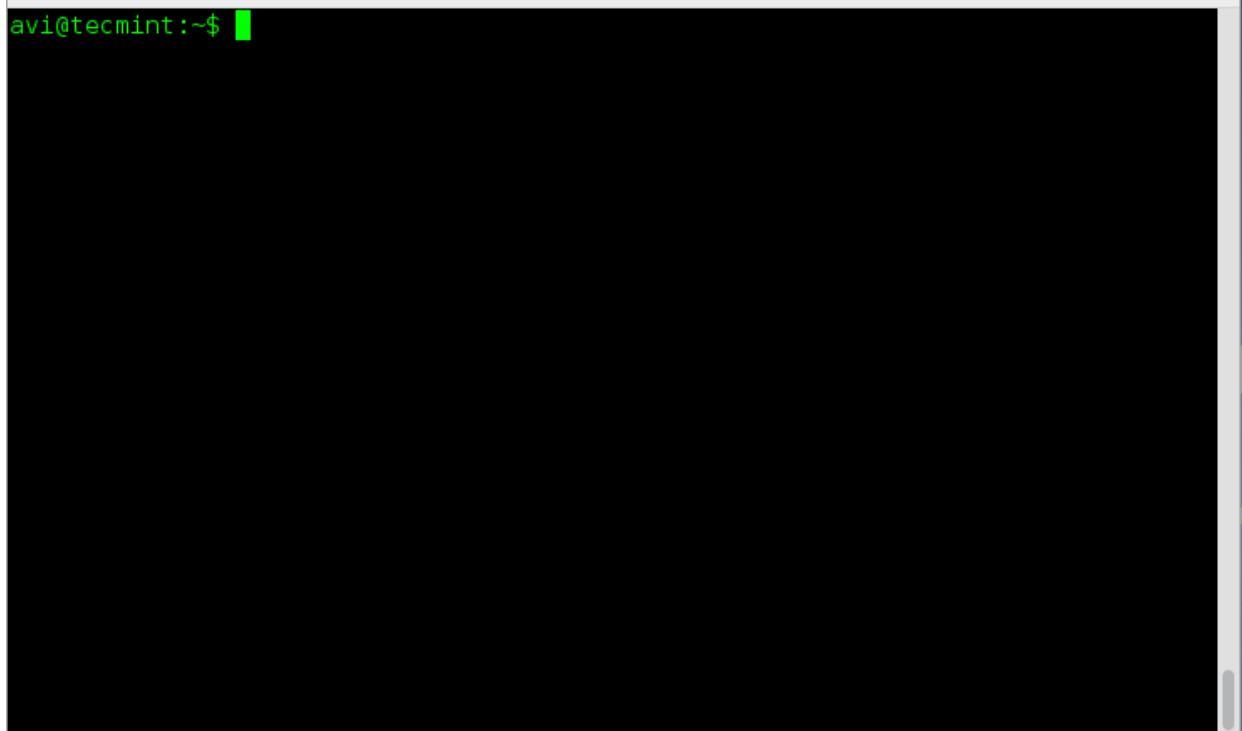
Exit status of command pwd:

0	Success
Non-zero	Failure

This article aims at providing you a deep insight of Linux command ‘pwd‘ with practical examples.

1. Print your current working directory.

```
avi@tecmint:~$ /bin/pwd  
/home/avi
```



The image shows a terminal window with a black background and white text. At the top left, it says 'avi@tecmint:~\$'. Below that, there is a large, mostly blank black area. At the very bottom right corner of the terminal window, there is a small vertical scroll bar.

Print Working Directory

2. Create a symbolic link of a folder (say **/var/www/html** into your home directory as **htm**). Move to the newly created directory and print working directory with symbolic links and without symbolic links.

Create a symbolic link of folder /var/www/html as htm in your home directory and move to it.

```
avi@tecmint:~$ ln -s /var/www/html/ htm  
avi@tecmint:~$ cd htm  
avi@tecmint:~$
```



Create Symbolic Link

3. Print working directory from environment even if it contains symlinks.

```
avi@tecmint:~$ /bin/pwd -L  
/home/avi/htm
```

```
avi@tecmint:~/htm$ █
```

Print Current Working Directory

4. Print actual physical current working directory by resolving all symbolic links.

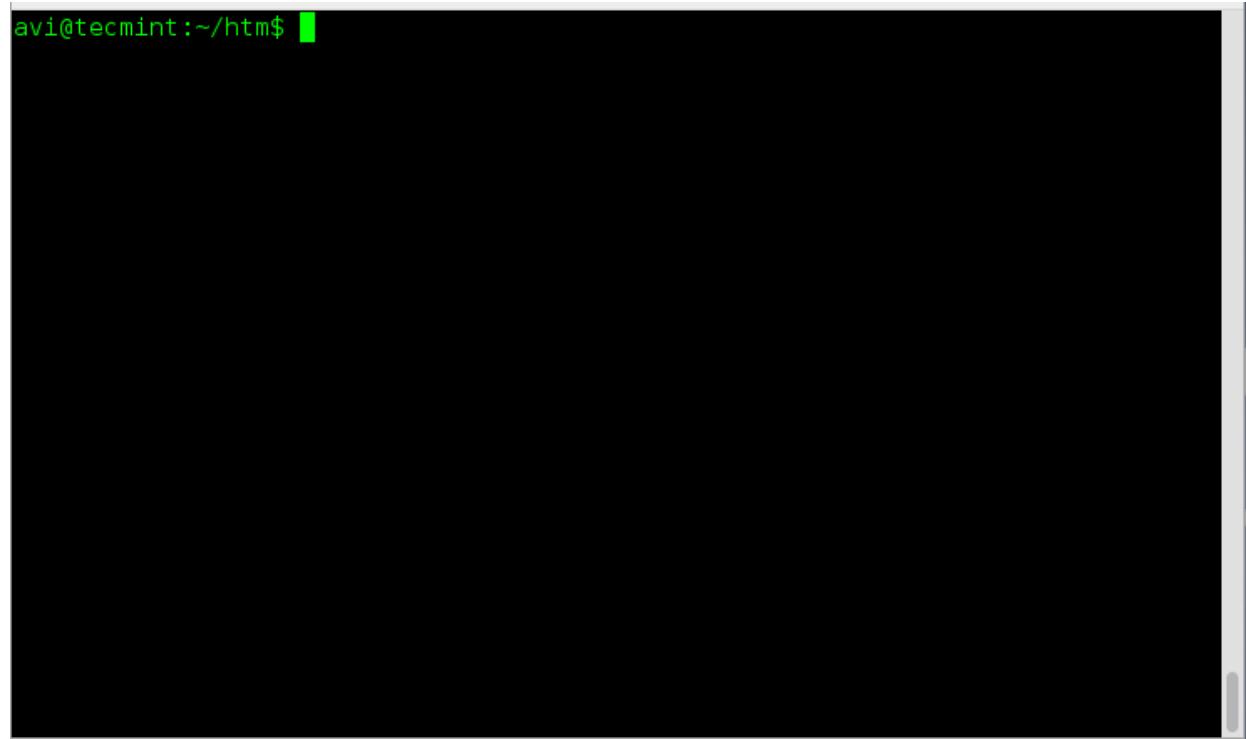
```
avi@tecmint:~$ /bin/pwd -P  
/var/www/html
```

```
avi@tecmint:~/htm$
```

Print Physical Working Directory

5. Check if the output of command “**pwd**” and “**pwd -P**” are same or not i.e., if no options are given at run-time does “**pwd**” takes option **-P** into account or not, automatically.

```
avi@tecmint:~$ /bin/pwd  
/var/www/html  
avi@tecmint:~/html$ █
```

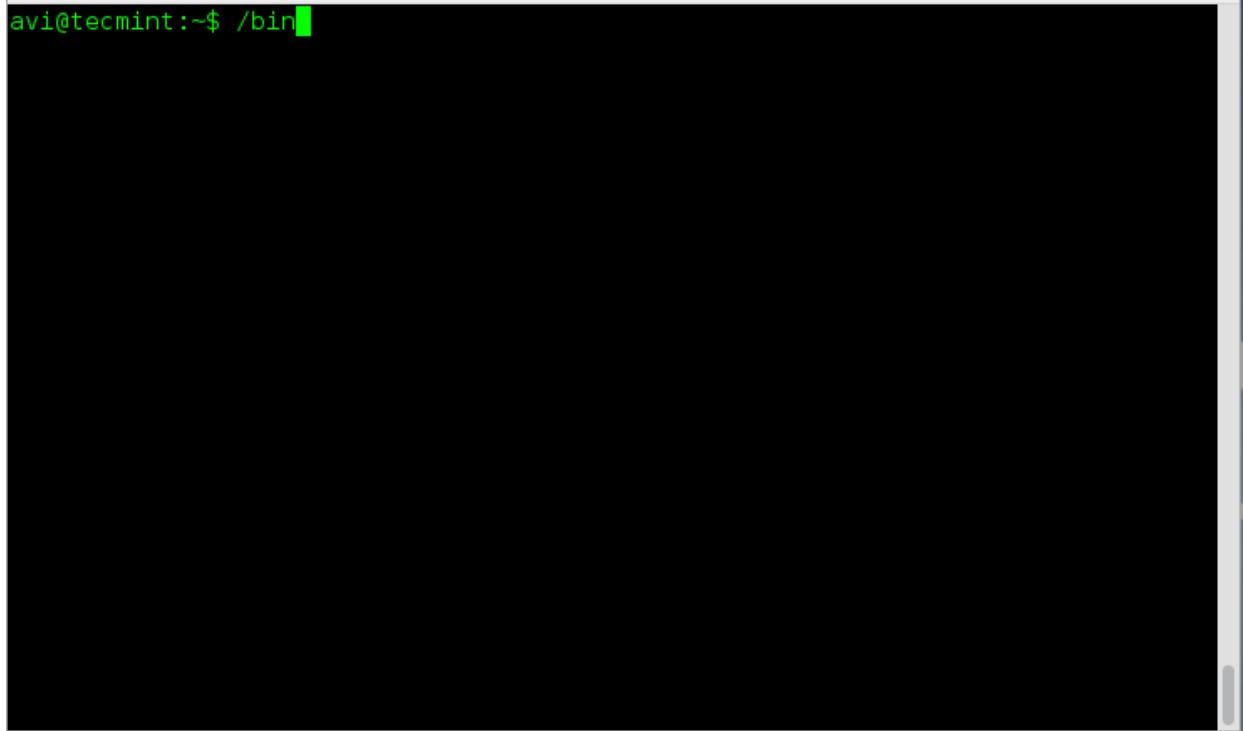


Check pwd Output

Result: It's clear from the above output of example 4 and 5 (both result are same) thus, when no options are specified with command “**pwd**”, it automatically takes option “**-P**” into account.

6. Print version of your ‘pwd’ command.

```
avi@tecmint:~$ /bin/pwd --version  
pwd (GNU coreutils) 8.23  
Copyright (C) 2014 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later  
<http://gnu.org/licenses/gpl.html>.  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
Written by Jim Meyering.
```

A screenshot of a terminal window. The prompt is "avi@tecmint:~\$". The user has typed "/bin/" and is pressing the Enter key. The terminal is black with white text.

Check pwd Version

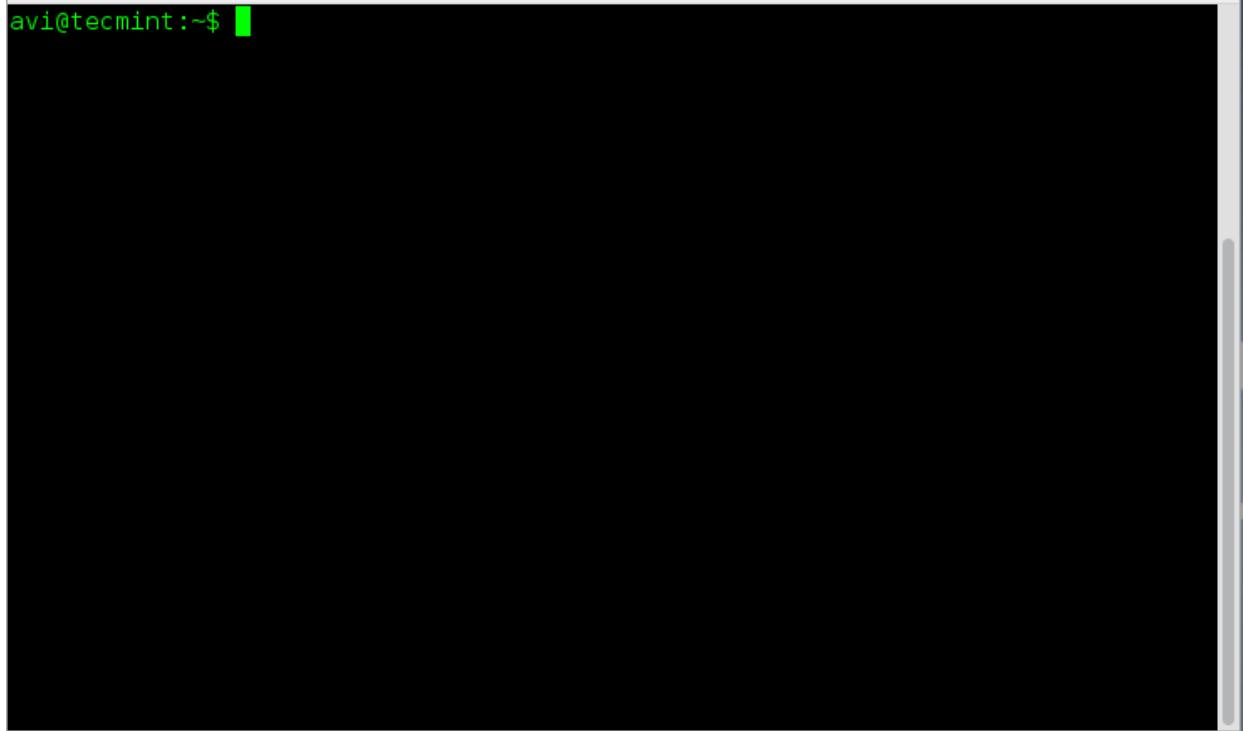
Note: A ‘pwd’ command is often used without options and never used with arguments.

Important: You might have noticed that we are executing the above command as “**/bin/pwd**” and not “**pwd**”.

So what's the difference? Well “**pwd**” alone means shell built-in pwd. Your shell may have different version of pwd. Please refer manual. When we are using **/bin/pwd**, we are calling the binary version of that command. Both the shell and the binary version of command Prints Current Working Directory, though the binary version have more options.

7. Print all the locations containing executable named pwd.

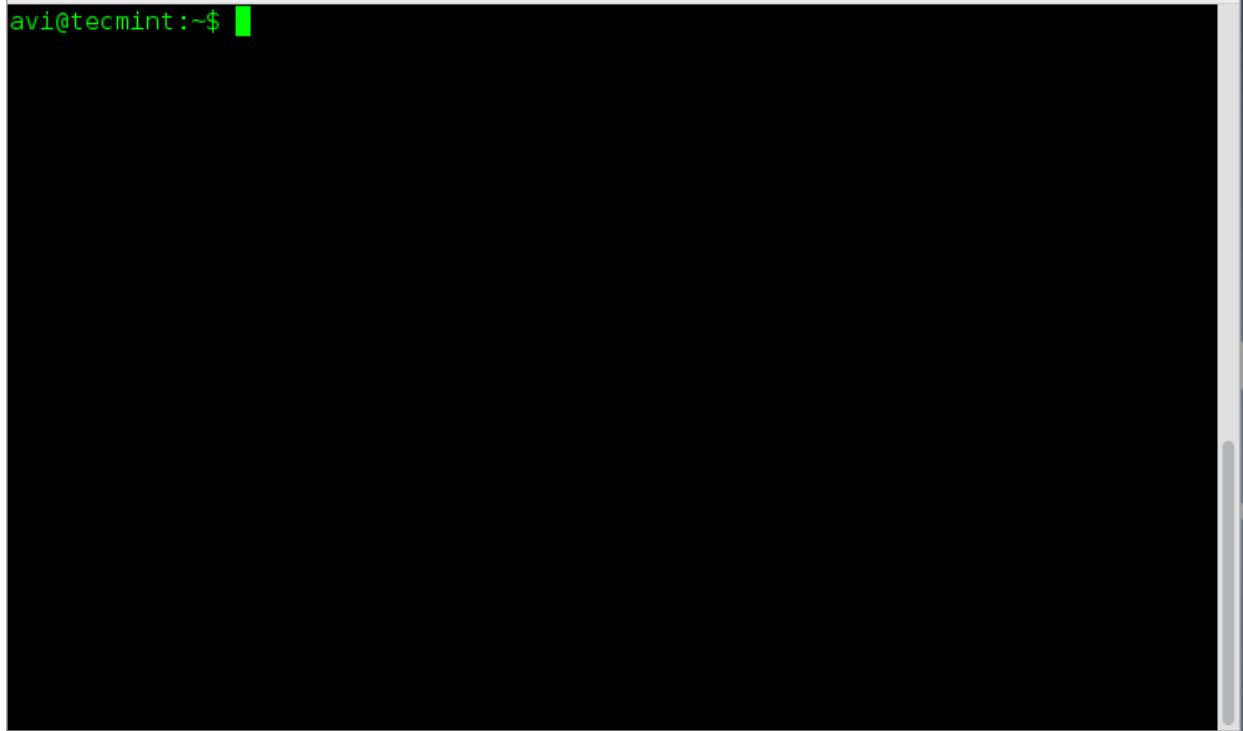
```
avi@tecmint:~$ type -a pwd
pwd is a shell builtin
pwd is /bin/pwd
```

A black terminal window with a white border, showing a command prompt "avi@tecmint:~\$". The main area of the terminal is completely black, indicating no output or a blank command line.

Print Executable Locations

8. Store the value of “**pwd**” command in variable (say **a**), and print its value from the variable (important for shell scripting perspective).

```
avi@tecmint:~$ a=$(pwd)
avi@tecmint:~$ echo "Current working directory is : $a"
Current working directory is : /home/avi
```

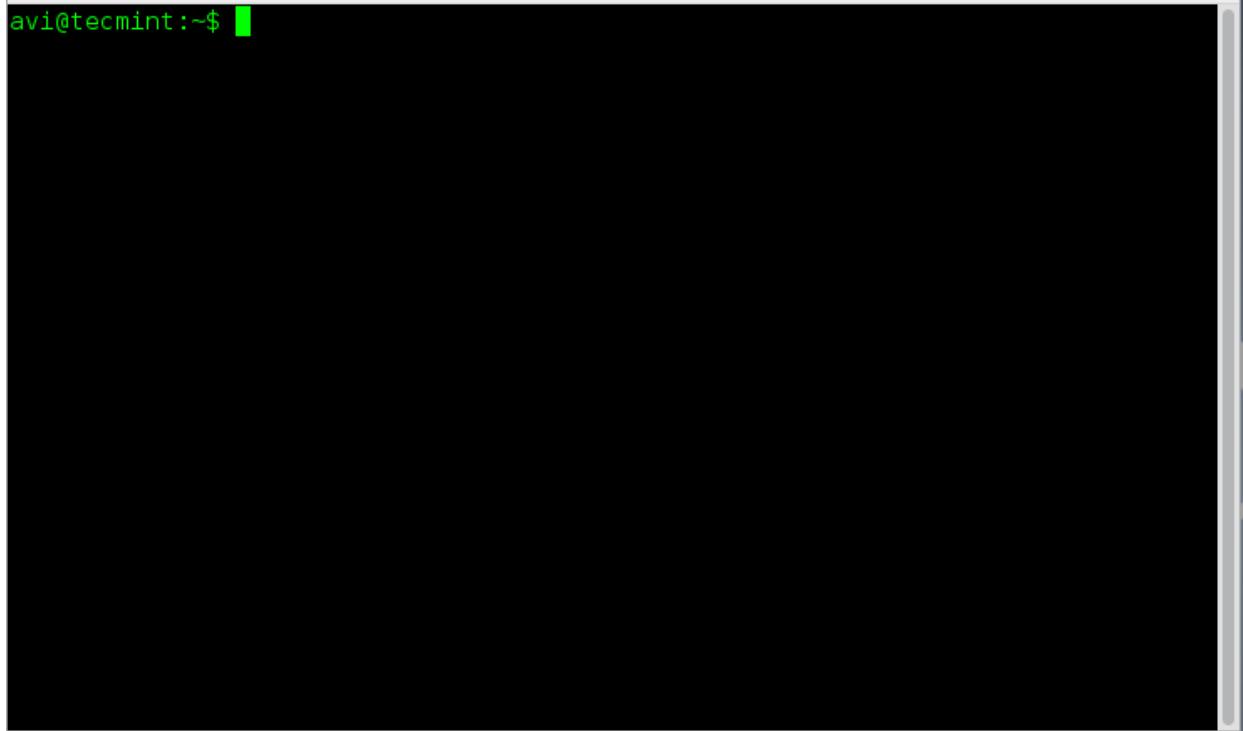
A black terminal window with a white border, showing a command prompt "avi@tecmint:~\$". The main area of the terminal is completely black, indicating no output or a blank command line.

Store Pwd Value in Variable

Alternatively, we can use **printf**, in the above example.

9. Change current working directory to anything (say **/home**) and display it in command line prompt. Execute a command (say '**ls**') to verify is everything is **OK**.

```
avi@tecmint:~$ cd /home
avi@tecmint:~$ PS1='$pwd> '
> ls [Notice single quotes in the example]
```

A black terminal window with a white border, showing a command-line interface. The prompt "avi@tecmint:~\$" is visible at the top left. The rest of the screen is blank, indicating no output from the command.

```
avi@tecmint:~$
```

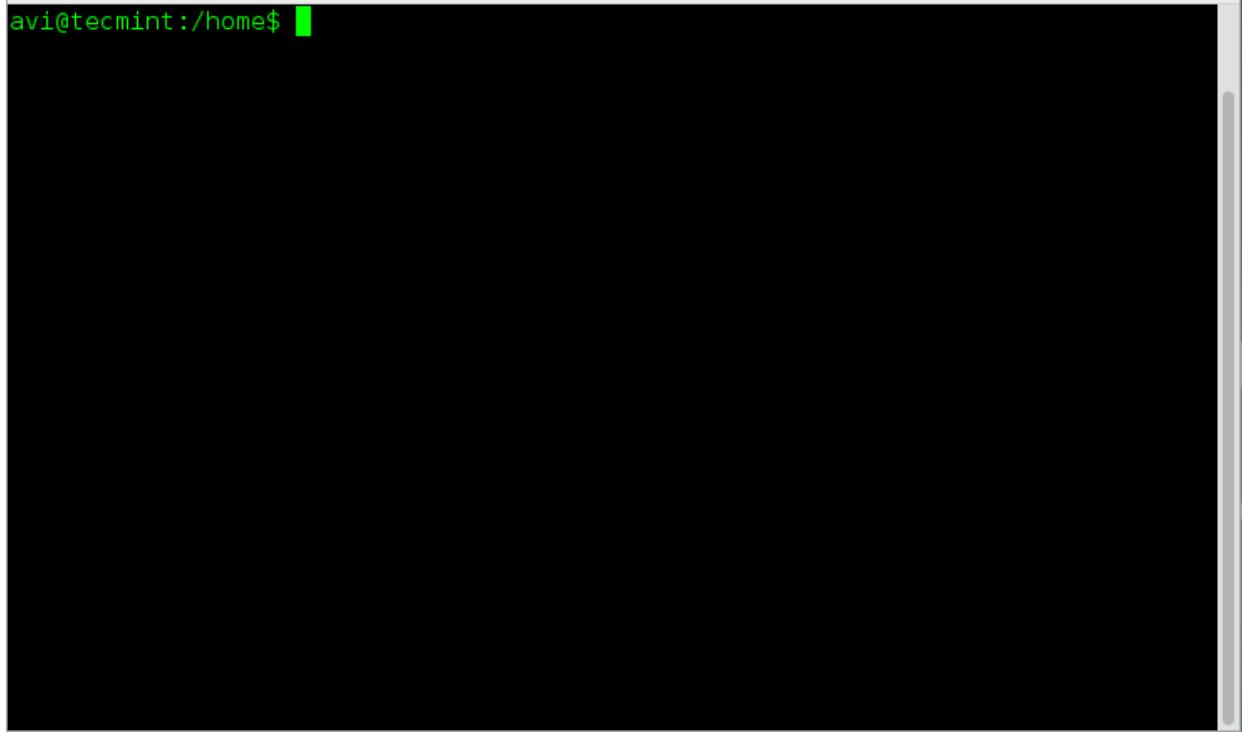
Change Current Working Directory

10. Set multi-line command line prompt (say something like below).

```
/home  
123#Hello#!
```

And then execute a command (say **ls**) to check is everything is **OK**.

```
avi@tecmint:~$ PS1='  
> $PWD  
$ 123#Hello#!  
$ '  
/home  
123#Hello#!
```

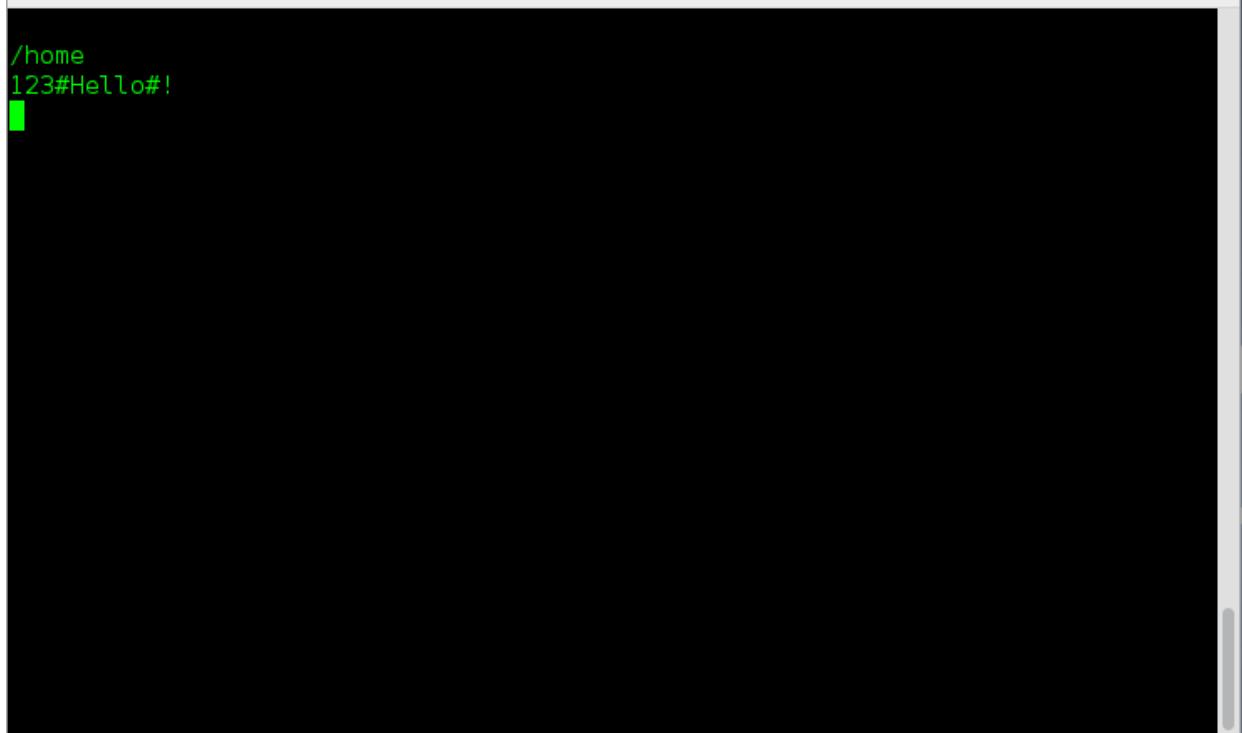


```
avi@tecmint:/home$
```

Set Multi Commandline Prompt

- 11.** Check the current working directory and previous working directory in one GO!

```
avi@tecmint:~$ echo "$PWD $OLDPWD"  
/home /home/avi
```



```
/home  
123#Hello#!
```

Check Present Previous Working Directory

12. What is the absolute path (starting from /) of the pwd binary file.

/bin/pwd

13. What is the absolute path (starting from /) of the pwd source file.

/usr/include/pwd.h

14. Print the absolute path (starting from /) of the pwd manual pages file.

/usr/share/man/man1/pwd.1.gz

15. Write a shell script analyses current directory (say **tecmint**) in your home directory. If you are under directory **tecmint** it output “**Well! You are in tecmint directory**” and then print “**Good Bye**” else create a directory **tecmint** under your home directory and ask you to **cd** to it.

Let's first create a ‘tecmint’ directory, under it create a following shell script file with name ‘pwd.sh’.

```
avi@tecmint:~$ mkdir tecmint
avi@tecmint:~$ cd tecmint
avi@tecmint:~$ nano pwd.sh
```

Next, add the following script to the pwd.sh file.

```
#!/bin/bash
x="$PWD"
if [ "$x" == "/home/$USER/tecmint" ]
then
{
echo "Well you are in tecmint directory"
echo "Good Bye"
}
else
{
mkdir /home/$USER/tecmint
echo "Created Directory tecmint you may now cd to it"
}
fi
```

Give execute permission and run it.

```
avi@tecmint:~$ chmod 755 pwd.sh
avi@tecmint:~$ ./pwd.sh
Well you are in tecmint directory
Good Bye
```

Conclusion

pwd is one of the simplest yet most popular and most widely used command. A good command over pwd is basic to use Linux terminal. That's all for now. I'll be here again with another interesting article soon, till then stay tuned and connected to Tecmint.

15 Practical Examples of ‘cd’ Command in Linux

by [Editor](#) | Published: August 19, 2014 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In Linux ‘cd’ (**C**hange **D**irectory) command is one of the most important and most widely used command for newbies as well as system administrators. For admins on a headless server, ‘cd’ is the only way to navigate to a directory to check log, execute a program/application/script and for every other task. For newbie it is among those initial commands they make their hands dirty with.



15 cd command examples in linux

Thus, keeping in mind, we here bringing you **15** basic commands of ‘cd’ using tricks and shortcuts to reduce your efforts on the terminal and save time by using these known tricks.

Tutorial Details

1. **Command Name :** cd
2. **Stands for :** Change Directory
3. **Availability :** All Linux Distribution
4. **Execute On :** Command Line
5. **Permission :** Access own directory or otherwise assigned.

6. **Level** : Basic/Beginners

1. Change from current directory to /usr/local.

```
avi@tecmint:~$ cd /usr/local  
avi@tecmint:/usr/local$
```

2. Change from current directory to /usr/local/lib using absolute path.

```
avi@tecmint:/usr/local$ cd /usr/local/lib  
avi@tecmint:/usr/local/lib$
```

3. Change from current working directory to /usr/local/lib using relative path.

```
avi@tecmint:/usr/local$ cd lib  
avi@tecmint:/usr/local/lib$
```

4. (a) Move one directory back from where you are now.

```
avi@tecmint:/usr/local/lib$ cd -  
/usr/local  
avi@tecmint:/usr/local$
```

4. (b) Change Current directory to parent directory.

```
avi@tecmint:/usr/local/lib$ cd ..  
avi@tecmint:/usr/local$
```

5. Show last working directory from where we moved (use ‘-‘ switch) as shown.

```
avi@tecmint:/usr/local$ cd --  
/home/avi
```

6. Move two directory up from where you are now.

```
avi@tecmint:/usr/local$ cd ../../..  
avi@tecmint:/usr$
```

7. Move to users home directory from anywhere.

```
avi@tecmint:/usr/local$ cd ~  
avi@tecmint:~$  
or  
avi@tecmint:/usr/local$ cd  
avi@tecmint:~$
```

8. Change working directory to current working directory (seems no use in General).

```
avi@tecmint:~/Downloads$ cd .  
avi@tecmint:~/Downloads$
```

```
or  
avi@tecmint:~/Downloads$ cd ./  
avi@tecmint:~/Downloads$
```

9. Your present working Directory is “/usr/local/lib/python3.4/dist-packages/ ”, change it to “/home/avi/Desktop/ ”, in one line command, by moving up in the directory till ‘/’ then using absolute path.

```
avi@tecmint:/usr/local/lib/python3.4/dist-packages$ cd  
../../../../../../../home/avi/Desktop/  
avi@tecmint:~/Desktop$
```

10. Change from current working directory to /var/www/html without typing in full using TAB.

```
avi@tecmint:/var/www$ cd /v<TAB>/w<TAB>/h<TAB>  
avi@tecmint:/var/www/html$
```

11. Navigate from your current working directory to /etc/v__ __, Oops! You forgot the name of directory and not supposed to use TAB.

```
avi@tecmint:~$ cd /etc/v*  
avi@tecmint:/etc/vbox$
```

Note: This will move to ‘vbox’ only if there is only one directory starting with ‘v’. If more than one directory starting with ‘v’ exist, and no more criteria is provided in command line, it will move to the first directory starting with ‘v’, alphabetically as their presence in standard dictionary.

12. You need to navigate to user ‘av’ (not sure if it is avi or avt) home directory, without using TAB.

```
avi@tecmint:/etc$ cd /home/av?  
avi@tecmint:~$
```

13. What are pushd and popd in Linux?

Pushd and popd are Linux commands in bash and certain other shell which saves current working directory location to memory and bring to the directory from memory as current working directory, respectively as well as changes directory.

```
avi@tecmint:~$ pushd /var/www/html  
/var/www/html ~  
avi@tecmint:/var/www/html$
```

The above command saves the current location to memory and changes to the requested directory. As soon as popd is fired, it fetch the saved directory location from memory and makes it current working directory.

```
avi@tecmint:/var/www/html$ popd
```

```
~  
avi@tecmint:~$
```

14. Change to a directory containing white spaces.

```
avi@tecmint:~$ cd test\ tecmint/  
avi@tecmint:~/test tecmint$  
or  
avi@tecmint:~$ cd 'test tecmint'  
avi@tecmint:~/test tecmint$  
or  
avi@tecmint:~$ cd "test tecmint"/  
avi@tecmint:~/test tecmint$
```

15. Change from current working directory to Downloads and list all its settings in one go.

```
avi@tecmint:/usr$ cd ~/Downloads && ls  
...  
.  
service_locator_in.xls  
sources.list  
teamviewer_linux_x64.deb  
tor-browser-linux64-3.6.3_en-US.tar.xz  
. ....
```

This is our attempt, to make you aware of Linux Workings and executions in least possible words and with as much as user friendliness as it used to be before.

That's all for now. I'll be here again with another interesting topic soon. Till then stay tuned and connected to Tecmint. Don't forget to provide us with your valuable feedback in the comments below.

How to Find Number of Files in a Directory and Subdirectories

by [Aaron Kili](#) | Published: January 17, 2017 | Last Updated: January 17, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this guide, we will cover how to display the total number of files in the current working directory or any other directory and its subdirectories on a Linux system.

We will use the [find command](#) which is used to search for files in a directory hierarchy together with [wc command](#) which prints newline, word, and byte counts for each file, alternatively data read from standard input.

Following are the options that we can use with [find command](#) as follows:

1. `-type` – specifies the file type to search for, in the case above, the `f` means find all regular files.
2. `-print` – an action to print the absolute path of a file.
3. `-l` – this option prints the total number of newlines, which is equals to the total number of absolute file paths output by [find command](#).

The general syntax of find command.

```
# find . -type f -print | wc -l
$ sudo find . -type f -print | wc -l
```

Important: Use [sudo command](#) to read all files in the specified directory including those in the subdirectories with superuser privileges, in order to avoid “**Permission denied**” errors as in the screen shot below:

```
aaronkilik@tecmint ~ $ find . -type f -print | wc -l
find: './.config/etcher': Permission denied
find: './.pki': Permission denied
find: './.cache/dconf': Permission denied
58043
aaronkilik@tecmint ~ $ sudo find . -type f -print | wc -l
58061
aaronkilik@tecmint ~ $
```

Find Number of Files in Linux

You can see that in the first command above, not all files in the current working directory are read by **find** command.

The following are extra examples to show total number of regular files in `/var/log` and `/etc` directories respectively:

```
$ sudo find /var/log/ -type f -print | wc -l  
$ sudo find /etc/ -type f -print | wc -l
```

For more examples on Linux **find command** and **wc command** go through the following series of articles for additional usage options, tips and related commands:

1. [35 Useful ‘find’ Command Examples in Linux](#)
2. [How to Find Recent or Today’s Modified Files in Linux](#)
3. [Find Top 10 Directories and Files Disk Space in Linux](#)
4. [6 Useful ‘wc’ Command Examples to Count Lines, Words and Characters](#)

That's all! In case you know of any other method to display the total number of files in a directory and its subdirectories, do share it with us in the comments.

Learn How to Set Your \$PATH Variables Permanently in Linux

by [Marin Todorov](#) | Published: April 27, 2016 | April 27, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In Linux (also UNIX) **\$PATH** is environment variable, used to tell the shell where to look for executable files. **\$PATH** variable provides great flexibility and security to the Linux systems and it is definitely safe to say that it is one of the most important environment variables.

Don't Miss: [How to Set and Unset Local, User and System Wide Environment Variables](#)

Programs/scripts that are located within the **\$PATH**'s directory, can be executed directly in your shell, without specifying the full path to them. In this tutorial you are going to learn how to set **\$PATH** variable globally and locally.

First, let's see your current **\$PATH**'s value. Open a terminal and issue the following command:

```
$ echo $PATH
```

The result should be something like this:

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

The result shows a list of directories separated by colons. You can easily add more directories by editing your user's shell profile file.

In different shells this can be:

1. Bash shell -> `~/.bash_profile`, `~/.bashrc` or `profile`
2. Korn Shell -> `~/.kshrc` or `.profile`
3. Z shell -> `~/.zshrc` or `.zprofile`

Please note that depending on how you are logging to the system in question, different file might be read. Here is what the bash manual says, keep in mind that the files are similar for other shells:

```
/bin/bash  
The bash executable  
/etc/profile  
The systemwide initialization file, executed for login shells  
~/.bash_profile  
The personal initialization file, executed for login shells
```

```
~/.bashrc  
The individual per-interactive-shell startup file  
~/.bash_logout  
The individual login shell cleanup file, executed when a login shell exits  
~/.inputrc  
Individual readline initialization file
```

Considering the above, you can add more directories to the **\$PATH** variable by adding the following line to the corresponding file that you will be using:

```
$ export PATH=$PATH:/path/to/newdir
```

Of course in the above example, you should change “**/path/to/newdir**” with the exact path that you wish to set. Once you have modified your **.*rc** or **.*_profile** file you will need to call it again using the “**source**” command.

For example in bash you can do this:

```
$ source ~/.bashrc
```

Below, you can see an example of mine **\$PATH** environment on a local computer:

```
marin@[TecMint]:[/home/marin] $ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/  
local/games:/home/marin/bin
```

This is actually a good practice to create a local “**bin**” folder for users where they can place their executable files. Each user will have its separate folder to store his contents. This is also a good measure to keep your system secured.

If you have any questions or difficulties setting your **\$PATH** environment variable, please do not hesitate to submit your questions in the comment section below.

Learn Why ‘less’ is Faster Than ‘more’ Command for Effective File Navigation

by [Matei Cezar](#) | Published: April 14, 2016 | Last Updated: April 14, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

More is a *nix command line used to display the contents of a file in a console. The basic usage of **more** command is to run the command against a file as shown below:

Read Also: [Learn Difference Between ‘cat’ and ‘tac’ Commands with Examples](#)

Learn Linux ‘more’ Command

```
# more /var/log/auth.log
View Contents of auth.log File
Apr 12 11:50:01 tecmint CRON[6932]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 11:50:01 tecmint CRON[6932]: pam_unix(cron:session): session closed
for user root
Apr 12 11:55:01 tecmint CRON[7159]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 11:55:01 tecmint CRON[7160]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 11:55:01 tecmint CRON[7160]: pam_unix(cron:session): session closed
for user root
Apr 12 11:55:02 tecmint CRON[7159]: pam_unix(cron:session): session closed
for user root
Apr 12 12:00:01 tecmint CRON[7290]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 12:00:01 tecmint CRON[7290]: pam_unix(cron:session): session closed
for user root
Apr 12 12:05:01 tecmint CRON[7435]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 12:05:01 tecmint CRON[7436]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 12:05:01 tecmint CRON[7436]: pam_unix(cron:session): session closed
for user root
Apr 12 12:05:02 tecmint CRON[7435]: pam_unix(cron:session): session closed
for user root
Apr 12 12:09:01 tecmint CRON[7542]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 12:09:01 tecmint CRON[7542]: pam_unix(cron:session): session closed
for user root
Apr 12 12:10:01 tecmint CRON[7577]: pam_unix(cron:session): session opened
for user root by (uid=0)
Apr 12 12:10:01 tecmint CRON[7577]: pam_unix(cron:session): session closed
for user root
Apr 12 12:15:01 tecmint CRON[7699]: pam_unix(cron:session): session opened
for user root by (uid=0)
```

```
Apr 12 12:15:01 tecmint CRON[7700]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Apr 12 12:15:01 tecmint CRON[7700]: pam_unix(cron:session): session closed  
for user root  
Apr 12 12:15:01 tecmint CRON[7699]: pam_unix(cron:session): session closed  
for user root  
....
```

Another way to use **more** command in conjunction (pipe) with other commands, such as [cat command](#), as presented on below example:

```
# cat /var/log/auth.log | more  
tecmint@mytecmint ~ $ cat /var/log/auth.log | more|
```

View and Navigate Contents of File

In order to navigate through the file line by line press `Enter` key or press `Spacebar` key to navigate one page at a time, the page being your current terminal screen size. To exit the command just press `q` key.

A useful option of more command is the `-number` switch which allows you to set the number of line a page should contain. As an example display the `auth.log` file as a page of 10 lines:

```
# more -10 /var/log/auth.log  
tecmint@mytecmint ~ $ more -10 /var/log/auth.log
```

Show Only First 10 Lines of File

Also, you can display a page starting from a specific line number using the `+number` option as illustrated below:

```
# more +14 /var/log/auth.log
Show Only First 14 Lines of auth.log File
Apr 12 12:09:01 tecmint CRON[7542]: pam_unix(cron:session): session closed
for user root
Apr 12 12:10:01 tecmint CRON[7577]: pam_unix(cron:session): session opened
for user root by (
uid=0)
Apr 12 12:10:01 tecmint CRON[7577]: pam_unix(cron:session): session closed
for user root
Apr 12 12:15:01 tecmint CRON[7699]: pam_unix(cron:session): session opened
for user root by (
uid=0)
Apr 12 12:15:01 tecmint CRON[7700]: pam_unix(cron:session): session opened
for user root by (
uid=0)
Apr 12 12:15:01 tecmint CRON[7700]: pam_unix(cron:session): session closed
for user root
Apr 12 12:15:01 tecmint CRON[7699]: pam_unix(cron:session): session closed
for user root
Apr 12 12:16:01 tecmint mate-screensaver-dialog: gkr-pam: unlocked login
keyring
Apr 12 12:17:01 tecmint CRON[7793]: pam_unix(cron:session): session opened
for user root by (
uid=0)
Apr 12 12:17:01 tecmint CRON[7793]: pam_unix(cron:session): session closed
for user root
Apr 12 12:20:01 tecmint CRON[7905]: pam_unix(cron:session): session opened
for user root by (
uid=0)
Apr 12 12:20:01 tecmint CRON[7905]: pam_unix(cron:session): session closed
for user root
Apr 12 12:25:01 tecmint CRON[8107]: pam_unix(cron:session): session opened
for user root by (
uid=0)
Apr 12 12:25:01 tecmint CRON[8108]: pam_unix(cron:session): session opened
for user root by (
```

Learn Linux ‘less’ Command

Similar to **more**, **less** command allows you to view the contents of a file and navigate through file. The main difference between **more** and **less** is that **less** command is faster because it does not load the entire file at once and allows navigation though file using page **up/down** keys.

In can be used as a standalone command issued against a file or used with pipes with a multitude of Linux commands in order to narrow their screen output allowing you to scroll through results.

```
# less /var/log/auth.log
# ls /etc | less
```

You can navigate through the file line by line pressing `Enter` key. Page navigation can be handled with `spacebar` key. The page size is represented by your current terminal screen size. To exit command type `q` key, same way as for more command.

A useful feature of `less` command is the use of `/word-to-search` option. For instance you can search and match all `sshd` messages from a log file by interactively specifying the `/sshd` string.

```
tecmint@mytecmint ~ $ less /var/log/auth.log
```

View File Content Using less Command

In order to display a file starting at a specific line number use the following syntax:

```
# less +5 /var/log/auth.log
```

If you need to track down the number of every line with **less** command use the **-N** option.

```
# less -N /var/log/daemon.log
```

Show Number for Every Line in File

```
1 Apr 12 11:50:01 tecmint CRON[6932]: pam_unix(cron:session): session opened for user root by (uid=0)
2 Apr 12 11:50:01 tecmint CRON[6932]: pam_unix(cron:session): session closed for user root
3 Apr 12 11:55:01 tecmint CRON[7159]: pam_unix(cron:session): session opened for user root by (uid=0)
4 Apr 12 11:55:01 tecmint CRON[7160]: pam_unix(cron:session): session opened for user root by (uid=0)
5 Apr 12 11:55:01 tecmint CRON[7160]: pam_unix(cron:session): session closed for user root
6 Apr 12 11:55:02 tecmint CRON[7159]: pam_unix(cron:session): session closed for user root
7 Apr 12 12:00:01 tecmint CRON[7290]: pam_unix(cron:session): session opened for user root by (uid=0)
8 Apr 12 12:00:01 tecmint CRON[7290]: pam_unix(cron:session): session closed for user root
9 Apr 12 12:05:01 tecmint CRON[7435]: pam_unix(cron:session): session opened for user root by (uid=0)
10 Apr 12 12:05:01 tecmint CRON[7436]: pam_unix(cron:session): session opened for user root by (uid=0)
11 Apr 12 12:05:01 tecmint CRON[7436]: pam_unix(cron:session): session closed for user root
```

By default the only way to exit **less** command is to hit **q** key. To change this behavior and automatically exit file when reaching the end of file use the **-e** or **-E** option:

```
# less -e /var/log/auth.log
# less -E /var/log/auth.log
```

To open a file at the first occurrence of a pattern use the following syntax:

```
# less +/sshd /var/log/auth.log
```

Show Given Matching String in File

```
Apr 12 16:19:39 tecmint sshd[16666]: Accepted password for tecmint from 192.168.0.15 port 41634 ssh2
Apr 12 16:19:39 tecmint sshd[16666]: pam_unix(sshd:session): session opened for user tecmint by (uid=0)
Apr 12 16:19:39 tecmint systemd-logind[954]: New session 1 of user tecmint.
Apr 12 16:19:48 tecmint sshd[16728]: Received disconnect from 192.168.0.15: 11: disconnected by user
Apr 12 16:19:48 tecmint sshd[16666]: pam_unix(sshd:session): session closed for user tecmint
Apr 12 16:20:01 tecmint CRON[16799]: pam_unix(cron:session): session opened for user root by (uid=0)
Apr 12 16:20:02 tecmint CRON[16799]: pam_unix(cron:session): session closed for user root
Apr 12 16:25:01 tecmint CRON[17026]: pam_unix(cron:session): session opened for user root by (uid=0)
```

```
Apr 12 16:25:01 tecmint CRON[17025]: pam_unix(cron:session): session opened  
for user root by (uid=0)
```

The above command tells **less** to open **auth.log** file at the first match of **sshd** string.

In order to automatically append the content of a file opened in **less** command use the **Shift+f** keys combination or run less with the following syntax.

```
# less +F /var/log/syslog
```

This makes less to run in interactive mode (live) and display new content on-fly while waiting for new data to be written to file. This behavior is similar to [tail -f command](#).

In combination with a pattern you can watch the log file interactively with **Shift+f** key stroke while matching a keyword. To exit live mode just press **Ctrl+c** keys.

```
# less +/CRON /var/log/syslog
```

Whether you decide to use **more** or **less**, which is a personal choice, remember that **less** is more with **more** features.

How to Set or Change System Hostname in Linux

by [Marin Todorov](#) | Published: April 14, 2016 | Last Updated: April 14, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Device or system hostnames are used to easily recognize a machine within a network in a human readable format. It is not much of a surprise, but on Linux system, the hostname can be easily changed by using simple command as “**hostname**“.

Read Also: [How to Set Static IP Address and Configure Network in Linux](#)

Running **hostname** on its own, without any parameters, will return the current hostname of your Linux system like this:

```
$ hostname  
TecMint
```

If you want to change or set hostname of your Linux system, simply run:

```
$ hostname NEW_HOSTNAME
```

Of course, you will need to replace “**NEW_HOSTNAME**” with the actual hostname that you wish to set. This will change the hostname of your system immediately, but there is one problem – the original hostname will be restored upon next reboot.

There is another way to change the hostname of your system – permanently. You might have already figured it out that this will require change in some configuration files and you will be correct.

Set System Hostname Permanently in Linux

Newer version of different Linux distributions such as latest **Ubuntu**, **Debian**, **CentOS**, **Fedora**, **RedHat**, etc. comes with **systemd**, a system and service manager that provides a **hostnamectl** command to manage hostnames in Linux.

To set system hostname on **SystemD** based distributions, we will use **hostnamectl** command as shown:

```
$ sudo hostnamectl set-hostname NEW_HOSTNAME
```

For Older Linux distributions, which uses **SysVinit** in short **init**, can have their hostnames changed by simply editing the hostname file located in:

```
# vi /etc/hostname
```

You then have to add another record for the hostname in:

```
# vi /etc/hosts
```

For example:

```
127.0.0.1 TecMint
```

You then need to run:

```
# /etc/init.d/hostname restart
```

On **RHEL/CentOS** based systems that use **init**, the hostname is changed by modifying:

```
# vi /etc/sysconfig/network
```

Here is a sample of that file:

```
/etc/sysconfig/network
NETWORKING=yes
HOSTNAME="tecmint.com"
GATEWAY="192.168.0.1"
GATEWAYDEV="eth0"
FORWARD_IPV4="yes"
```

To keep a permanent hostname change the value next to "HOSTNAME" to the one of your hostname.

Conclusion

This simple article meant to show you a simple Linux trick and I hope that you learned something new.

How to Use ‘cat’ and ‘tac’ Commands with Examples in Linux

by [Matei Cezar](#) | Published: April 8, 2016 | Last Updated: April 9, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

This article is a part of our [Linux Tricks and Tips](#) series, in this article we will cover some basic usage of **cat** command (most frequently used command in Linux) and **tac** (reverse of cat command – print files in reverse order) with some practical examples.

Read Also: [13 Useful ‘cat’ Command Examples in Linux](#)

Basic Usage of Cat Command in Linux

Cat command, acronym for **Concatenate**, is one of the most used commands in *nix systems. The most basic usage of the command is to read files and display them to **stdout**, meaning to display the content of files on your terminal.

```
# cat file.txt  
tecmint@tecmint ~ $
```

View Content of File in Linux

Another usage of the **cat** command is to read or combine multiple files together and send the output to a monitor as illustrated in the below examples.

```
# cat file1.txt file2.txt file3.txt  
tecmint@tecmint ~ $
```

View Content of Multiple Files

The command can also be used to concatenate (join) multiple files into one single file using the “>” Linux redirection operator.

```
# cat file1.txt file2.txt file3.txt > file-all.txt
```

```
tecmint@tecmint ~ $ cat file1.txt file2.txt file3.txt > file-all.txt
```

Join Multiple Files in Linux

By using the append redirector you can add the content of a new file to the bottom of the `file-all.txt` with the following syntax.

```
# cat file4.txt >> file-all.txt  
tecmint@tecmint ~ $ cat file4.
```

Append Content File to New File

The `cat` command can be used to copy the content of file to a new file. The new file can be renamed arbitrary. For example, copy the file from the current location to `/tmp/` directory.

```
# cat file1.txt > /tmp/file1.txt  
tecmint@tecmint ~ $ cat file1.txt >/tmp/file1.txt
```

Copy Content of File to New File

Copy the file from the current location to `/tmp/` directory and change its name.

```
# cat file1.txt > /tmp/newfile.cfg
```

```
tecmint@tecmint ~ $ cat file1.txt > /tmp/newfile.cfg
```

Copy File to /tmp Location

A less usage of the **cat** command is to create a new file with the below syntax. When finished editing the file hit **CTRL+D** to save and exit the new file.

```
# cat > new_file.txt
```

```
tecmint@tecmint ~ $
```

Create New File using Cat Command

In order to number all output lines of a file, including empty lines, use the **-n** switch.

```
# cat -n file-all.txt
```

```
tecmint@tecmint ~ $
```

Add Numbers to Lines in File

To display only the number of each non-empty line use the **-b** switch.

```
# cat -b file-all.txt
```

```
tecmint@tecmint ~ $
```

Print Line Numbers in File

Want to learn more about Linux cat command? then read our article about [13 Useful ‘cat’ Command Examples in Linux](#).

Learn How to Use Tac Command in Linux

On the other hand, a lesser known and less used command in *nix systems is `tac` command. **Tac** is practically the reverse version of `cat` command (also spelled backwards) which prints each line of a file starting from the bottom line and finishing on the top line to your machine standard output.

```
# tac file-all.txt
```

```
tecmint@tecmint ~ $
```

Print Content File in Reverse Order

One of the most important option of the command is represented by the `-s` switch, which separates the contents of the file based on a string or a keyword from the file.

```
# tac file-all.txt --separator "two"
```

```
tecmint@tecmint ~ $ tac file-all.txt --separator "two"
```

Remove Matching String in File

Next, most important usage of **tac** command is, that it can provide a great help in order to debug log files, reversing the chronological order of log contents.

```
$ tac /var/log/auth.log  
Or to display the last lines  
$ tail /var/log/auth.log | tac
```

Sample Output

```
tecmint@tecmint ~ $ tac /var/log/auth.log  
pr 6 16:09:01 tecmint CRON[17714]: pam_unix(cron:session): session closed  
for user root  
Apr 6 16:09:01 tecmint CRON[17714]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Apr 6 16:05:01 tecmint CRON[17582]: pam_unix(cron:session): session closed  
for user root  
Apr 6 16:05:01 tecmint CRON[17583]: pam_unix(cron:session): session closed  
for user root  
Apr 6 16:05:01 tecmint CRON[17583]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Apr 6 16:05:01 tecmint CRON[17582]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Apr 6 16:00:01 tecmint CRON[17434]: pam_unix(cron:session): session closed  
for user root  
....  
tecmint@tecmint ~ $ tail /var/log/auth.log | tac  
Apr 6 16:09:01 tecmint CRON[17714]: pam_unix(cron:session): session closed  
for user root  
Apr 6 16:09:01 tecmint CRON[17714]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Apr 6 16:05:01 tecmint CRON[17582]: pam_unix(cron:session): session closed  
for user root  
Apr 6 16:05:01 tecmint CRON[17583]: pam_unix(cron:session): session closed  
for user root  
Apr 6 16:05:01 tecmint CRON[17583]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Apr 6 16:05:01 tecmint CRON[17582]: pam_unix(cron:session): session opened  
for user root by (uid=0)  
Apr 6 16:00:01 tecmint CRON[17434]: pam_unix(cron:session): session closed  
for user root  
Apr 6 16:00:01 tecmint CRON[17434]: pam_unix(cron:session): session opened  
for user root by (uid=0)
```

```
Apr  6 15:55:02 tecmint CRON[17194]: pam_unix(cron:session): session closed  
for user root  
Apr  6 15:55:01 tecmint CRON[17195]: pam_unix(cron:session): session closed  
for user root  
...
```

Same as `cat` command, `tac` does an excellent job in [manipulating text files](#), but it should be avoided in other type of files, especially binary files or on files where the first line denotes the program that will run it.

ifconfig vs ip: What's Difference and Comparing Network Configuration

by [Gunjit Khera](#) | Published: February 23, 2016 | February 23, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Linux based distributions have featured set of commands which provide way to configure networking in easy and powerful way through command-line. These set of commands are available from **net-tools** package which has been there for a long time on almost all distributions, and includes commands like: **ifconfig**, **route**, **nameif**, **iwconfig**, **iptunnel**, **netstat**, **arp**.

```
tecmint@tecmint ~ $ ifconfig eth0
eth0      Link encap:Ethernet HWaddr 28:d2:44:eb:bd:98
          inet addr:192.168.0.104 Bcast:192.168.0.255
          Mask:255.255.255.0
          inet6 addr: fe80::2ad2:44ff:feeb:bd98/64 Scop
          e:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:426955 errors:0 dropped:0 overruns:0 frame:0
          TX packets:309304 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:470361359 (470.3 MB) TX bytes:38417
          974 (38.4 MB)
```

```
tecmint@tecmint ~ $ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
      link/ether 28:d2:44:eb:bd:98 brd ff:ff:ff:ff:ff:ff
      inet 192.168.0.104/24 brd 192.168.0.255 scope global eth0
          valid_lft forever preferred_lft forever
      inet6 fe80::2ad2:44ff:feeb:bd98/64 scope link
          valid_lft forever preferred_lft forever
tecmint@tecmint ~ $
```

ifconfig vs ip: What's Difference and Comparing Commands

Ifconfig Vs IP Command

These commands are just about sufficient in configuring the network in a way any novice or an expert Linux user would want, but due to advancement in Linux kernel over past years and unmaintainable of this packaged set of commands, they are getting deprecated and a more powerful alternative which has ability to replace all of these commands is emerging.

This alternative has also been there for quite some time now and is much more powerful than any of these commands. Rest of sections would highlight this alternative and compare it with one of the command from net-tools package i.e. **ifconfig**.

ip – A Replacement for ifconfig

ifconfig has been there for a long time and is still used to configure, display and control network interfaces by many, but a new alternative now exists on Linux distributions which is much more powerful than it. This alternative is **ip** command from **iproute2util** package.

Although this command might seem a bit complex at first site but it is much broader in functionality than **ifconfig**. It is functionally organized on two layers of Networking Stack i.e. **Layer 2 (Link Layer)**, **Layer 3 (IP Layer)** and does the work of all the above mentioned commands from net-tools package.

While **ifconfig** mostly displays or modifies the interfaces of a system, this command is capable of doing following tasks:

1. Displaying or Modifying Interface properties.
2. Adding, Removing ARP Cache entries along creating new Static ARP entry for a host.
3. Displaying MAC addresses associated with all the interfaces.
4. Displaying and modifying kernel routing tables.

One of the main highlight which separates it from its ancient counterpart **ifconfig** is that latter uses **ioctl** for network configuration, which is a less appreciated way of interaction with kernel while former takes advantage of netlink socket mechanism for the same which is a much more flexible successor of ioctl for inter-communication between kernel and user space using rtnetlink (which adds networking environment manipulation capability).

We can now begin to highlight the features of **ifconfig** and how they are effectively replaced by **ip** command.

ip vs ifconfig Commands

Following section highlights some of **ifconfig** commands and their replacement using **ip** commands:

1. Displaying all Network Interfaces in Linux

Here, one distinguishing feature between **ip** and **ifconfig** is that whereas ifconfig only shows enabled interfaces, ip shows all the interfaces whether enabled or disabled.

ifconfig Command

```
$ ifconfig
```

```
tecmint@tecmint ~ $ ifconfig
eth0      Link encap:Ethernet HWaddr 28:d2:44:eb:bd:98
          inet addr:192.168.0.104 Bcast:192.168.0.255 Mask:255.255.255
          inet6 addr: fe80::2ad2:44ff:feeb:bd98/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:342087 errors:0 dropped:0 overruns:0 frame:0
          TX packets:233764 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:406375041 (406.3 MB) TX bytes:25096967 (25.0 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:5146 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5146 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:469809 (469.8 KB) TX bytes:469809 (469.8 KB)

wlan0     Link encap:Ethernet HWaddr 38:b1:db:7c:78:c7
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

tecmint@tecmint ~ $
```

ifconfig: Check IP Address

ip Command

```
$ ip a
```

```
tecmint@tecmint ~ $ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group 0
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group 0
    link/ether 28:d2:44:eb:bd:98 brd ff:ff:ff:ff:ff:ff
        inet 192.168.0.104/24 brd 192.168.0.255 scope global eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::2ad2:44ff:feeb:bd98/64 scope link
            valid_lft forever preferred_lft forever
3: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group 0
    link/ether 38:b1:db:7c:78:c7 brd ff:ff:ff:ff:ff:ff
tecmint@tecmint ~ $
```

ip: Check IP Address

2. Adding or Deleting an IP Address in Linux

The below command assigns the IP address **192.168.80.174** to the interface **eth0**.

ifconfig – Add/Del IP Address

```
# ifconfig eth0 add 192.168.80.174
```

Syntax for adding/removing an interface using ifconfig command:

```
# ifconfig eth0 add 192.168.80.174
# ifconfig eth0 del 192.168.80.174
```

ip – Add/Del IP Address

```
# ip a add 192.168.80.174 dev eth0
```

Syntax for adding/removing an interface using ip command:

```
# ip a add 192.168.80.174 dev eth0
# ip a del 192.168.80.174 dev eth0
```

4. Add MAC Hardware Address to Network Interface

The below command sets the hardware address for the interface `eth0` to the value specified in the command. This can be verified by checking the `HWaddr` value in the output of `ifconfig` command.

ifconfig – Add MAC Address

Here, the syntax for adding MAC address using ifconfig command:

```
# ifconfig eth0 hw ether 00:0c:29:33:4e:aa
```

ip – Add MAC Address

Here, the syntax for adding MAC address using ip command:

```
# ip link set dev eth0 address 00:0c:29:33:4e:aa
```

4. Setting Other Configurations of Network Interface

Apart from setting IP address or Hardware address, other configurations that can be applied to an interface include:

1. MTU (Maximum Transfer Unit)
2. Multicast flag
3. Transmit Queue length
4. Promiscuous mode
5. Enable or disable all multicast mode

ifconfig – Other Network Configurations

ip – Other Network Configurations

a. Set MTU value to 2000.

```
# ifconfig eth0 mtu 2000
# ip link set dev eth0 mtu 2000
```

b. Enable or Disable multicast flag.

```
# ifconfig eth0 multicast
# ip link set dev eth0 multicast on
```

c. Setting the transmit queue length.

```
# ifconfig eth0 txqueuelen 1200  
# ip link set dev eth0 txqueuelen 1200
```

d. Enabling or disabling promiscuous mode.

```
# ifconfig eth0 promisc  
# ip link set dev eth0 promisc on
```

e. Enable or disable all multicast mode.

```
# ifconfig eth0 allmulti  
# ip link set dev eth0 allmulti on
```

5. Enabling or Disabling Network Interface

The below commands enable or disable specific network interface.

ifconfig – Disable/Enable Network Interface

The below command disables the interface `eth0` and it is verified by output of **ifconfig** which by default shows only those interfaces which are up.

```
# ifconfig eth0 down
```

To re-enable the interface, just replace **down** by **up**.

```
# ifconfig eth0 up
```

ip – Disable/Enable Network Interface

The below **ip** command is alternative for ifconfig to disable a specific interface. This can be verified by the output of '`ip a`' command which shows all the interfaces by default, either up or down, but highlights their status along with the description.

```
# ip link set eth0 down
```

To re-enable the interface, just replace **down** with **up**.

```
# ip link set eth0 up
```

6. Enable or disable the use of ARP protocol

The below commands enable or disable ARP protocol on specific network interface.

ifconfig – Enable/Disable ARP Protocol

The command enables ARP protocol to be used with interface **eth0**. To disable this option, just replace arp with **-arp**.

```
# ifconfig eth0 arp
```

ip – Enable/Disable ARP Protocol

This command is the ip alternative to enable ARP for the interface eth0. To disable, just replace **on** with **off**.

```
# ip link set dev eth0 arp on
```

Conclusion

Thus, we have highlighted features of **ifconfig** command and how they can be done using **ip** command. Currently, Linux distributions provides a user with both the commands so that he can use according to his convenience. So, which command is convenient according to you which you would prefer to use? Do mention this in your comments.

If you want to learn more about these two commands, then you should go through our previous articles that shows some practical examples of ifconfig and ip command in more detailed fashion.

15 Useful “ifconfig” Commands to Configure Network Interface in Linux

by [Ravi Saive](#) | Published: January 14, 2013 | Last Updated: February 23, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

ifconfig in short “**interface configuration**” utility for system/network administration in **Unix/Linux** operating systems to configure, manage and query network interface parameters via command line interface or in a system configuration scripts.

The “**ifconfig**” command is used for displaying current network configuration information, setting up an ip address, netmask or broadcast address to an network interface, creating an alias for network interface, setting up hardware address and enable or disable network interfaces.



15 Useful ifconfig Commands

This article covers “**15 Useful “ifconfig” Commands**” with their practical examples, that might be very helpful to you in managing and configuring network interfaces in Linux systems.

Update : The networking command **ifconfig** is deprecated and replaced by [ip command \(Learn 10 Examples of IP Command\)](#) in most Linux distributions.

Don't Miss: [ifconfig vs ip: What's Difference Between Them](#)

1. View All Network Setting

The “**ifconfig**” command with no arguments will display all the active interfaces details. The **ifconfig** command also used to check the assigned IP address of an server.

```
[root@tecmint ~]# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0B:CD:1C:18:5A
inet addr:172.16.25.126 Bcast:172.16.25.63 Mask:255.255.255.224
inet6 addr: fe80::20b:cdff:fe1c:185a/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2341604 errors:0 dropped:0 overruns:0 frame:0
TX packets:2217673 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:293460932 (279.8 MiB) TX bytes:1042006549 (993.7 MiB)
Interrupt:185 Memory:f7fe0000-f7ff0000
lo        Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:5019066 errors:0 dropped:0 overruns:0 frame:0
TX packets:5019066 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:2174522634 (2.0 GiB) TX bytes:2174522634 (2.0 GiB)
tun0      Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:10.1.1.1 P-t-P:10.1.1.2 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

2. Display Information of All Network Interfaces

The following **ifconfig** command with **-a** argument will display information of all active or inactive network interfaces on server. It displays the results for **eth0**, **lo**, **sit0** and **tun0**.

```
[root@tecmint ~]# ifconfig -a
eth0      Link encap:Ethernet HWaddr 00:0B:CD:1C:18:5A
inet addr:172.16.25.126 Bcast:172.16.25.63 Mask:255.255.255.224
inet6 addr: fe80::20b:cdff:fe1c:185a/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2344927 errors:0 dropped:0 overruns:0 frame:0
TX packets:2220777 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:293839516 (280.2 MiB) TX bytes:1043722206 (995.3 MiB)
Interrupt:185 Memory:f7fe0000-f7ff0000
lo        Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:5022927 errors:0 dropped:0 overruns:0 frame:0
TX packets:5022927 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:2175739488 (2.0 GiB) TX bytes:2175739488 (2.0 GiB)
sit0      Link encap:IPv6-in-IPv4
NOARP MTU:1480 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
```

```
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
tun0      Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:10.1.1.1 P-t-P:10.1.1.2 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

3. View Network Settings of Specific Interface

Using interface name (**eth0**) as an argument with “**ifconfig**” command will display details of specific network interface.

```
[root@tecmint ~]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:0B:CD:1C:18:5A
inet addr:172.16.25.126 Bcast:172.16.25.63 Mask:255.255.255.224
inet6 addr: fe80::20b:cdff:fe1c:185a/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2345583 errors:0 dropped:0 overruns:0 frame:0
TX packets:2221421 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:293912265 (280.2 MiB) TX bytes:1044100408 (995.7 MiB)
Interrupt:185 Memory:f7fe0000-f7ff0000
```

4. How to Enable an Network Interface

The “**up**” or “**ifup**” flag with interface name (**eth0**) activates an network interface, if it is not in active state and allowing to send and receive information. For example, “**ifconfig eth0 up**” or “**ifup eth0**” will activate the **eth0** interface.

```
[root@tecmint ~]# ifconfig eth0 up
OR
[root@tecmint ~]# ifup eth0
```

5. How to Disable an Network Interface

The “**down**” or “**ifdown**” flag with interface name (**eth0**) deactivates the specified network interface. For example, “**ifconfig eth0 down**” or “**ifdown eth0**” command deactivates the **eth0** interface, if it is in active state.

```
[root@tecmint ~]# ifconfig eth0 down
OR
[root@tecmint ~]# ifdown eth0
```

6. How to Assign a IP Address to Network Interface

To assign an IP address to an specific interface, use the following command with an interface name (**eth0**) and ip address that you want to set. For example, “**ifconfig eth0 172.16.25.125**” will set the IP address to interface **eth0**.

```
[root@tecmint ~]# ifconfig eth0 172.16.25.125
```

7. How to Assign a Netmask to Network Interface

Using the “**ifconfig**” command with “**netmask**” argument and interface name as (**eth0**) allows you to define an netmask to an given interface. For example, “**ifconfig eth0 netmask 255.255.255.224**” will set the network mask to an given interface **eth0**.

```
[root@tecmint ~]# ifconfig eth0 netmask 255.255.255.224
```

8. How to Assign a Broadcast to Network Interface

Using the “**broadcast**” argument with an interface name will set the broadcast address for the given interface. For example, “**ifconfig eth0 broadcast 172.16.25.63**” command sets the broadcast address to an interface **eth0**.

```
[root@tecmint ~]# ifconfig eth0 broadcast 172.16.25.63
```

9. How to Assign a IP, Netmask and Broadcast to Network Interface

To assign an IP address, Netmask address and Broadcast address all at once using “**ifconfig**” command with all arguments as given below.

```
[root@tecmint ~]# ifconfig eth0 172.16.25.125 netmask 255.255.255.224  
broadcast 172.16.25.63
```

10. How to Change MTU for an Network Interface

The “**mtu**” argument set the maximum transmission unit to an interface. The **MTU** allows you to set the limit size of packets that are transmitted on an interface. The **MTU** able to handle maximum number of octets to an interface in one single transaction. For example, “**ifconfig eth0 mtu 1000**” will set the maximum transmission unit to given set (i.e. **1000**). Not all network interfaces supports **MTU** settings.

```
[root@tecmint ~]# ifconfig eth0 mtu 1000
```

11. How to Enable Promiscuous Mode

What happens in normal mode, when a packet received by a network card, it verifies that the packet belongs to itself. If not, it drops the packet normally, but in the promiscuous mode is used to accept all the packets that flows through the network card.

Most of the today's network tools uses the promiscuous mode to capture and analyze the packets that flows through the network interface. To set the promiscuous mode, use the following command.

```
[root@tecmint ~]# ifconfig eth0 promisc
```

12. How to Disable Promiscuous Mode

To disable promiscuous mode, use the “**-promisc**” switch that drops back the network interface in normal mode.

```
[root@tecmint ~]# ifconfig eth0 -promisc
```

13. How to Add New Alias to Network Interface

The **ifconfig** utility allows you to configure additional network interfaces using **alias** feature. To add alias network interface of **eth0**, use the following command. Please note that alias network address in same sub-net mask. For example, if your **eth0** network ip address is **172.16.25.125**, then alias ip address must be **172.16.25.127**.

```
[root@tecmint ~]# ifconfig eth0:0 172.16.25.127
```

Next, verify the newly created alias network interface address, by using “**ifconfig eth0:0**” command.

```
[root@tecmint ~]# ifconfig eth0:0
eth0:0      Link encap:Ethernet  HWaddr 00:01:6C:99:14:68
inet addr:172.16.25.123  Bcast:172.16.25.63  Mask:255.255.255.240
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:17
```

14. How to Remove Alias to Network Interface

If you no longer required an alias network interface or you incorrectly configured it, you can remove it by using the following command.

```
[root@tecmint ~]# ifconfig eth0:0 down
```

15. How to Change the MAC address of Network Interface

To change the **MAC (Media Access Control)** address of an **eth0** network interface, use the following command with argument “**hw ether**”. For example, see below.

```
[root@tecmint ~]# ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF
```

These are the most useful commands for configuring network interfaces in **Linux**, for more information and usage of **ifconfig** command use the manpages like “**man ifconfig**” at the terminal. Check out some other networking utilities below.

Other Networking Utilities

1. [Tcpdump](#) — is an command-line packet capture and analyzer tool for monitoring network traffic.
2. [Netstat](#) — is an open source command line network monitoring tool that monitors incoming and outgoing network packets traffic.
3. [Wireshark](#) — is an open source network protocol analyzer that is used to troubleshoot network related issues.
4. [Munin](#) — is an web based network and system monitoring application that is used to display results in graphs using rrdtool.
5. [Cacti](#) — is an complete web based monitoring and graphing application for network monitoring.

To get more information and options for any of the above tools, see the manpages by entering “**man toolname**” at the command prompt. For example, to get the information for “**netstat**” tool, use the command as “**man netstat**“.

10 Useful “IP” Commands to Configure Network Interfaces

by [Ravi Saive](#) | Published: March 12, 2013 | Last Updated: February 23, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this post, we are going to review how we can assign **Static IP Address**, **Static Route**, **Default Gateway** etc. Assigning IP Address on demand using **IP** command. [IFCONFIG command](#) is deprecated and replaced by **IP** command in **Linux**. However, **IFCONFIG** command is still works and available for most of the Linux distributions.

Don't Miss: [ifconfig vs ip: What's Difference and Comparing Commands](#)



10 IP Command Examples

Note: Please take configuration file backup before doing any changes.

How do i Configure Static IP Address Internet Protocol (IPv4)

To configure static IP Address, you need to update or edit network configuration file to assign an Static IP Address to a system. You must be superuser with **su** (**switch user**) command from terminal or command prompt.

For RHEL/CentOS/Fedora

Open and edit network configuration file for (**eth0** or **eth1**) using your favorite editor. For example, to assigning IP Address to **eth0** interface as follows.

```
[root@tecmint ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

Simple output:

```
DEVICE="eth0"
BOOTPROTO=static
ONBOOT=yes
TYPE="Ethernet"
IPADDR=192.168.50.2
NAME="System eth0"
HWADDR=00:0C:29:28:FD:4C
GATEWAY=192.168.50.1
```

For Ubuntu/Debian/Linux Mint

Assign Static IP Address to **eth0** interface editing configuration file **/etc/network/interfaces** to make permanent changes as shown below.

```
auto eth0
iface eth0 inet static
address 192.168.50.2
netmask 255.255.255.0
gateway 192.168.50.1
```

Next, restart network services after entering all the details using the following command.

```
# /etc/init.d/networking restart
$ sudo /etc/init.d/networking restart
```

1. How to Assign a IP Address to Specific Interface

The following command used to assign IP Address to a specific interface (**eth1**) on the fly.

```
# ip addr add 192.168.50.5 dev eth1
$ sudo ip addr add 192.168.50.5 dev eth1
```

Note: Unfortunately all these settings will be lost after a system restart.

2. How to Check an IP Address

To get the depth information of your network interfaces like IP Address, MAC Address information, use the following command as shown below.

```
# ip addr show
$ sudo ip addr show
```

Sample Output

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
```

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
link/ether 00:0c:29:28:fd:4c brd ff:ff:ff:ff:ff:ff
inet 192.168.50.2/24 brd 192.168.50.255 scope global eth0
inet6 fe80::20c:29ff:fe28:fd4c/64 scope link
valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
link/ether 00:0c:29:28:fd:56 brd ff:ff:ff:ff:ff:ff
inet 192.168.50.5/24 scope global eth1
inet6 fe80::20c:29ff:fe28:fd56/64 scope link
valid_lft forever preferred_lft forever
```

3. How to Remove an IP Address

The following command will remove an assigned IP address from the given interface (**eth1**).

```
# ip addr del 192.168.50.5/24 dev eth1
$ sudo ip addr del 192.168.50.5/24 dev eth1
```

4. How to Enable Network Interface

The “**up**” flag with interface name (**eth1**) enables a network interface. For example, the following command will activates the **eth1** network interface.

```
# ip link set eth1 up
$ sudo ip link set eth1 up
```

5. How to Disable Network Interface

The “**down**” flag with interface name (**eth1**) disables a network interface. For example, the following command will De-activates the **eth1** network interface.

```
# ip link set eth1 down
$ sudo ip link set eth1 down
```

6. How do I Check Route Table?

Type the following command to check the routing table information of system.

```
# ip route show
$ sudo ip route show
```

Sample Output

```
10.10.20.0/24 via 192.168.50.100 dev eth0
192.168.160.0/24 dev eth1 proto kernel scope link src 192.168.160.130
metric 1
192.168.50.0/24 dev eth0 proto kernel scope link src 192.168.50.2
169.254.0.0/16 dev eth0 scope link metric 1002
default via 192.168.50.1 dev eth0 proto static
```

7. How do I Add Static Route

Why you need to add Static routes or Manual routes, because that the traffic must not pass through the default gateway. We need to add Static routes to pass traffic from best way to reach the destination.

```
# ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0
$ sudo ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0
```

8. How to Remove Static Route

To remove assigned static route, simply type the following command.

```
# ip route del 10.10.20.0/24
$ sudo ip route del 10.10.20.0/24
```

9. How do I Add Persistence Static Routes

All the above route will be lost after a system restart. To add permanent Static route, edit file **/etc/sysconfig/network-scripts/route-eth0** (We are storing static route for **(eth0)**) and add the following lines and save and exist. By default **route-eth0** file will not be there, need to be created.

For RHEL/CentOS/Fedora

```
# vi /etc/sysconfig/network-scripts/route-eth0
10.10.20.0/24 via 192.168.50.100 dev eth0
```

For Ubuntu/Debian/Linux Mint

Open the file **/etc/network/interfaces** and at the end add the persistence Static routes. IP Addresses may differ in your environment.

```
$ sudo vi /etc/network/interfaces
auto eth0
iface eth0 inet static
address 192.168.50.2
netmask 255.255.255.0
gateway 192.168.50.100
#####{Static Route}#####
up ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0
```

Next, restart network services after entering all the details using the following command.

```
# /etc/init.d/network restart
$ sudo /etc/init.d/network restart
```

10. How do I Add Default Gateway

Default gateway can be specified globally or for in interface-specific config file. Advantage of default gateway is If we have more than one NIC is present in the system. You can add default gateway on the fly as shown below command.

```
# ip route add default via 192.168.50.100  
$ sudo ip route add default via 192.168.50.100
```

Kindly correct me if i missed out. Please refer manual page doing **man ip** from terminal/command prompt to know more about IP Command.

8 Linux ‘Parted’ Commands to Create, Resize and Rescue Disk Partitions

by [Marin Todorov](#) | Published: February 1, 2016 | February 1, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Parted is a famous command line tool that allows you to easily manage hard disk partitions. It can help you add, delete, shrink and extend disk partitions along with the file systems located on them. Parted has gone a long way from when it first came out. Some of its functions have been removed, others have been added.



8 'Parted' Commands to Manage Linux Disks

Parted Command to Manage Linux Disk Partitions

In this tutorial you will learn the basics of parted and we will show you some practical examples. If you don't have any previous experience with parted, please be aware that parted writes the changes immediately to your disk, so be careful if you try to modify your disk partitions.

If you plan on testing parted, the better option would be to simply use a virtual machine or old computer/laptop without any valuable information on it. To make modifications on a disk partition it must not be in use. If you need to work on primary partition, you may boot into rescue mode.

Note: You will need to have root access to the machine you will be working on in order to use parted.

How to Install Parted on Linux

On many Linux distributions, **parted** comes pre-installed. If it is not included in your distro, you can install it with:

```
$ sudo apt-get install parted          [On Debian/Ubuntu systems]
# yum install parted                 [On RHEL/CentOS and Fedora]
# dnf install parted                [On Fedora 22+ versions]
```

Once you have make sure that **parted** is installed, you can proceed further to check out some real world examples of parted command in the rest of this article.

1. Check Parted Version

Run the following command, you see message similar to the one shown on the image below. Don't worry if your parted version is different. Unless specified otherwise, parted will use your primary drive, which in most cases will be `/dev/sda`.

```
$ parted
[root@TecMint $ parted
GNU Parted 3.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list o
(parted) [
```

Check Parted Command Version

If you want to exit parted, simply type:

```
$ quit
```

2. List Linux Disk Partitions

Now that parted is started, let's list the partitions of the selected hard disk. As mentioned earlier, parted chooses your first drive by default. To see the disk partitions run `print`.

```
(parted) print
```

```
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Partition Flags:
```

Number	Start	End	Size	Type	File system
1	1049kB	525MB	524MB	primary	xfs
2	525MB	53.7GB	53.2GB	primary	

```
(parted) █
```

Check Linux Partitions

When running `print`, it will also display the hard disk information and model. Here is example from a real hard disk (not virtual as shown on the image above) :

```
(parted) print
Model: ATA TOSHIBA MQ01ACF0 (scsi)
Disk /dev/sda: 320GB
Sector size (logical/physical): 512B/4096B
Partition Table: msdos
Number  Start   End     Size    Type      File system  Flags
 1      1049kB  256MB  255MB  primary   ext2        boot
 2      257MB   320GB  320GB  extended
 5      257MB   320GB  320GB  logical          lvm
```

In the example above, you can see the disk model, capacity sector size and partition table.

3. List or Switch to Different Disk

If you have more than one hard disk, you can easily switch between disks, by using the “`select`” command. In the example below, I will switch from `/dev/sda` to `/dev/sdb` which is a secondary drive on my system.

To easily switch between disks you can use:

```
(parted) select /dev/sdX
```

```
(parted) select /dev/sdb
Using /dev/sdb
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 32.2GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:
```

Number	Start	End	Size	File system	Flags
1	0.00B	32.2GB	32.2GB	ext4	

```
(parted) █
```

Select Different Disk

Change "x" with the letter of the disk to which you wish to switch.

4. Create Primary or Logical Partition in Linux

Parted can be used to create primary and logical disk partitions. In this example, I will show you how to create primary partition, but the steps are the same for logical partitions.

To create new partition, parted uses “`mkpart`”. You can give it additional parameters like “`primary`” or “`logical`” depending on the partition type that you wish to create.

Before you start creating partitions, it's important to make sure that you are using (you have selected) the right disk.

Start by using `print`:

```
(parted) print
```

```
(parted) print
Error: /dev/sdb: unrecognised disk label
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 34.4GB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
(parted) 
```

Show Current Linux Disk

As shown on the above image, we are using a virtual drive of **34 GB**. First we will give the new disk a label and then create a partition and set a file system on it.

Now the first step is to give the new [disk a label name](#) with:

```
(parted) mklabel msdos
```

Now create the new partition with `mkpart`. The listed units are in megabytes (**MB**). We will create a **10 GB** partition starting from **1** to **10000**:

```
(parted) mkpart
Partition type? primary/extended? primary
File system type? [ext2]?
Start? 1
End? 10000
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 34.4GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
Number  Start   End     Size    Type      File system  Flags
1        1049kB  10.0GB  9999MB  primary    ext2        lba
```

```
(parted) mkpart ←
Partition type? primary/extended? primary ←
File system type? [ext2]?
Start? 1 ←
End? 10000 ←
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 34.4GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
```

Number	Start	End	Size	Type	File system
1	1049kB	10.0GB	9999MB	primary	ext2

Create Primary or Logical Linux Partitions

Next, exit parted with "quit" command. We will format our new partition in ext4 file system using `mkfs`. To make this happen run the following command:

```
# mkfs.ext4 /dev/sdb1
```

Note: It's important to select the right disk and partition when executing the above command!

Now let's verify our results, by printing the partition table on our secondary disk. Under file system column, you should see ext4 or the file system type that you have decided to use for your partition:

```
root@TecMint:~# parted
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of
(parted) select /dev/sdb
sdb   sdb1
(parted) select /dev/sdb
Using /dev/sdb
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 34.4GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Partition Flags:
Number  Start    End      Size     Type      File system
 1       1049kB  10.0GB   9999MB  primary   ext4
(parted)
```

Verify Disk Partition Filesystem

5. Resize Linux Disk Partition

Parted includes multiple useful functions and one of them is "resizepart". As you have probably figured this out by now, "resizepart" helps you resize a partition.

In the example below, you will see how to resize an existing partition. For the purpose of this example, we will be using the earlier created partition.

First you will need to know the number of the partition that you will be resizing. This can be easily found by using "print":

```
(parted) print
```

```
[parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 34.4GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Partition Flags:

  Number  Start    End     Size   Type      File system
    1      1049kB  10.0GB  9999MB primary   ext4

(parted) 
```

Find Linux Partition Number

In our example, the partition number is "1". Now run the resizepart command:

```
(parted) resizepart
```

You will be asked for the number of the partition that you will resize. Enter it's number. After that, you will be asked to set the new ending point for this partition. Remember that by default the units are in **MB**. In our example, we have set the new partition size to **15 GB**:

```
(parted) resizepart
Partition number? 1
End? [10.0GB]? 15000
```

Now verify the results with "print":

```
(parted) print
```

```
[parted] resizepart  
[Partition number? 1  
[End? [10.0GB]? 15000  
[(parted) print  
Model: ATA VBOX HARDDISK (scsi)  
Disk /dev/sdb: 34.4GB  
Sector size (logical/physical): 512B/512B  
Partition Table: msdos  
Disk Flags:
```

Number	Start	End	Size	Type	File system
1	1049kB	15.0GB	15.0GB	primary	ext4

Verify Linux Resize Partition

6. Delete Linux Partition

The next thing you will learn is how to delete a partition from your hard drive. To do this, you will need to use the "rm" command within parted. To delete a disk partition you will need to know it's number.

As mentioned earlier, you can easily obtain this number by using "print". In our example, we will delete the partition with number 1 from our secondary drive /dev/sdb1:

```
(parted) rm 1
```

Verify the results by printing the partitions table:

```
[parted) print
Model: Unknown (unknown)
Disk /dev/sdb1: 15.0GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:
```

Number	Start	End	Size	File system	Flags
1	0.00B	15.0GB	15.0GB	ext4	

```
[parted) rm 1
[parted) print
Model: Unknown (unknown)
Disk /dev/sdb1: 15.0GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:
```

Number	Start	End	Size	File system	Flags
--------	-------	-----	------	-------------	-------

```
(parted) █
```

Delete a Linux Partition

7. Rescue Linux Disk Partition

Parted supports a “rescue” utility that helps you recover a lost partition between a starting and ending point. If a partition is found within that range, it will attempt to restore it.

Here is an example:

```
(parted) rescue
```

```
Start? 1
End? 15000
(parted) print
Model: Unknown (unknown)
Disk /dev/sdb1: 15.0GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:
Number Start End Size File system Flags
1 0.00B 15.0GB 15.0GB ext4
```

8 Change Linux Partition Flag

Using parted, you can change the state of a flag for disk partitions. The supported flags are:

1. boot
2. root
3. swap
4. hidden
5. raid
6. lvm
7. lba
8. legacy_boot
9. irst
10. esp
11. palo

The states can be either "on" or "off". To change a flag simply run "set" command within parted:

```
(parted) set 2 lba on
```

The above command sets lba flag to on for second partition. Verify the results with print:

```
[(parted) set 2 lba on
[(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 34.4GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start    End      Size     Type   File system
 1       1049kB  15.0GB   15.0GB   primary ext4
 2       15.0GB   16.0GB   1000MB  primary
```

(parted) █

Change Partition Flag

Conclusion

Parted is a useful and powerful utility that can help you manage your disk partitions in Linux systems. As always, when working with disk partitions you need to be extra careful. It is strongly recommended to go through parted man pages to learn how you can customize its output and find more information about its capabilities.

If you have any questions or comments, please do not hesitate to use the comment section below.

How to Set and Unset Local, User and System Wide Environment Variables in Linux

by [Gunjit Khera](#) | Published: January 23, 2016 | Last Updated: January 23, 2016

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Environment Variables are some special variables that are defined in shell and are needed by programs while execution. They can be system defined or user defined. System defined variables are those which are set by system and are used by system level programs.



The image is a screenshot of a web page from Tecmint.com. It features a dark red header bar. In the center is a white illustration of the Tux penguin, the official mascot of Linux. To the right of the penguin is the Tecmint logo, which includes a blue square icon with a white arrow pointing down and to the left, followed by the word "Tecmint" in a bold, white, sans-serif font, with ".com" in a smaller font below it. Below the logo, the text "Linux Howto's Guide" is written in a smaller, white, sans-serif font. The main title of the article, "Learn about Local, User and System Wide Environment Variables in Linux", is displayed in a large, white, sans-serif font just below the header. The main content area has a dark red background and contains the article's title in a large, white, sans-serif font.

Set and Unset Linux Environment Variables

For e.g. [PWD](#) command is a very common system variable which is used to store the present working directory. User defined variables are typically set by user, either temporarily for the current shell or permanently. The whole concept of setting and un-setting environment variables revolves around some set of files and few commands and different shells.

In Broader terms, an environment variable can be in three types:

1. Local Environment Variable

One defined for the current session. These environment variables last only till the current session, be it remote login session, or local terminal session. These variables are not specified in any configuration files and are created, and removed by using a special set of commands.

2. User Environment Variable

These are the variables which are defined for a particular user and are loaded every time a user logs in using a local terminal session or that user is logged in using remote login session. These variables are typically set in and loaded from following configuration files: `.bashrc`, `.bash_profile`, `.bash_login`, `.profile` files which are present in user's home directory.

3. System wide Environment Variables

These are the environment variables which are available system-wide, i.e. for all the users present on that system. These variables are present in system-wide configuration files present in following directories and files: `/etc/environment`, `/etc/profile`, `/etc/profile.d/`, `/etc/bash.bashrc`. These variables are loaded every time system is powered on and logged in either locally or remotely by any user.

Understanding User-Wide and System-wide Configuration files

Here, we briefly describe various configuration files listed above that hold Environment Variables, either system wide or user specific.

.bashrc

This file is user specific file that gets loaded each time user creates a new local session i.e. in simple words, opens a new terminal. All environment variables created in this file would take effect every time a new local session is started.

.bash_profile

This file is user specific remote login file. Environment variables listed in this file are invoked every time the user is logged in remotely i.e. using ssh session. If this file is not present, system looks for either `.bash_login` or `.profile` files.

/etc/environment

This file is system wide file for creating, editing or removing any environment variables. Environment variables created in this file are accessible all throughout the system, by each and every user, both locally and remotely.

/etc/bash.bashrc

System wide `bashrc` file. This file is loaded once for every user, each time that user opens a local terminal session. Environment variables created in this file are accessible for all users but

only through local terminal session. When any user on that machine is accessed remotely via a remote login session, these variables would not be visible.

/etc/profile

System wide profile file. All the variables created in this file are accessible by every user on the system, but only if that user's session is invoked remotely, i.e. via remote login. Any variable in this file will not be accessible for local login session i.e. when user opens a new terminal on his local system.

Note: Environment variables created using **system-wide** or **user-wide** configuration files can be removed by removing them from these files only. Just that after each change in these files, either log out and log in again or just type following command on the terminal for changes to take effect:

```
$ source <file-name>
```

Set or Unset Local or Session-wide Environment Variables in Linux

Local Environment Variables can be created using following commands:

```
$ var=value  
OR  
$ export var=value
```

These variables are session wide and are valid only for current terminal session. To Clear these session-wide environment variables following commands can be used:

1. Using env

By default, "env" command lists all the current environment variables. But, if used with '-i' switch, it temporarily clears out all the environment variables and lets user execute a command in current session in absence of all the environment variables.

```
$ env -i [Var=Value]... command args...
```

Here, `var=value` corresponds to any local environment variable that you want to use with this command only.

```
$ env -i bash
```

Will give bash shell which temporarily would not have any of the environment variable. But, as you exit from the shell, all the variables would be restored.

2. Using unset

Another way to clear local environment variable is by using unset command. To unset any local environment variable temporarily,

```
$ unset <var-name>
```

Where, `var-name` is the name of local variable which you want to un-set or clear.

3. Set the variable name to ”

Another less common way would be to set the name of the variable which you want to clear, to '' (Empty). This would clear the value of the local variable for current session for which it is active.

NOTE – YOU CAN EVEN PLAY WITH AND CHANGE THE VALUES OF SYSTEM OR USER ENVIRONMENT VARIABLES, BUT CHANGES WOULD REFLECT IN CURRENT TERMINAL SESSION ONLY AND WOULD NOT BE PERMANENT.

Learn How to Create, User-Wide and System-Wide Environment Variables in Linux

In section, we will going to learn how to set or unset local, user and system wide environment variables in Linux with below examples:

1. Set and Unset Local Variables in Linux

a.) Here, we create a local variable `VAR1` and set it to any value. Then, we use `unset` to remove that local variable, and at the end that variable is removed.

```
$ VAR1='TecMint is best Site for Linux Articles'  
$ echo $VAR1  
$ unset VAR1  
$ echo $VAR1  
tecmint@tecmint ~ $ VAR1='TecMint is best Site for Linux Articles'  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ echo $VAR1  
TecMint is best Site for Linux Articles  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ unset VAR1  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ echo $VAR1  
  
tecmint@tecmint ~ $ □
```

Set Unset Local Environment Variables

b.) Another way of creating a local variable is by using `export` command. The local variable created will be available for current session. To unset the variable simply set the value of variable to ''.

```
$ export VAR='TecMint is best Site for Linux Articles'  
$ echo $VAR  
$ VAR=  
$ echo $VAR  
tecmint@tecmint ~ $ export VAR='TecMint is best Site for Linux Art  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ echo $VAR  
TecMint is best Site for Linux Articles  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ VAR=  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ echo $VAR  
  
tecmint@tecmint ~ $ □
```

Export Local Environment Variables in Linux

c.) Here, we created a local variable `VAR2` and set it to a value. Then in-order to run a command temporarily clearing out all local and other environment variables, we executed '`env -i`' command. This command here executed bash shell by clearing out all other environment variables. After entering '`exit`' on the invoked bash shell, all variables would be restored.

```
$ VAR2='TecMint is best Site for Linux Articles'  
$ echo $VAR2  
$ env -i bash  
$ echo $VAR2  
tecmint@tecmint ~ $ VAR2='TecMint is best Site for Linux Articles  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ echo $VAR2  
TecMint is best Site for Linux Articles  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ env -i bash  
tecmint@tecmint /home/tecmint $  
tecmint@tecmint /home/tecmint $ echo $VAR2  
  
tecmint@tecmint /home/tecmint $ □
```

Use Env Command to Unset Variables

2. Set and Unset User-Wide Environment Variables in Linux

a.) Modify `.bashrc` file in your home directory to export or set the environment variable you need to add. After that **source** the file, to make the changes take effect. Then you would see the variable ('`CD`' in my case), taking effect. This variable will be available every time you open a new terminal for this user, but not for remote login sessions.

```
$ vi .bashrc
```

Add the following line to `.bashrc` file at the bottom.

```
export CD='This is TecMint Home'
```

Now run the following command to take new changes and test it.

```
$ source .bashrc
$ echo $CD
tecmint@tecmint ~ $ vi .bashrc
tecmint@tecmint ~ $ cat .bashrc
#powerline-daemon -q
#POWERLINE_BASH_CONTINUATION=1
#POWERLINE_BASH_SELECT=1
#. /usr/local/lib/python2.7/dist-packages/powerline/bindings/bash,

#export PS1="\e[0;34m\e[47m\u@\h.com \w>\e[m "
export CD='This is TecMint Home'
tecmint@tecmint ~ $ source .bashrc
tecmint@tecmint ~ $
tecmint@tecmint ~ $ echo $CD
This is TecMint Home
tecmint@tecmint ~ $ 
```

Set User-Wide Environment Variables in Linux

To remove this variable, just remove the following line in `.bashrc` file and re-source it:

b.) To add a variable which will be available for remote login sessions (i.e. when you ssh to the user from remote system), modify `.bash_profile` file.

```
$ vi .bash_profile
```

Add the following line to `.bash_profile` file at the bottom.

```
export VAR2='This is TecMint Home'
```

When on sourcing this file, the variable will be available when you ssh to this user, but not on opening any new local terminal.

```
$ source .bash_profile  
$ echo $VAR2
```

Here, VAR2 is not initially available but, on doing ssh to user on localhost, the variable becomes available.

```
$ ssh tecmint@localhost  
$ echo $VAR2  
tecmint@tecmint ~ $ ssh tecmint@localhost  
tecmint@localhost's password:  
Welcome to Linux Mint 17.3 Rosa (GNU/Linux 3.19.0-32-generic x86_64)  
  
Welcome to Linux Mint  
* Documentation: http://www.linuxmint.com  
  
51 packages can be updated.  
0 updates are security updates.  
  
Last login: Sat Jan 23 12:37:41 2016 from localhost  
tecmint@tecmint ~ $  
tecmint@tecmint ~ $ echo $VAR2  
This is TecMint Home  
tecmint@tecmint ~ $ □
```

Export User Wide Variables in Bash Profile

To remove this variable, just remove the line in `.bash_profile` file which you added, and re-source the file.

NOTE: These variables will be available every time you are logged in to current user but not for other users.

3. Set and Unset System-Wide Environment Variables in Linux

a.) To add system wide no-login variable (i.e. one which is available for all users when any of them opens new terminal but not when any user of machine is remotely accessed) add the variable to `/etc/bash.bashrc` file.

```
export VAR='This is system-wide variable'
```

```
# enable bash completion in interactive shells
if [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
fi

# if the command-not-found package is installed, use it
if [ -x /usr/lib/command-not-found ]; then
    function command_not_found_handle {
        # check because c-n-f could've been removed in the meantime
        if [ -x /usr/lib/command-not-found ]; then
            /usr/bin/python /usr/lib/command-not-found -- $command_name
            return $?
        else
            return 127
        fi
    }
fi

/usr/bin/mint-fortune

export VAR='This is system-wide variable'
-- INSERT --
99
```

Add System Wide Environment Variables

After that, source the file.

```
$ source /etc/bash.bashrc
```

Now this variable will be available for every user when he opens any new terminal.

```
$ echo $VAR
$ sudo su
$ echo $VAR
$ su -
$ echo $VAR
```

```
tecmint@tecmint ~ $ echo $VAR
This is system-wide variable
tecmint@tecmint ~ $
tecmint@tecmint ~ $
tecmint@tecmint ~ $
tecmint@tecmint ~ $ sudo su
tecmint tecmint #
tecmint tecmint # echo $VAR
This is system-wide variable
tecmint tecmint #
tecmint tecmint # su -
tecmint ~ # whoami
root
tecmint ~ # echo $VAR
This is system-wide variable
tecmint ~ # 
```

Check System Wide Variables

Here, same variable is available for **root** user as well as normal user. You can verify this by logging in to other user.

b.) If you want any environment variable to be available when any of the user on your machine is remotely logged in, but not on opening any new terminal on local machine, then you need to edit the file – '/etc/profile'.

```
export VAR1='This is system-wide variable for only remote sessions'
```

```
if [ "`id -u`" -eq 0 ]; then
    PS1='# '
else
    PS1='${debian_chroot:+($debian_chroot)}$ '
fi
fi

# The default umask is now handled by pam_umask.
# See pam_umask(8) and /etc/login.defs.

if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh; do
        if [ -r $i ]; then
            . $i
        fi
    done
    unset i
fi

export VAR1='This is system-wide variable for only remote sessions'
```

INSERT → /etc/profile +
-- INSERT --

unix < utf-8 < sh <

Add System Wide Variables in Profile

After adding the variable, just re-source the file. Then the variable would be available.

```
$ source /etc/profile
$ echo $VAR1
```

To remove this variable, remove the line from /etc/profile file and re-source it.

c.) However, if you want to add any environment which you want to be available all throughout the system, on both remote login sessions as well as local sessions(i.e. opening a new terminal window) for all users, just export the variable in **/etc/environment** file.

```
export VAR12='I am available everywhere'
```

Add System Variable in Environment File

After that just source the file and the changes would take effect.

```
$ source /etc/environment  
$ echo $VAR12  
$ sudo su  
$ echo $VAR12  
$ exit  
$ ssh localhost  
$ echo $VAR12
```

```
tecmint@tecmint ~ $ vi /etc/environment
tecmint@tecmint ~ $ source /etc/environment
tecmint@tecmint ~ $
tecmint@tecmint ~ $ echo $VAR12
I am available everywhere
tecmint@tecmint ~ $
tecmint@tecmint ~ $ sudo su
[sudo] password for tecmint:
tecmint tecmint # echo $VAR12
I am available everywhere
tecmint tecmint #
tecmint tecmint # exit
exit
tecmint@tecmint ~ $ ssh localhost
tecmint@localhost's password:
Welcome to Linux Mint 17.3 Rosa (GNU/Linux 3.19.0-32-generic x86_64)

Welcome to Linux Mint
 * Documentation:  http://www.linuxmint.com

51 packages can be updated.
0 updates are security updates.

Last login: Sat Jan 23 12:39:11 2016 from localhost
tecmint@tecmint ~ $ echo $VAR12
I am available everywhere
tecmint@tecmint ~ $ |
```

Check Environment Variable for All Users

Here, as we see the environment variable is available for normal user, root user, as well as on remote login session (here, to **localhost**).

To clear out this variable, just remove the entry in the **/etc/environment** file and re-source it or login again.

NOTE: Changes take effect when you source the file. But, if not then you might need to log out and log in again.

Conclusion

Thus, these are few ways we can modify the environment variables. If you find any new and interesting tricks for the same do mention in your comments.

30 Useful Linux Commands for System Administrators

by [Ravi Saive](#) | Published: December 22, 2012 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this article we are going to review some of the useful and frequently used **Linux or Unix** commands for **Linux System Administrators** that are used in their daily life. This is not a complete but it's a compact list of commands to refer when needed. Let us start one by one how we can use those commands with examples.



30 Useful Linux System Administration Commands

1. Uptime Command

In Linux **uptime** command shows since how long your system is running and the number of users are currently logged in and also displays load average for **1,5** and **15** minutes intervals.

```
# uptime
08:16:26 up 22 min,  1 user,  load average: 0.00, 0.03, 0.22
```

Check Uptime Version

Uptime command don't have other options other than **uptime** and **version**. It gives information only in **hours:mins** if it less than **1** day.

```
[tecmint@tecmint ~]$ uptime -V
procps version 3.2.8
```

2. W Command

It will displays users currently logged in and their process along-with shows **load averages**. also shows the **login name**, **tty name**, **remote host**, **login time**, **idle time**, **JCPU**, **PCPU**, command and processes.

```
# w
08:27:44 up 34 min, 1 user, load average: 0.00, 0.00, 0.08
USER      TTY      FROM           LOGIN@     IDLE     JCPU     PCPU WHAT
tecmint   pts/0    192.168.50.1    07:59     0.00s   0.29s   0.09s w
```

Available options

1. **-h** : displays no header entries.
2. **-s** : without JCPU and PCPU.
3. **-f** : Removes from field.
4. **-V** : (upper letter) – Shows versions.

3. Users Command

Users command displays currently logged in users. This command don't have other parameters other than help and version.

```
# users
tecmint
```

4. Who Command

who command simply return **user name**, **date**, **time** and **host information**. who command is similar to **w** command. Unlike **w** command **who** doesn't print what users are doing. Lets illustrate and see the different between **who** and **w** commands.

```
# who
tecmint  pts/0          2012-09-18 07:59 (192.168.50.1)
# w
08:43:58 up 50 min, 1 user, load average: 0.64, 0.18, 0.06
USER      TTY      FROM           LOGIN@     IDLE     JCPU     PCPU WHAT
tecmint   pts/0    192.168.50.1    07:59     0.00s   0.43s   0.10s w
```

Who command Options

1. **-b** : Displays last system reboot date and time.
2. **-r** : Shows current runlet.
3. **-a, -all** : Displays all information in cumulatively.

5. Whoami Command

whoami command print the name of current user. You can also use “**who am i**” command to display the current user. If you are logged in as a root using sudo command “**whoami**” command

return **root** as current user. Use “**who am i**” command if you want to know the exact user logged in.

```
# whoami  
tecmint
```

6. ls Command

ls command display list of files in human readable format.

```
# ls -l  
total 114  
dr-xr-xr-x.    2 root root  4096 Sep 18 08:46 bin  
dr-xr-xr-x.    5 root root  1024 Sep  8 15:49 boot
```

Sort file as per last modified time.

```
# ls -ltr  
total 40  
-rw-r--r--. 1 root root  6546 Sep 17 18:42 install.log.syslog  
-rw-r--r--. 1 root root 22435 Sep 17 18:45 install.log  
-rw-------. 1 root root  1003 Sep 17 18:45 anaconda-ks.cfg
```

For more examples of **ls** command, please check out our article on [15 Basic ‘ls’ Command Examples in Linux](#).

7. Crontab Command

List schedule jobs for current user with **crontab** command and **-l** option.

```
# crontab -l  
00 10 * * * /bin/ls >/ls.txt
```

Edit your **crontab** with **-e** option. In the below example will open schedule jobs in **VI editor**. Make a necessary changes and quit pressing **:wq** keys which saves the setting automatically.

```
# crontab -e
```

For more examples of **Linux Cron Command**, please read our earlier article on [11 Cron Scheduling Task Examples in Linux](#).

8. Less Command

less command allows quickly view file. You can page up and down. Press ‘**q**’ to quit from less window.

```
# less install.log  
Installing setup-2.8.14-10.el6.noarch
```

```
warning: setup-2.8.14-10.el6.noarch: Header V3 RSA/SHA256 Signature, key ID  
c105b9de: NOKEY  
Installing filesystem-2.4.30-2.1.el6.i686  
Installing ca-certificates-2010.63-3.el6.noarch  
Installing xml-common-0.6.3-32.el6.noarch  
Installing tzdata-20101-1.el6.noarch  
Installing iso-codes-3.16-2.el6.noarch
```

9. More Command

more command allows quickly view file and shows details in percentage. You can page up and down. Press ‘q’ to quit out from more window.

```
# more install.log  
Installing setup-2.8.14-10.el6.noarch  
warning: setup-2.8.14-10.el6.noarch: Header V3 RSA/SHA256 Signature, key ID  
c105b9de: NOKEY  
Installing filesystem-2.4.30-2.1.el6.i686  
Installing ca-certificates-2010.63-3.el6.noarch  
Installing xml-common-0.6.3-32.el6.noarch  
Installing tzdata-20101-1.el6.noarch  
Installing iso-codes-3.16-2.el6.noarch  
--More-- (10%)
```

10. CP Command

Copy file from source to destination preserving same mode.

```
# cp -p fileA fileB
```

You will be prompted before overwrite to file.

```
# cp -i fileA fileB
```

11. MV Command

Rename **fileA** to **fileB**. **-i** options prompt before overwrite. Ask for confirmation if exist already.

```
# mv -i fileA fileB
```

12. Cat Command

cat command used to view multiple file at the same time.

```
# cat fileA fileB
```

You combine **more** and **less** command with cat command to view file contain if that doesn’t fit in single screen / page.

```
# cat install.log | less
```

```
# cat install.log | more
```

For more examples of Linux cat command read our article on [13 Basic Cat Command Examples in Linux](#).

13. Cd command (change directory)

with cd command (change directory) it will goes to **fileA** directory.

```
# cd /fileA
```

14. pwd command (print working directory)

pwd command return with present working directory.

```
# pwd  
/root
```

15. Sort command

Sorting lines of text files in ascending order. with **-r** options will sort in descending order.

```
#sort fileA.txt  
#sort -r fileA.txt
```

16. VI Command

Vi is a most popular text editor available most of the **UNIX-like OS**. Below examples open file in read only with **-R** option. Press '**:q**' to quit from vi window.

```
# vi -R /etc/shadows
```

17. SSH Command (Secure Shell)

SSH command is used to login into remote host. For example the below ssh command will connect to remote host (**192.168.50.2**) using user as **narad**.

```
# ssh narad@192.168.50.2
```

To check the version of ssh use option **-V** (uppercase) shows version of ssh.

```
# ssh -V  
OpenSSH_5.3p1, OpenSSL 1.0.0-fips 29 Mar 2010
```

18. Ftp or sftp Command

ftp or **sftp** command is used to connect to remote ftp host. **ftp** is (**file transfer protocol**) and **sftp** is (**secure file transfer protocol**). For example the below commands will connect to **ftp** host (**192.168.50.2**).

```
# ftp 192.168.50.2
# sftp 192.168.50.2
```

Putting multiple files in remote host with **mput** similarly we can do **mget** to download multiple files from remote host.

```
# ftp > mput *.txt
# ftp > mget *.txt
```

19. Service Command

Service command call script located at **/etc/init.d/** directory and execute the script. There are two ways to start the any service. For example we start the service called **httpd** with service command.

```
# service httpd start
OR
# /etc/init.d/httpd start
```

20. Free command

Free command shows **free**, **total** and **swap memory** information in bytes.

```
# free
total        used         free        shared      buffers       cached
Mem:       1030800      735944      294856          0      51648      547696
-/+ buffers/cache:   136600      894200
Swap:      2064376          0      2064376
```

Free with **-t** options shows **total memory** used and available to use in bytes.

```
# free -t
total        used         free        shared      buffers       cached
Mem:       1030800      736096      294704          0      51720      547704
-/+ buffers/cache:   136672      894128
Swap:      2064376          0      2064376
Total:     3095176      736096      2359080
```

21. Top Command

top command displays processor activity of your system and also displays tasks managed by kernel in real-time. It'll show **processor** and **memory** are being used. Use top command with '**u**' option this will display specific User process details as shown below. Press '**O**' (**uppercase letter**) to sort as per desired by you. Press '**q**' to quit from top screen.

```
# top -u tecmint
top - 11:13:11 up 3:19, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 116 total, 1 running, 115 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.3%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si,
0.0%st
Mem: 1030800k total, 736188k used, 294612k free, 51760k buffers
Swap: 2064376k total, 0k used, 2064376k free, 547704k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1889 tecmint 20 0 11468 1648 920 S 0.0 0.2 0:00.59 sshd
1890 tecmint 20 0 5124 1668 1416 S 0.0 0.2 0:00.44 bash
6698 tecmint 20 0 11600 1668 924 S 0.0 0.2 0:01.19 sshd
6699 tecmint 20 0 5124 1596 1352 S 0.0 0.2 0:00.11 bash
```

For more about top command we've already compiled a list of [12 TOP Command Examples in Linux](#).

22. Tar Command

tar command is used to compress files and folders in Linux. For example the below command will create a archive for **/home** directory with file name as **archive-name.tar**.

```
# tar -cvf archive-name.tar /home
```

To extract tar archive file use the option as follows.

```
# tar -xvf archive-name.tar
```

To understand more about **tar command** we've created a complete **how-to guide** on tar command at [18 Tar Command Examples in Linux](#).

23. Grep Command

grep search for a given string in a file. Only **tecmint** user displays from **/etc/passwd** file. we can use **-i** option for ignoring case sensitive.

```
# grep tecmint /etc/passwd
tecmint:x:500:500::/home/tecmint:/bin/bash
```

24. Find Command

Find command used to search **files**, **strings** and **directories**. The below example of find command search **tecmint** word in '/' partition and return the output.

```
# find / -name tecmint
/var/spool/mail/tecmint
/home/tecmint
/root/home/tecmint
```

For complete guide on **Linux find command** examples fount at [35 Practical Examples of Linux Find Command](#).

25. lsof Command

lsof mean List of all open files. Below lsof command list of all opened files by user **tecmint**.

```
# lsof -u tecmint
COMMAND  PID   USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
sshd    1889 tecmint cwd DIR    253,0    4096    2 /
sshd    1889 tecmint txt REG   253,0  532336 298069 /usr/sbin/sshd
sshd    1889 tecmint DEL REG   253,0          412940
/lib/libcom_err.so.2.1
sshd    1889 tecmint DEL REG   253,0          393156 /lib/ld-2.12.so
sshd    1889 tecmint DEL REG   253,0          298643
/usr/lib/libcrypto.so.1.0.0
sshd    1889 tecmint DEL REG   253,0          393173 /lib/libnsl-
2.12.so
sshd    1889 tecmint DEL REG   253,0          412937
/lib/libkrb5support.so.0.1
sshd    1889 tecmint DEL REG   253,0          412961 /lib/libplc4.so
```

For more **lsof command examples** visit [10 lsof Command Examples in Linux](#).

26. last command

With **last** command we can watch user's activity in the system. This command can execute normal user also. It will display complete user's info like **terminal, time, date, system reboot or boot** and **kernel version**. Useful command to troubleshoot.

```
# last
tecmint pts/1      192.168.50.1      Tue Sep 18 08:50  still logged in
tecmint pts/0      192.168.50.1      Tue Sep 18 07:59  still logged in
reboot  system boot 2.6.32-279.el6.i  Tue Sep 18 07:54 - 11:38  (03:43)
root    pts/1      192.168.50.1      Sun Sep 16 10:40 - down  (03:53)
root    pts/0      :0.0                Sun Sep 16 10:36 - 13:09  (02:32)
root    tty1       :0                  Sun Sep 16 10:07 - down  (04:26)
reboot  system boot 2.6.32-279.el6.i  Sun Sep 16 09:57 - 14:33  (04:35)
narad   pts/2      192.168.50.1      Thu Sep 13 08:07 - down  (01:15)
```

You can use **last** with **username** to know for specific user's activity as shown below.

```
# last tecmint
tecmint pts/1      192.168.50.1      Tue Sep 18 08:50  still logged in
tecmint pts/0      192.168.50.1      Tue Sep 18 07:59  still logged in
tecmint pts/1      192.168.50.1      Thu Sep 13 08:07 - down  (01:15)
tecmint pts/4      192.168.50.1      Wed Sep 12 10:12 - 12:29  (02:17)
```

27. ps command

ps command displays about processes running in the system. Below example show **init** process only.

```
# ps -ef | grep init
root      1      0  0 07:53 ?          00:00:04 /sbin/init
root    7508  6825  0 11:48 pts/1      00:00:00 grep init
```

28. kill command

Use **kill** command to terminate process. First find process **id** with **ps** command as shown below and kill process with **kill -9** command.

```
# ps -ef | grep init
root      1      0  0 07:53 ?          00:00:04 /sbin/init
root    7508  6825  0 11:48 pts/1      00:00:00 grep init
# kill -9 7508
```

29. rm command

rm command used to remove or delete a file without prompting for confirmation.

```
# rm filename
```

Using **-i** option to get confirmation before removing it. Using options '**-r**' and '**-f**' will remove the file forcefully without confirmation.

```
# rm -i test.txt
rm: remove regular file `test.txt'?
```

30. mkdir command example.

mkdir command is used to create directories under Linux.

```
# mkdir directoryname
```

This is a handy day to day useable basic commands in Linux / Unix-like operating system. Kindly share through our comment box if we missed out.

Difference between soft links and hard links in Linux

By [Ghansham](#) | December 18, 2016 | [cPanel/WHM](#), [How to](#), [Linux](#)

Hi Friends, We have always little bit of confuse about hard link and soft link. I am trying to clear confusion. Let's see difference between soft links and hard links in Linux. Explaining the difference between a symbolic link or a symlink and a hard link is easy and vital to knowing about how Linux/Unix environments work.

There are 2 types of links that you can create within Linux:

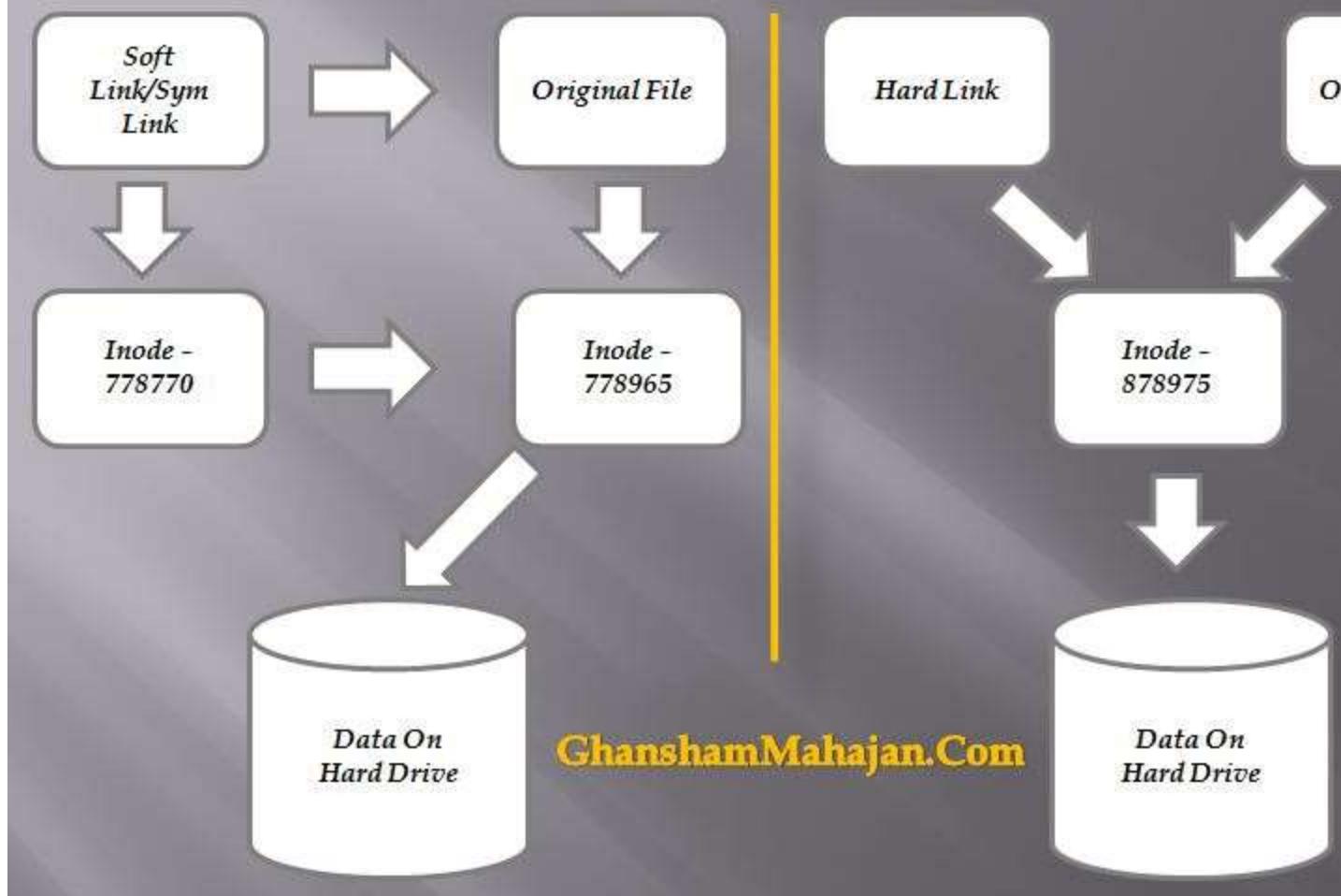
Hard Links

Symbolic Links

A **symbolic link** is much like a desktop shortcut within Windows. The symbolic link merely points to the location of a file.

A **hard link** is actually the same file that it links to but with a different name. It's share the same inode.

Difference between soft links and hard links in Linux



Difference between soft links and hard links in Linux

Hard link: Hard link refers to "The specific location of physical data".

- Hard Link is a mirror copy of the original file.
- Hard links share the same inode.
- Any changes made to the original or Hard linked file will reflect the other.
- Even if you delete any one of the files, nothing will happen to the other hard links.
- But soft link which points to deleted hard link become a dangling soft link.
- You can't link a directory even within the same file system.
- Hard links can't cross file systems.

Soft link (also called symbolic link): Soft link refers to "A symbolic path indicating the abstract location of another file".

- Soft Link is a symbolic link to the original file.(more like windows shortcuts)
- Soft Links will have a different Inode value.
- Any changes made to the soft link will reflect the original file and its hard links.
- A soft link points to the original file. If you delete the original file, the soft link fails. It would become dangling symbolic link.
- If you delete the soft link, nothing will happen.
- You can link a directory using soft link on same file system and also on other file system.
- Soft links can cross file systems

Soft Link and Hard Link



First we Need To Know About Inodes

Hard links and **soft link** is an important concept in the Linux file system, which relates to the index node in the file system (inode). Inode is one of the four basic concepts in Linux virtual file system (VFS). Through the analysis of the relation and the difference between hard links and soft links, we can better understand Linux VFS.

We know that the file has file name and data, which is divided into two parts: user data and metadata. The user data, namely the file data blocks (data block), data block is a place that store the real content. metadata is additional properties of file, such as: file size, created time of file, owner information of file. In Linux, the inode in the metadata, (inode is a part of the file metadata but which do not contain a file name, inode, namely the inode number) is uniquely identifies of the file rather than the file name. The file name is only for memory and is convenient for people to use. through the inode number, system can find the correct file data block. The below figure shows how the program obtain the contents of the file by file name.

In order to solve the problem of file sharing, Linux system introduce two links: hard link (hard link) and soft link (also called a symbolic link). A link will resolve file sharing for the Linux system. If a inode is correspond to multiple file name, said these files as hard links. In other words, a hard link is the same file with multiple aliases. Hard links can be created by the command **link** or **In**.

Soft link is a common file, there is only a little special for the content of data block. A soft link has its own inode number and the user data block , pls see the below figure. So soft link is created without many restrictions similar with hard links:

- A soft link has its own file attributes and authority
- You can create a soft link to the file or directory does not exist
- A soft link can cross file system
- Soft links can be created to the file or directory
- Create a soft link, link count i_nlink does not increase
- Removing the soft link does not affect the pointed file, but the original file if it is to be deleted, then the soft links is called the dead link

Q. What is the one line answer to the question “What is the main difference between hard links & soft links” ?

A. A softlink will have a different Inode number than the source file, which will be having a pointer to the source file but hardlink will be using the same Inode number as the source file.

Q. How can I find all the Soft Links in my system ?

A. Use this command for the same “find /etc -type l -exec ls -li {} \;”

Q. How can I find all the files having Hard Links in my system ?

A. Use this command for the same “find / -links +2 -type f -exec ls -li {} \;”

Q. How to find whether a file is a softlink ?

A. Simply using this command “ls -l” will tell you whether a file is pointing to some other file or not.

Q. How to check whether a file have any softlink pointing to it ?

A. Till now, I am not aware of any way to do that. If I will find any, I will surely update my post.

Q. How can I find out the source file of a hard link ?

A. No, you can't find out the source file of a hard link. Once hard link is created, there is no way to tell which was the first file created.

Q. Can I make a Soft link to a Hard link and Vice Versa ?

A. Yes, both soft links and hard links acts as normal files of the file system, so you can do both.

How file is deleted having hard links:

So, as it's pretty clear from the above article that hard links are just the reference to the main file location, and even if you delete one link, the data will still be intact. So, to remove a hard link, you need to remove all the links, which are referring to the file. Once the "link count" goes to "0", then the inode is removed by the filesystem, and file is deleted.

When to use Soft Link:

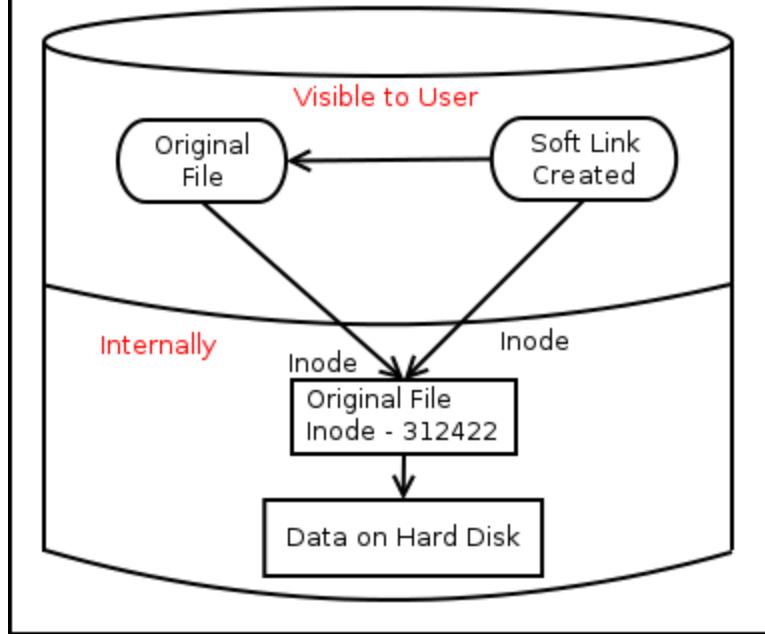
1. Link across filesystems: If you want to link files across the filesystems, you can only use symlinks/soft links.
2. Links to directory: If you want to link directories, then you must be using Soft links, as you can't create a hard link to a directory.

When to use Hard Link:

1. Storage Space: Hard links takes very negligible amount of space, as there are no new inodes created while creating hard links. In soft links we create a file which consumes space (usually 4KB, depending upon the filesystem)
2. Performance: Performance will be slightly better while accessing a hard link, as you are directly accessing the disk pointer instead of going through another file.
3. Moving file location: If you move the source file to some other location on the same filesystem, the hard link will still work, but soft link will fail.
4. Redundancy: If you want to make sure safety of your data, you should be using hard link, as in hard link, the data is safe, until all the links to the files are deleted, instead of that in soft link, you will lose the data if the master instance of the file is deleted.

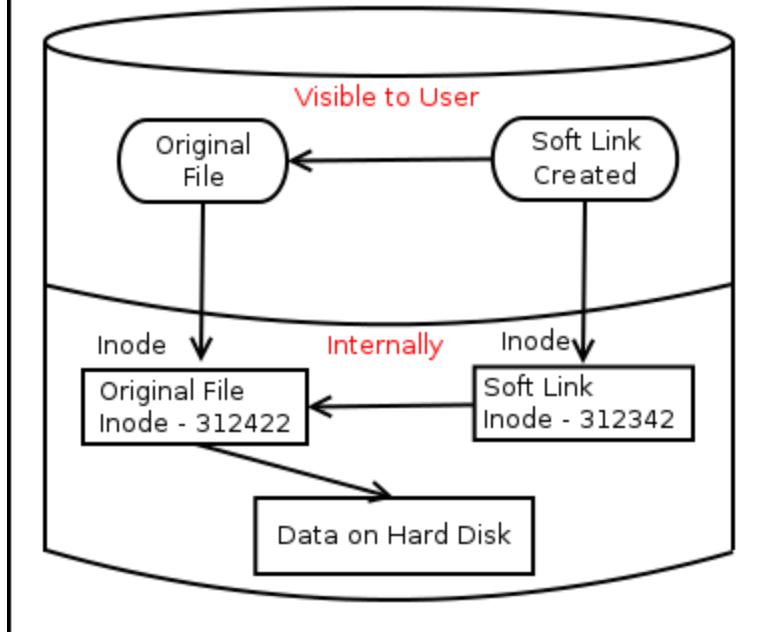
Hard Link

Hard Link is direct pointer to the original inode of the original file. If you compare the original file with hard link, there won't be any differences.



Soft Link / Symlink

A softlink is a file that have the information to point to another file/inode. That inode points to the data on the hard drive.



PRACTICAL:

Symbolic links are created with the “ln” command in linux. The syntax of the command is:

```
$ ln -s
```

-s = This flag tells to create a symlink (if you don't use this it will create a hard link, which we will talk about soon).

For Example, if you want to create a soft link of one fo your favorite application, like gedit, on your desktop, use the command like this:

```
$ ln -s /usr/bin/gedit ~/Desktop/gedit
```

I hope now the concept of Soft Links should be clear.

You can create a hard link with the same command “ln” like this

```
# ln
```

So, to create a hard link of gedit program on your desktop, you will use the command like this:

```
# ln /usr/bin/gedit ~/Desktop/gedit
```

Now, the bigger question is, who will decide what is better and when to use soft link or hard link

NOTE: The only information not included in an inode, is the file name and directory. These are stored in the special directory files.

Hard link and Soft link Difference Hard Link, Soft Link inode

What is link?

Link is connection between two file

Here there are two links are there Soft link and hard link

Hard link syntax

```
#ln /absolute-path/original-file /hard-link-file
```

```
[root@localhost /]# ln rhelpfile /var/
```

Soft Link syntax

```
#ln -s /absolute-path/original-file soft-link-file
```

```
#ln -s
```

Soft link is also called symbolic link, sym link

HardLink	SoftLink
<ul style="list-style-type: none">Original and link file will have same inode no.It cannot be created across the partitionsIf original file is deleted then also the link file will be accessibleEditing of original file will replicate in the linked fileSize of Hardlink file is same as original file.	<ul style="list-style-type: none">Inode no. of the link file will be differentIt can be created across the partitionsIf original file is deleted the link file will not be accessibleEditing of original file will replicate in the linked fileSize of Softlink file is smaller than original file.

Difference between Hard link and Soft link

Hard link

- 1)Links have the original source content.
- 2)If we delete original file we can get back the data
- 3)Link inode values are same (what is inode)

```
[root@localhost /]# ls -i rhelpfile
```

```
32771 rhelpfile
```

```
[root@localhost var]# ls -i rhelpfile
```

```
32771 rhelpfile
```

- 4)Hard link we can link only file

Link the directory is not possible

```
[root@localhost ~]# ln rhel/ /var/
```

```
ln: `rhel/': hard link not allowed for directory
```

- 5)We can not use hard link in Network File service

6) if the original file is edited it will reflect in the linked file

Soft Link

- 1) Link has the path of the source file.
- 2) If we delete the original file of link we can not get it back
#ls -i it will list out list of inode value of file
- 3) Soft link inode value is different (what is inode)
- 4) Soft link we can create the link file and folder
- 5) Soft link we can use the Network file service
- 6) it is same like hard link if file is edited it will be reflected

SOFT LINK	HARD LINK
original file is delete(broken), it will show in RED color .	It will present without unchanged
Create the soft link between partition. We were getting lot of advantage production server, if the /opt partition is 100% , then there is no need to worry alert that particular partition . From the another free partition just create the SOFT-LINK.	NOT EFFICIENT
PERMISSION: If we change the permission in original file or soft link it will reflect only to the original file only. soft link remain unchanged with full permission 777.	When we change the permission original file it 'll automatically reflect to the Hard-link (because i nodes are same between original file and hard link).
Possible for Directory.	Not possible for Directory.

15 Practical Examples of ‘echo’ command in Linux

by [Editor](#) | Published: August 21, 2014 | Last Updated: January 27, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

echo is one of the most commonly and widely used built-in command for Linux bash and C shells, that typically used in scripting language and batch files to display a line of text/string on standard output or a file.



echo command examples

The syntax for echo is:

```
echo [option(s)] [string(s)]
```

1. Input a line of text and display on standard output

```
$ echo Tecmint is a community of Linux Nerds
```

Outputs the following text:

```
Tecmint is a community of Linux Nerds
```

2. Declare a variable and echo its value. For example, Declare a variable of **x** and assign its value=**10**.

```
$ x=10
```

echo its value:

```
$ echo The value of variable x = $x
The value of variable x = 10
```

Note: The ‘-e‘ option in Linux acts as interpretation of escaped characters that are backslashed.

3. Using option ‘\b‘ – backspace with backslash interpreter ‘-e‘ which removes all the spaces in between.

```
$ echo -e "Tecmint \bis \ba \bcommunity \bof \bLinux \bNerds"
TecmintisacomunityofLinuxNerds
```

4. Using option ‘\n‘ – New line with backspace interpreter ‘-e‘ treats new line from where it is used.

```
$ echo -e "Tecmint \nis \na \ncommunity \nof \nLinux \nNerds"
Tecmint
is
a
community
of
Linux
Nerds
```

5. Using option ‘\t‘ – horizontal tab with backspace interpreter ‘-e‘ to have horizontal tab spaces.

```
$ echo -e "Tecmint \tis \ta \tcommunity \tof \tLinux \tNerds"
Tecmint      is      a      community      of      Linux      Nerds
```

6. How about using option new Line ‘\n‘ and horizontal tab ‘\t‘ simultaneously.

```
$ echo -e "\n\tTecmint \n\tis \n\tta \n\tcommunity \n\ttof \n\tLinux \n\tNerds"
Tecmint
is
a
community
of
Linux
Nerds
```

7. Using option ‘\v‘ – vertical tab with backspace interpreter ‘-e‘ to have vertical tab spaces.

```
$ echo -e "\vTecmint \vis \va \vcommunity \vof \vLinux \vNerds"
Tecmint
is
```

```
a  
community  
of  
Linux  
Nerds
```

8. How about using option new Line ‘\n‘ and vertical tab ‘\v‘ simultaneously.

```
$ echo -e "\n\vTecmint \n\vis \n\va \n\vcommunity \n\vof \n\vLinux \n\vNerds"  
Tecmint  
is  
a  
community  
of  
Linux  
Nerds
```

Note: We can double the vertical tab, horizontal tab and new line spacing using the option two times or as many times as required.

9. Using option ‘\r‘ – carriage return with backspace interpreter ‘-e‘ to have specified carriage return in output.

```
$ echo -e "Tecmint \ris a community of Linux Nerds"  
is a community of Linux Nerds
```

10. Using option ‘\c‘ – suppress trailing new line with backspace interpreter ‘-e‘ to continue without emitting new line.

```
$ echo -e "Tecmint is a community \cof Linux Nerds"  
Tecmint is a community avi@tecmint:~$
```

11. Omit echoing trailing new line using option ‘-n‘.

```
$ echo -n "Tecmint is a community of Linux Nerds"  
Tecmint is a community of Linux Nerdsavi@tecmint:~/Documents$
```

12. Using option ‘\a‘ – alert return with backspace interpreter ‘-e‘ to have sound alert.

```
$ echo -e "Tecmint is a community of \aLinux Nerds"  
Tecmint is a community of Linux Nerds
```

Note: Make sure to check Volume key, before firing.

13. Print all the files/folder using echo command (ls command alternative).

```
$ echo *  
103.odt 103.pdf 104.odt 104.pdf 105.odt 105.pdf 106.odt 106.pdf 107.odt  
107.pdf 108a.odt 108.odt 108.pdf 109.odt 109.pdf 110b.odt 110.odt 110.pdf  
111.odt 111.pdf 112.odt 112.pdf 113.odt linux-headers-3.16.0-
```

```
customkernel_1_amd64.deb linux-image-3.16.0-customkernel_1_amd64.deb  
network.jpeg
```

14. Print files of a specific kind. For example, let's assume you want to print all ‘.jpeg‘ files, use the following command.

```
$ echo *.jpeg  
network.jpeg
```

15. The echo can be used with redirect operator to output to a file and not standard output.

```
$ echo "Test Page" > testpage  
## Check Content  
avi@tecmint:~$ cat testpage  
Test Page
```

echo Options

Options	Description
-n	do not print the trailing newline.
-e	enable interpretation of backslash escapes.
\b	backspace
\\\	backslash
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

That's all for now and don't forget to provide us with your valuable feedback in the comments below

7 ‘dmesg’ Commands for Troubleshooting and Collecting Information of Linux Systems

by [Narad Shrestha](#) | Published: July 15, 2014 | Last Updated: September 16, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

The ‘**dmesg**’ command displays the messages from the kernel ring buffer. A system passes multiple runlevel from where we can get lot of information like system architecture, cpu, attached device, RAM etc. When computer boots up, a kernel (core of an operating system) is loaded into memory. During that period number of messages are being displayed where we can see hardware devices detected by kernel.

Read Also: [10 Linux Commands to Collect System and Hardware Information](#)



dmesg Command Examples

The messages are very important in terms of diagnosing purpose in case of device failure. When we connect or disconnect hardware device on the system, with the help of dmesg command we come to know detected or disconnected information on the fly. The **dmesg** command is available on most **Linux and Unix** based Operating System.

Let’s throw some light on most famous tool called ‘dmesg’ command with their practical examples as discussed below. The exact syntax of dmesg as follows.

```
# dmseg [options...]
```

1. List all loaded Drivers in Kernel

We can use text-manipulation tools i.e. ‘**more**‘, ‘**tail**‘, ‘**less**‘ or ‘**grep**‘ with dmesg command. As output of dmesg log won’t fit on a single page, using dmesg with pipe more or less command will display logs in a single page.

```
[root@tecmint.com ~]# dmesg | more
[root@tecmint.com ~]# dmesg | less
```

Sample Output

```
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuart
[    0.000000] Linux version 3.11.0-13-generic (buildd@aatxe) (gcc version
4.8.1 (Ubuntu/Linaro 4.8.1-10ubuntu8) ) #20-Ubuntu SMP Wed Oct 23 17:26:33
UTC 2013
(Ubuntu 3.11.0-13.20-generic 3.11.6)
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   NSC Geode by NSC
[    0.000000]   Cyrix CyrixInstead
[    0.000000]   Centaur CentaurHauls
[    0.000000]   Transmeta GenuineTMx86
[    0.000000]   Transmeta TransmetaCPU
[    0.000000]   UMC UMC UMC UMC
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x000000000f0000-0x000000000000ffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000100000-0x000000007dc08bff] usable
[    0.000000] BIOS-e820: [mem 0x000000007dc08c00-0x000000007dc5cbff] ACPI
NVS
[    0.000000] BIOS-e820: [mem 0x000000007dc5cc00-0x000000007dc5ebff] ACPI
data
[    0.000000] BIOS-e820: [mem 0x000000007dc5ec00-0x000000007fffffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000e0000000-0x00000000efffffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000fec00000-0x00000000fed003ff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000fed20000-0x00000000fed9ffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000fee00000-0x00000000feeffffff]
reserved
[    0.000000] BIOS-e820: [mem 0x00000000ffb00000-0x00000000ffffffffff]
reserved
[    0.000000] NX (Execute Disable) protection: active
.....
```

Read Also: [Manage Linux Files Effectively using commands head, tail and cat](#)

2. List all Detected Devices

To discover which hard disks has been detected by kernel, you can search for the keyword “**sda**” along with “**grep**” like shown below.

```
[root@tecmint.com ~]# dmesg | grep sda
[    1.280971] sd 2:0:0:0: [sda] 488281250 512-byte logical blocks: (250
GB/232 GiB)
[    1.281014] sd 2:0:0:0: [sda] Write Protect is off
[    1.281016] sd 2:0:0:0: [sda] Mode Sense: 00 3a 00 00
[    1.281039] sd 2:0:0:0: [sda] Write cache: enabled, read cache: enabled,
doesn't support DPO or FUA
[    1.359585]   sda: sda1 sda2 < sda5 sda6 sda7 sda8 >
[    1.360052] sd 2:0:0:0: [sda] Attached SCSI disk
[    2.347887] EXT4-fs (sda1): mounted filesystem with ordered data mode.
Opts: (null)
[    22.928440] Adding 3905532k swap on /dev/sda6. Priority:-1 extents:1
across:3905532k FS
[    23.950543] EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
[    24.134016] EXT4-fs (sda5): mounted filesystem with ordered data mode.
Opts: (null)
[    24.330762] EXT4-fs (sda7): mounted filesystem with ordered data mode.
Opts: (null)
[    24.561015] EXT4-fs (sda8): mounted filesystem with ordered data mode.
Opts: (null)
```

NOTE: The ‘sda’ first SATA hard drive, ‘sdb’ is the second SATA hard drive and so on. Search with ‘hda’ or ‘hdb’ in the case of IDE hard drive.

3. Print Only First 20 Lines of Output

The ‘head’ along with dmesg will show starting lines i.e. ‘dmesg | head -20’ will print only 20 lines from the starting point.

```
[root@tecmint.com ~]# dmesg | head -20
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuart
[    0.000000] Linux version 3.11.0-13-generic (buildd@aatxe) (gcc version
4.8.1 (Ubuntu/Linaro 4.8.1-10ubuntu8) ) #20-Ubuntu SMP Wed Oct 23 17:26:33
UTC 2013 (Ubuntu 3.11.0-13.20-generic 3.11.6)
[    0.000000] KERNEL supported cpus:
[    0.000000]     Intel GenuineIntel
[    0.000000]     AMD AuthenticAMD
[    0.000000]     NSC Geode by NSC
[    0.000000]     Cyrix CyrixInstead
[    0.000000]     Centaur CentaurHauls
[    0.000000]     Transmeta GenuineTMx86
[    0.000000]     Transmeta TransmetaCPU
[    0.000000]     UMC UMC UMC UMC
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x0000000000f0000-0x000000000000ffff]
reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x0000000007dc08bff] usable
```

```
[    0.000000] BIOS-e820: [mem 0x000000007dc08c00-0x000000007dc5cbff] ACPI  
NVS  
[    0.000000] BIOS-e820: [mem 0x000000007dc5cc00-0x000000007dc5ebff] ACPI  
data  
[    0.000000] BIOS-e820: [mem 0x000000007dc5ec00-0x000000007fffffff]  
reserved
```

4. Print Only Last 20 Lines of Output

The ‘tail’ along with dmesg command will print only 20 last lines, this is useful in case we insert removable device.

```
[root@tecmint.com ~]# dmesg | tail -20  
parport0: PC-style at 0x378, irq 7 [PCSPP,TRISTATE]  
ppdev: user-space parallel port driver  
EXT4-fs (sda1): mounted filesystem with ordered data mode  
Adding 2097144k swap on /dev/sda2. Priority:-1 extents:1 across:2097144k  
readahead-disable-service: delaying service audited  
ip_tables: (C) 2000-2006 Netfilter Core Team  
nf_conntrack version 0.5.0 (16384 buckets, 65536 max)  
NET: Registered protocol family 10  
lo: Disabled Privacy Extensions  
e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None  
Slow work thread pool: Starting up  
Slow work thread pool: Ready  
FS-Cache: Loaded  
CacheFiles: Loaded  
CacheFiles: Security denies permission to nominate security context: error -  
95  
eth0: no IPv6 routers present  
type=1305 audit(1398268784.593:18630): audit_enabled=0 old=1 auid=4294967295  
ses=4294967295 res=1  
readahead-collector: starting delayed service auditd  
readahead-collector: sorting  
readahead-collector: finished
```

5. Search Detected Device or Particular String

It's difficult to search particular string due to length of dmesg output. So, filter the lines with are having string like ‘usb’ ‘dma’ ‘tty’ and ‘memory’ etc. The ‘-i’ option instruct to [grep command](#) to ignore the case (upper or lower case letters).

```
[root@tecmint.com log]# dmesg | grep -i usb  
[root@tecmint.com log]# dmesg | grep -i dma  
[root@tecmint.com log]# dmesg | grep -i tty  
[root@tecmint.com log]# dmesg | grep -i memory
```

Sample Output

```
[    0.000000] Scanning 1 areas for low memory corruption  
[    0.000000] initial memory mapped: [mem 0x00000000-0x01ffff]  
[    0.000000] Base memory trampoline at [c009b000] 9b000 size 16384  
[    0.000000] init_memory_mapping: [mem 0x00000000-0x000ffff]
```

```
[ 0.000000] init_memory_mapping: [mem 0x37800000-0x379fffff]
[ 0.000000] init_memory_mapping: [mem 0x34000000-0x377fffff]
[ 0.000000] init_memory_mapping: [mem 0x00100000-0x33fffff]
[ 0.000000] init_memory_mapping: [mem 0x37a00000-0x37bfdfff]
[ 0.000000] Early memory node ranges
[ 0.000000] PM: Registered nosave memory: [mem 0x0009f000-0x000effff]
[ 0.000000] PM: Registered nosave memory: [mem 0x000f0000-0x000fffff]
[ 0.000000] please try 'cgroup_disable=memory' option if you don't want
memory cgroups
[ 0.000000] Memory: 2003288K/2059928K available (6352K kernel code, 607K
rwdata, 2640K rodata, 880K init, 908K bss, 56640K reserved, 1146920K highmem)
[ 0.000000] virtual kernel memory layout:
[ 0.004291] Initializing cgroup subsys memory
[ 0.004609] Freeing SMP alternatives memory: 28K (c1a3e000 - c1a45000)
[ 0.899622] Freeing initrd memory: 23616K (f51d0000 - f68e0000)
[ 0.899813] Scanning for low memory corruption every 60 seconds
[ 0.946323] agpgart-intel 0000:00:00.0: detected 32768K stolen memory
[ 1.360318] Freeing unused kernel memory: 880K (c1962000 - c1a3e000)
[ 1.429066] [drm] Memory usable by graphics device = 2048M
```

6. Clear dmesg Buffer Logs

Yes, we can clear dmesg logs if required with below command. It will clear dmesg ring buffer message logs till you executed the command below. Still you can view logs stored in ‘/var/log/dmesg’ files. If you connect any device will generate dmesg output.

```
[root@tecmint.com log]# dmesg -c
```

7. Monitoring dmesg in Real Time

Some distro allows command ‘tail -f /var/log/dmesg’ as well for real time dmesg monitoring.

```
[root@tecmint.com log]# watch "dmesg | tail -20"
```

Conclusion: The dmesg command is useful as dmesg records all the system changes done or occur in real time. As always you can **man dmesg** to get more information.

Exploring /proc File System in Linux

by [Rob Krul](#) | Published: October 25, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Today, we are going to take a look inside the **/proc** directory and develop a familiarity with it. The **/proc** directory is present on all **Linux** systems, regardless of flavor or architecture.

One misconception that we have to immediately clear up is that the **/proc** directory is **NOT** a real **File System**, in the sense of the term. It is a **Virtual File System**. Contained within the **procs** are information about processes and other system information. It is mapped to **/proc** and mounted at **boot** time.



Exploring /proc File System

First, lets get into the **/proc** directory and have a look around:

```
# cd /proc
```

The first thing that you will notice is that there are some **familiar sounding files**, and then a whole bunch of **numbered directories**. The **numbered directories** represent **processes**, better known as **PIDs**, and within them, a command that occupies them. The files contain system information such as **memory (meminfo)**, **CPU information (cpuinfo)**, and available **filesystems**.

Read Also: [Linux Free Command to Check Physical Memory and Swap Memory](#)

Let's take a look at one of the files first:

```
# cat /proc/meminfo
```

Sample Output

which returns something similar to this:

```
MemTotal:           604340 kB
MemFree:            54240 kB
Buffers:             18700 kB
Cached:              369020 kB
SwapCached:          0 kB
Active:              312556 kB
Inactive:             164856 kB
Active(anon):        89744 kB
Inactive(anon):       360 kB
Active(file):         222812 kB
Inactive(file):       164496 kB
Unevictable:          0 kB
Mlocked:              0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                  0 kB
Writeback:              0 kB
AnonPages:             89724 kB
Mapped:                18012 kB
Shmem:                  412 kB
Slab:                   50104 kB
SReclaimable:          40224 kB
...
...
```

As you can see, **/proc/meminfo** contains a bunch of information about your system's memory, including the total amount available (in **kb**) and the amount free on the top two lines.

Running the [cat command](#) on any of the files in **/proc** will output their contents. Information about any files is available in the man page by running:

```
# man 5 /proc/<filename>
```

I will give you quick rundown on **/proc**'s files:

1. **/proc/cmdline** – Kernel command line information.
2. **/proc/console** – Information about current consoles including tty.
3. **/proc/devices** – Device drivers currently configured for the running kernel.
4. **/proc/dma** – Info about current DMA channels.
5. **/proc/fb** – Framebuffer devices.
6. **/proc/filesystems** – Current filesystems supported by the kernel.
7. **/proc/iomem** – Current system memory map for devices.

8. **/proc/iports** – Registered port regions for input output communication with device.
9. **/proc/loadavg** – System load average.
10. **/proc/locks** – Files currently locked by kernel.
11. **/proc/meminfo** – Info about system memory (see above example).
12. **/proc/misc** – Miscellaneous drivers registered for miscellaneous major device.
13. **/proc/modules** – Currently loaded kernel modules.
14. **/proc/mounts** – List of all mounts in use by system.
15. **/proc/partitions** – Detailed info about partitions available to the system.
16. **/proc/pci** – Information about every PCI device.
17. **/proc/stat** – Record or various statistics kept from last reboot.
18. **/proc/swap** – Information about swap space.
19. **/proc/uptime** – Uptime information (in seconds).
20. **/proc/version** – Kernel version, gcc version, and Linux distribution installed.

Within **/proc**'s numbered directories you will find a few **files** and **links**. Remember that these directories' numbers correlate to the **PID** of the command being run within them. Let's use an example. On my system, there is a folder name **/proc/12**:

```
# cd /proc/12
# ls
```

Sample Output

attr	coredump_filter	io	mounts	oom_score_adj	smaps
wchan					
autogroup	cpuset	latency	mountstats	pagemap	stack
auxv	cwd	limits	net	personality	stat
cgroup	environ	loginuid	ns	root	statm
clear_refs	exe	maps	numa_maps	sched	status
cmdline	fd	mem	oom_adj	schedstat	syscall
comm	fdinfo	mountinfo	oom_score	sessionid	task

If I run:

```
# cat /proc/12/status
```

I get the following:

```
Name: xenwatch
State: S (sleeping)
Tgid: 12
Pid: 12
PPid: 2
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 64
Groups:
Threads: 1
SigQ: 1/4592
SigPnd: 0000000000000000
```

```
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: ffffffff ffffffff
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: ffffffff ffffffff
CapEff: ffffffff ffffffff
CapBnd: ffffffff ffffffff
Cpus_allowed: 1
Cpus_allowed_list: 0
Mems_allowed: 00000000,00000001
Mems_allowed_list: 0
voluntary_ctxt_switches: 84
nonvoluntary_ctxt_switches: 0
```

So, what does this mean? Well, the important part is at the top. We can see from the status file that this process belongs to **xenwatch**. Its current state is **sleeping**, and its process **ID** is **12**, obviously. We also can see who is running this, as **UID** and **GID** are **0**, indicating that this process belongs to the **root** user.

In any numbered directory, you will have a similar file structure. The most important ones, and their descriptions, are as follows:

1. **cmdline** – command line of the process
2. **environ** – environmental variables
3. **fd** – file descriptors
4. **limits** – contains information about the limits of the process
5. **mounts** – related information

You will also notice a number of links in the numbered directory:

1. **cwd** – a link to the current working directory of the process
2. **exe** – link to the executable of the process
3. **root** – link to the work directory of the process

This should get you started with familiarizing yourself with the **/proc** directory. It should also provide insight to how a number of commands obtain their info, such as **uptime**, **lsof**, **mount**, and **ps**, just to name a few.

10 Useful du (Disk Usage) Commands to Find Disk Usage of Files and Directories

by [Ravi Saive](#) | Published: January 21, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

The Linux “**du**” (**Disk Usage**) is a standard **Unix/Linux** command, used to check the information of disk usage of files and directories on a machine. The **du** command has many parameter options that can be used to get the results in many formats. The **du** command also displays the files and directory sizes in a recursively manner.



Check Disk Usage of Files and Folders In Linux

This article explains **10 useful “du” commands** with their examples, that might helps you to find out the sizes of files and directories in Linux. The information provided in this article are taken from the man pages of **du** command.

Read Also:

1. [12 “df” Command to Check Linux System Disk Space](#)

1. To find out the disk usage summary of a **/home/tecmint** directory tree and each of its sub directories. Enter the command as:

```
[root@tecmint]# du /home/tecmint
40      /home/tecmint/downloads
4       /home/tecmint/.mozilla/plugins
4       /home/tecmint/.mozilla/extensions
12      /home/tecmint/.mozilla
12      /home/tecmint/.ssh
689112  /home/tecmint/Ubuntu-12.10
```

```
689360 /home/tecmint
```

The output of the above command displays the number of disk blocks in the **/home/tecmint** directory along with its sub-directories.

2. Using “**-h**” option with “**du**” command provides results in “**Human Readable Format**”. Means you can see sizes in **Bytes**, **Kilobytes**, **Megabytes**, **Gigabytes** etc.

```
[root@tecmint]# du -h /home/tecmint
40K      /home/tecmint/downloads
4.0K     /home/tecmint/.mozilla/plugins
4.0K     /home/tecmint/.mozilla/extensions
12K      /home/tecmint/.mozilla
12K      /home/tecmint/.ssh
673M    /home/tecmint/Ubuntu-12.10
674M    /home/tecmint
```

3. To get the summary of a grand total disk usage size of an directory use the option “**-s**” as follows.

```
[root@tecmint]# du -sh /home/tecmint
674M    /home/tecmint
```

4. Using “**-a**” flag with “**du**” command displays the disk usage of all the files and directories.

```
[root@tecmint]# du -a /home/tecmint
4      /home/tecmint/.bash_logout
12     /home/tecmint/downloads/uploadprogress-1.0.3.1.tgz
24     /home/tecmint/downloads/Phpfiles-org.tar.bz2
40     /home/tecmint/downloads
12     /home/tecmint/uploadprogress-1.0.3.1.tgz
4      /home/tecmint/.mozilla/plugins
4      /home/tecmint/.mozilla/extensions
12     /home/tecmint/.mozilla
4      /home/tecmint/.bashrc
689108 /home/tecmint/Ubuntu-12.10/ubuntu-12.10-server-i386.iso
689112 /home/tecmint/Ubuntu-12.10
689360 /home/tecmint
```

5. Using “**-a**” flag along with “**-h**” displays disk usage of all files and folders in human readable format. The below output is more easy to understand as it shows the files in **Kilobytes**, **Megabytes** etc.

```
[root@tecmint]# du -ah /home/tecmint
4.0K    /home/tecmint/.bash_logout
12K    /home/tecmint/downloads/uploadprogress-1.0.3.1.tgz
24K    /home/tecmint/downloads/Phpfiles-org.tar.bz2
40K    /home/tecmint/downloads
12K    /home/tecmint/uploadprogress-1.0.3.1.tgz
4.0K    /home/tecmint/.mozilla/plugins
4.0K    /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
```

```
4.0K /home/tecmint/.bashrc
673M /home/tecmint/Ubuntu-12.10/ubuntu-12.10-server-i386.iso
673M /home/tecmint/Ubuntu-12.10
674M /home/tecmint
```

6. Find out the disk usage of a directory tree with its subtress in **Kilobyte** blocks. Use the “**-k**” (displays size in **1024** bytes units).

```
[root@tecmint]# du -k /home/tecmint
40      /home/tecmint/downloads
4       /home/tecmint/.mozilla/plugins
4       /home/tecmint/.mozilla/extensions
12      /home/tecmint/.mozilla
12      /home/tecmint/.ssh
689112  /home/tecmint/Ubuntu-12.10
689360  /home/tecmint
```

7. To get the summary of disk usage of directory tree along with its subtrees in **Megabytes (MB)** only. Use the option “**-mh**” as follows. The “**-m**” flag counts the blocks in **MB** units and “**-h**” stands for human readable format.

```
[root@tecmint]# du -mh /home/tecmint
40K     /home/tecmint/downloads
4.0K    /home/tecmint/.mozilla/plugins
4.0K    /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
12K    /home/tecmint/.ssh
673M   /home/tecmint/Ubuntu-12.10
674M   /home/tecmint
```

8. The “**-c**” flag provides a grand total usage disk space at the last line. If your directory taken **674MB** space, then the last two line of the output would be.

```
[root@tecmint]# du -ch /home/tecmint
40K     /home/tecmint/downloads
4.0K    /home/tecmint/.mozilla/plugins
4.0K    /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
12K    /home/tecmint/.ssh
673M   /home/tecmint/Ubuntu-12.10
674M   /home/tecmint
674M   total
```

9. The below command calculates and displays the disk usage of all files and directories, but excludes the files that matches given pattern. The below command excludes the “**.txt**” files while calculating the total size of directory. So, this way you can exclude any file formats by using flag “**--exclude**”. See the output there is no **txt** files entry.

```
[root@tecmint]# du -ah --exclude="*.txt" /home/tecmint
4.0K   /home/tecmint/.bash_logout
12K   /home/tecmint/downloads/uploadprogress-1.0.3.1.tgz
24K   /home/tecmint/downloads/Phpfiles-org.tar.bz2
```

```
40K    /home/tecmint/downloads
12K    /home/tecmint/uploadprogress-1.0.3.1.tgz
4.0K   /home/tecmint/.bash_history
4.0K   /home/tecmint/.bash_profile
4.0K   /home/tecmint/.mozilla/plugins
4.0K   /home/tecmint/.mozilla/extensions
12K    /home/tecmint/.mozilla
4.0K   /home/tecmint/.bashrc
24K    /home/tecmint/Phpfiles-org.tar.bz2
4.0K   /home/tecmint/geoipupdate.sh
4.0K   /home/tecmint/.zshrc
120K   /home/tecmint/goaccess-0.4.2.tar.gz.1
673M   /home/tecmint/Ubuntu-12.10/ubuntu-12.10-server-i386.iso
673M   /home/tecmint/Ubuntu-12.10
674M   /home/tecmint
```

10. Display the disk usage based on modification of time, use the flag “**–time**” as shown below.

```
[root@tecmint]# du -ha --time /home/tecmint
4.0K   2012-10-12 22:32      /home/tecmint/.bash_logout
12K    2013-01-19 18:48      /home/tecmint/downloads/uploadprogress-
1.0.3.1.tgz
24K    2013-01-19 18:48      /home/tecmint/downloads/Phpfiles-org.tar.bz2
40K    2013-01-19 18:48      /home/tecmint/downloads
12K    2013-01-19 18:32      /home/tecmint/uploadprogress-1.0.3.1.tgz
4.0K   2012-10-13 00:11      /home/tecmint/.bash_history
4.0K   2012-10-12 22:32      /home/tecmint/.bash_profile
0      2013-01-19 18:32      /home/tecmint/xyz.txt
0      2013-01-19 18:32      /home/tecmint/abc.txt
4.0K   2012-10-12 22:32      /home/tecmint/.mozilla/plugins
4.0K   2012-10-12 22:32      /home/tecmint/.mozilla/extensions
12K    2012-10-12 22:32      /home/tecmint/.mozilla
4.0K   2012-10-12 22:32      /home/tecmint/.bashrc
24K    2013-01-19 18:32      /home/tecmint/Phpfiles-org.tar.bz2
4.0K   2013-01-19 18:32      /home/tecmint/geoipupdate.sh
4.0K   2012-10-12 22:32      /home/tecmint/.zshrc
120K   2013-01-19 18:32      /home/tecmint/goaccess-0.4.2.tar.gz.1
673M   2013-01-19 18:51      /home/tecmint/Ubuntu-12.10/ubuntu-12.10-
server-i386.iso
673M   2013-01-19 18:51      /home/tecmint/Ubuntu-12.10
674M   2013-01-19 18:52      /home/tecmint
```

12 Useful “df” Commands to Check Disk Space in Linux

by [Ravi Saive](#) | Published: January 15, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

On the internet you will find plenty of tools for checking disk space utilization in Linux. However, Linux has a strong built in utility called ‘**df**’. The ‘**df**’ command stand for “**disk filesystem**”, it is used to get full summary of available and used disk space usage of file system on Linux system.

Using ‘**-h**’ parameter with (**df -h**) will shows the file system disk space statistics in “**human readable**” format, means it gives the details in bytes, mega bytes and gigabyte.



Useful df Command Examples

This article explain a way to get the full information of Linux disk space usage with the help of ‘**df**’ command with their practical examples. So, you could better understand the usage of **df** command in Linux.

1. Check File System Disk Space Usage

The “**df**” command displays the information of device name, total blocks, total disk space, used disk space, available disk space and mount points on a file system.

```
[root@tecmint ~]# df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/cciss/c0d0p2   78361192  23185840  51130588  32% /
/dev/cciss/c0d0p5   24797380  22273432  1243972  95% /home
/dev/cciss/c0d0p3   29753588  25503792  2713984  91% /data
/dev/cciss/c0d0p1   295561      21531   258770   8% /boot
```

```
tmpfs          257476        0     257476    0% /dev/shm
```

2. Display Information of all File System Disk Space Usage

The same as above, but it also displays information of dummy file systems along with all the file system disk usage and their memory utilization.

```
[root@tecmint ~]# df -a
Filesystem      1K-blocks   Used   Available  Use% Mounted on
/dev/cciss/c0d0p2    78361192 23186116 51130312  32% /
proc                  0       0       0       -  /proc
sysfs                 0       0       0       -  /sys
devpts                 0       0       0       -  /dev/pts
/dev/cciss/c0d0p5    24797380 22273432 1243972  95% /home
/dev/cciss/c0d0p3    29753588 25503792 2713984  91% /data
/dev/cciss/c0d0p1    295561    21531   258770   8% /boot
tmpfs                  257476     0     257476    0% /dev/shm
none                   0       0       0       - 
/proc/sys/fs/binfmt_misc
sunrpc                  0       0       0       - 
/var/lib/nfs/rpc_pipefs
```

3. Show Disk Space Usage in Human Readable Format

Have you noticed that above commands displays information in bytes, which is not readable yet all, because we are in a habit of reading the sizes in megabytes, gigabytes etc. as it makes very easy to understand and remember.

The **df** command provides an option to display sizes in **Human Readable** formats by using ‘-h’ (prints the results in human readable format (e.g., 1K 2M 3G)).

```
[root@tecmint ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/cciss/c0d0p2    75G  23G  49G  32% /
/dev/cciss/c0d0p5    24G  22G  1.2G  95% /home
/dev/cciss/c0d0p3    29G  25G  2.6G  91% /data
/dev/cciss/c0d0p1   289M  22M  253M   8% /boot
tmpfs                  252M     0   252M   0% /dev/shm
```

4. Display Information of /home File System

To see the information of only device **/home** file system in human readable format use the following command.

```
[root@tecmint ~]# df -hT /home
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/cciss/c0d0p5  ext3  24G  22G  1.2G  95% /home
```

5. Display Information of File System in Bytes

To display all file system information and usage in **1024-byte** blocks, use the option ‘**-k**’ (e.g. –block-size=1K) as follows.

```
[root@tecmint ~]# df -k
Filesystem      1K-blocks   Used   Available  Use% Mounted on
/dev/cciss/c0d0p2    78361192 23187212 51129216  32% /
/dev/cciss/c0d0p5    24797380 22273432 1243972  95% /home
/dev/cciss/c0d0p3    29753588 25503792 2713984  91% /data
/dev/cciss/c0d0p1     295561   21531   258770   8% /boot
tmpfs                  257476       0   257476   0% /dev/shm
```

6. Display Information of File System in MB

To display information of all file system usage in **MB (Mega Byte)** use the option as ‘**-m**’.

```
[root@tecmint ~]# df -m
Filesystem      1M-blocks   Used   Available  Use% Mounted on
/dev/cciss/c0d0p2    76525   22644   49931   32% /
/dev/cciss/c0d0p5    24217   21752   1215   95% /home
/dev/cciss/c0d0p3    29057   24907   2651   91% /data
/dev/cciss/c0d0p1      289     22    253   8% /boot
tmpfs                  252       0    252   0% /dev/shm
```

7. Display Information of File System in GB

To display information of all file system statistics in **GB (Gigabyte)** use the option as ‘**df -h**’.

```
[root@tecmint ~]# df -h
Filesystem      Size   Used  Avail Use% Mounted on
/dev/cciss/c0d0p2    75G   23G   49G  32% /
/dev/cciss/c0d0p5    24G   22G   1.2G  95% /home
/dev/cciss/c0d0p3    29G   25G   2.6G  91% /data
/dev/cciss/c0d0p1   289M   22M   253M  8% /boot
tmpfs                  252M       0   252M  0% /dev/shm
```

8. Display File System Inodes

Using ‘**-i**’ switch will display the information of number of used inodes and their percentage for the file system.

```
[root@tecmint ~]# df -i
Filesystem      Inodes   IUsed   IFree  IUse% Mounted on
/dev/cciss/c0d0p2  20230848 133143 20097705   1% /
/dev/cciss/c0d0p5  6403712  798613 5605099  13% /home
/dev/cciss/c0d0p3  7685440 1388241 6297199  19% /data
/dev/cciss/c0d0p1   76304      40    76264   1% /boot
tmpfs                  64369       1   64368   1% /dev/shm
```

9. Display File System Type

If you notice all the above commands output, you will see there is no file system type mentioned in the results. To check the file system type of your system use the option ‘**T**’. It will display file system type along with other information.

```
[root@tecmint ~]# df -T
Filesystem      Type  1K-blocks  Used   Available Use% Mounted on
/dev/cciss/c0d0p2  ext3    78361192 23188812 51127616 32% /
/dev/cciss/c0d0p5  ext3    24797380 22273432 1243972 95% /home
/dev/cciss/c0d0p3  ext3    29753588 25503792 2713984 91% /data
/dev/cciss/c0d0p1  ext3    295561     21531   258770 8% /boot
tmpfs           tmpfs   257476      0   257476 0% /dev/shm
```

10. Include Certain File System Type

If you want to display certain file system type use the ‘**-t**’ option. For example, the following command will only display **ext3** file system.

```
[root@tecmint ~]# df -t ext3
Filesystem      1K-blocks  Used   Available Use% Mounted on
/dev/cciss/c0d0p2  78361192 23190072 51126356 32% /
/dev/cciss/c0d0p5  24797380 22273432 1243972 95% /home
/dev/cciss/c0d0p3  29753588 25503792 2713984 91% /data
/dev/cciss/c0d0p1  295561     21531   258770 8% /boot
```

11. Exclude Certain File System Type

If you want to display file system type that doesn’t belongs to **ext3** type use the option as ‘**-x**’. For example, the following command will only display other file systems types other than **ext3**.

```
[root@tecmint ~]# df -x ext3
Filesystem      1K-blocks  Used   Available Use% Mounted on
tmpfs           257476      0   257476 0% /dev/shm
```

12. Display Information of df Command.

Using ‘**–help**’ switch will display a list of available option that are used with **df** command.

```
[root@tecmint ~]# df --help
Usage: df [OPTION]... [FILE]...
Show information about the file system on which each FILE resides,
or all file systems by default.
Mandatory arguments to long options are mandatory for short options too.
-a, --all          include dummy file systems
-B, --block-size=SIZE use SIZE-byte blocks
-h, --human-readable print sizes in human readable format (e.g., 1K 234M 2G)
-H, --si           likewise, but use powers of 1000 not 1024
-i, --inodes       list inode information instead of block usage
-k                like --block-size=1K
-l, --local        limit listing to local file systems
--no-sync         do not invoke sync before getting usage info (default)
-P, --portability  use the POSIX output format
```

```
--sync           invoke sync before getting usage info
-t, --type=TYPE    limit listing to file systems of type TYPE
-T, --print-type     print file system type
-x, --exclude-type=TYPE  limit listing to file systems not of type TYPE
-v                  (ignored)
--help      display this help and exit
--version   output version information and exit
SIZE may be (or may be an integer optionally followed by) one of following:
kB 1000, K 1024, MB 1000*1000, M 1024*1024, and so on for G, T, P, E, Z, Y.
Report bugs to <bug-coreutils@gnu.org>.
```

How Do I Access or Mount Windows/USB NTFS Partition in RHEL/CentOS/Fedora

by [Ravi Saive](#) | Published: February 7, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

Sometimes it may happens in some stage, you may have to access data on a **Windows** partition, **USB** device or any similar device. Today most of the modern Linux systems automatically recognize and mount any disks.

However, in some occasions where you may required to configure your system manually to mount ntfs partitions on your Linux system. Specially when you are using dual boot operating environment. Fortunately, this process is not so complicated task its just very fairly straight forward.



How to mount Windows NTFS Partition in Linux

This article explains you on how to access or mount **Windows XP, Vista NTFS** or **USB** filesystem using the ‘**mount**’ command in **RHEL/CentOS/Fedora** systems.

How to Mount Windows NTFS Partition in Linux

First you need to enable **EPEL (Extra Packages for Enterprise Linux)** Repository. You may refer the article on how to [enable EPEL Repository](#) under **RHEL, CentOS** and **Fedora** systems.

To mount any **NTFS** based filesystem, you need to install a tool called **NTFS3G**. Before heading up for installation let's understand **NTGS3G**.

What is NTFS3G

NTFS3G is an open source cross-platform, stable, **GPL** licensed, **POSIX**, **NTFS R/W** driver used in **Linux**. It provides safe handling of **Windows NTFS** file systems viz create, remove, rename, move files, directories, hard links, etc.

Once **EPEL** is installed and enabled, let's install **ntfs-3g** package using the below command with root user.

```
# yum -y install ntfs-3g
```

Fuse Install

Next, install and load **FUSE** driver to mount detected devices with below command. FUSE module is included in the kernel itself in version 2.6.18-164 or newer.

```
# yum install fuse  
# modprobe fuse
```

Identify NTFS Partition

Once fuse module is loaded, type below command to find out **NTFS Partitions in Linux**.

```
# fdisk -l  
Device Boot      Start      End      Blocks   Id  System  
/dev/sdb1            1       21270     7816688   b  W95 FAT32
```

Mount NTFS partition

First create a mount point to mount the **NTFS** partition.

```
# mkdir /mnt/nts
```

Simply run the following command to mount the partition. Replace **sda1** with your actual partition found.

```
# mount -t ntfs-3g /dev/sda1 /mnt/nts
```

Once it's mounted on **/mnt/ntfs**, you may use regular Linux [ls -l](#) command to list the content of mounted filesystem.

```
[root@tecmint ntfs]# ls -l  
total 27328  
drwx----- 2 root root 16384 Sep  2 19:37 Cert  
drwx----- 20 root root 16384 Aug 24 2011 club_application  
drwx----- 6 root root 16384 Aug 11 15:37 docs  
drwx----- 7 root root 16384 Jul 31 2012 Downloads  
drwx----- 2 root root 16384 Dec 10 20:28 images  
-rwxr-xr-x 1 root root 31744 Jan 18 00:29 Material List.doc
```

If you want to make mount point permanent at the boot time, then simple add the following line at the end of **/etc/fstab** file. This will remain as permanent.

```
/dev/sda1      /mnt/usb      ntfs-3g      defaults      0      0
```

Unmount NTFS Partition

Simply, use the following command to unmount the mounted partition.

```
# umount /mnt/usb
```

How to Mount and Unmount an ISO Image in RHEL/CentOS/Fedora and Ubuntu

by [Ravi Saive](#) | Published: February 4, 2013 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

An **ISO** image or **.iso** (**I**nternational **O**rganization **f**or **S**tandardization) file is an archive file that contains a disk image called **ISO 9660** file system format. Every **ISO** file have **.ISO** extension has defined format name taken from the **ISO 9660** file system and specially used with **CD/DVD** Rom's. In simple words an iso file is a disk image.



mount and unmount iso images in linux

I have seen most of the Linux operating system that we download from the internet are **.ISO** format. Typically an **ISO** image contains installation of software's such as, operating system installation, games installation or any other applications. Sometimes it happens that we need to access files and view content from these **ISO** images, but without wasting disk space and time in burning them on to **CD/DVD**.

This article describes how to **mount** and **unmount** an ISO image on a Linux Operating system to access and list the content of files.

How to Mount an ISO Image

To mounting an **ISO** image on Linux (**RedHat, CentOS, Fedora or Ubuntu**), you must be logged in as “**root**” user or switch to “**sudo**” and run the following commands from a terminal to create a mount point.

```
# mkdir /mnt/iso  
OR
```

```
$ sudo mkdir /mnt/iso
```

Once you created mount point, use the “**mount**” command to mount an iso file called “**Fedora-18-i386-DVD.iso**“.

```
# mount -t iso9660 -o loop /home/tecmint/Fedora-18-i386-DVD.iso /mnt/iso/  
OR  
$ sudo mount -t iso9660 -o loop /home/tecmint/Fedora-18-i386-DVD.iso  
/mnt/iso/
```

After the **ISO** image mounted successfully, go the mounted directory at **/mnt/iso** and list the content of an ISO image. It will only mount in read-only mode, so none of the files can be modified.

```
# cd /mnt/iso  
# ls -l
```

You will see the list of files of an ISO image, that we have mounted in the above command. For example, the directory listing of an **Fedora-18-i386-DVD.iso** image would look like this.

```
total 16  
drwxrwsr-x 3 root 101737 2048 Jan 10 01:00 images  
drwxrwsr-x 2 root 101737 2048 Jan 10 01:00 isolinux  
drwxrwsr-x 2 root 101737 2048 Jan 10 01:00 LiveOS  
drwxrwsr-x 28 root 101737 4096 Jan 10 00:38 Packages  
drwxrwsr-x 2 root 101737 4096 Jan 10 00:43 repodata  
-r--r--r-- 1 root root 1538 Jan 10 01:00 TRANS.TBL
```

How to Unmount an ISO Image

Simply run the following command from the terminal either “**root**” or “**sudo**” to unmount an mounted ISO image.

```
# umount /mnt/iso  
OR  
$ sudo umount /mnt/iso
```

Where Options

1. **-t** : This argument is used to indicate the given filesystem type.
2. **ISO 9660** : It describes standard and default filesystem structure to be used on CD/DVD ROMs.
3. **-o** : Options are necessary with a -o argument followed by a separated comma string of options.
4. **loop**: The loop device is a pseudo-device that often used for mounting CD/DVD ISO image and makes those files accessible as a block device.

8 Practical Examples of Linux “Touch” Command

by [Ravi Saive](#) | Published: December 11, 2012 | Last Updated: January 13, 2017

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In **Linux** every single file is associated with timestamps, and every file stores the information of last access time, last modification time and last change time. So, whenever we create new file, access or modify an existing file, the timestamps of that file automatically updated.



Linux Touch Command Examples

In this article we will cover some useful practical examples of Linux **touch command**. The **touch command** is a standard program for **Unix/Linux** operating systems, that is used to create, change and modify timestamps of a file. Before heading up for touch command examples, please check out the following options.

Touch Command Options

1. **-a**, change the access time only
2. **-c**, if the file does not exist, do not create it
3. **-d**, update the access and modification times
4. **-m**, change the modification time only
5. **-r**, use the access and modification times of file
6. **-t**, creates a file using a specified time

1. How to Create an Empty File

The following touch command creates an empty (zero byte) new file called **sheena**.

```
# touch sheena
```

2. How to Create Multiple Files

By using touch command, you can also create more than one single file. For example the following command will create 3 files named, **sheena**, **meena** and **leena**.

```
# touch sheena meena leena
```

3. How to Change File Access and Modification Time

To change or update the last access and modification times of a file called **leena**, use the **-a** option as follows. The following command sets the current time and date on a file. If the **leena** file does not exist, it will create the new empty file with the name.

```
# touch -a leena
```

The most popular Linux commands such as [find command](#) and [ls command](#) uses timestamps for listing and finding files.

4. How to Avoid Creating New File

Using **-c** option with touch command avoids creating new files. For example the following command will not create a file called **leena** if it does not exists.

```
# touch -c leena
```

5. How to Change File Modification Time

If you like to change the only modification time of a file called **leena**, then use the **-m** option with touch command. Please note it will only updates the last modification times (not the access times) of the file.

```
# touch -m leena
```

6. Explicitly Set the Access and Modification times

You can explicitly set the time using **-c** and **-t** option with touch command. The format would be as follows.

```
# touch -c -t YYDDHHMM leena
```

For example the following command sets the access and modification date and time to a file **leena** as **17:30 (17:30 p.m.) December 10** of the current year (**2012**).

```
# touch -c -t 12101730 leena
```

Next verify the access and modification time of file **leena**, with **ls -l** command.

```
# ls -l  
total 2  
-rw-r--r--. 1 root      root    0 Dec 10 17:30 leena
```

7. How to Use the time stamp of another File

The following touch command with **-r** option, will update the time-stamp of file **meena** with the time-stamp of **leena** file. So, both the file holds the same time stamp.

```
# touch -r leena meena
```

8. Create a File using a specified time

If you would like to create a file with specified time other than the current time, then the format should be.

```
# touch -t YYMMDDHHMM.SS tecmint
```

For example the below command touch command with **-t** option will gives the **tecmint** file a time stamp of **18:30:55 p.m. on December 10, 2012**.

```
# touch -t 201212101830.55 tecmint
```

We've almost covered all the options available in the touch command for more options use "**man touch**". If we've still missed any options and you would like to include in this list, please update us via comment box

10 Wget (Linux File Downloader) Command Examples in Linux

by [Narad Shrestha](#) | Published: October 3, 2012 | Last Updated: January 3, 2015

Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](#) | [4 Free Shell Scripting eBooks](#)

In this post we are going to review **wget** utility which retrieves files from **World Wide Web (WWW)** using widely used protocols like **HTTP**, **HTTPS** and **FTP**. **Wget** utility is freely available package and license is under **GNU GPL License**. This utility can be install any **Unix-like** Operating system including **Windows** and **MAC OS**. It's a non-interactive command line tool. Main feature of **Wget** of it's robustness. It's designed in such way so that it works in slow or unstable network connections. **Wget** automatically start download where it was left off in case of network problem. Also downloads file recursively. It'll keep trying until file has be retrieved completely.



10 Linux Wget Command Examples

First, check whether **wget** utility is already installed or not in your Linux box, using following command.

```
# rpm -qa wget  
wget-1.12-1.4.el6.i686
```

Please install it using **YUM** command in case **wget** is not installed already or you can also download binary package at <http://ftp.gnu.org/gnu/wget/>.

```
# yum -y install wget
```

The **-y** option used here, is to prevent confirmation prompt before installing any package. For more **YUM** command examples and options read the article on [20 YUM Command Examples for Linux Package Management](#).

1. Single file download

The command will download single file and stores in a current directory. It also shows **download progress, size, date and time** while downloading.

```
# wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
--2012-10-02 11:28:30--  http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
Resolving ftp.gnu.org... 208.118.235.20, 2001:4830:134:3::b
Connecting to ftp.gnu.org|208.118.235.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 446966 (436K) [application/x-gzip]
Saving to: wget-1.5.3.tar.gz
100%[=====] 446,966      60.0K/s   in 7.4s
2012-10-02 11:28:38 (58.9 KB/s) - wget-1.5.3.tar.gz
```

2. Download file with different name

Using **-O (uppercase)** option, downloads file with different file name. Here we have given **wget.zip** file name as show below.

```
# wget -O wget.zip http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
--2012-10-02 11:55:54--  http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
Resolving ftp.gnu.org... 208.118.235.20, 2001:4830:134:3::b
Connecting to ftp.gnu.org|208.118.235.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 446966 (436K) [application/x-gzip]
Saving to: wget.zip
100%[=====] 446,966      60.0K/s   in 7.5s
2012-10-02 11:56:02 (58.5 KB/s) - wget.zip
```

3. Download multiple file with http and ftp protocol

Here we see how to download multiple files using **HTTP** and **FTP** protocol with **wget** command at ones.

```
# wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
ftp://ftp.gnu.org/gnu/wget/wget-1.10.1.tar.gz.sig
--2012-10-02 12:11:16--  http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
Resolving ftp.gnu.org... 208.118.235.20, 2001:4830:134:3::b
Connecting to ftp.gnu.org|208.118.235.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 446966 (436K) [application/x-gzip]
Saving to: wget-1.5.3.tar.gz
100%[=====] 446,966      56.7K/s   in 7.6s
2012-10-02 12:11:29 (57.1 KB/s) - wget-1.5.3.tar.gz
--2012-10-02 12:11:29--  ftp://ftp.gnu.org/gnu/wget/wget-1.10.1.tar.gz.sig
=> wget-1.10.1.tar.gz.sig
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
```

```

==> TYPE I ... done. ==> CWD (1) /gnu/wget ... done.
==> SIZE wget-1.10.1.tar.gz.sig ... 65
==> PASV ... done. ==> RETR wget-1.10.1.tar.gz.sig ... done.
Length: 65 (unauthoritative)
100%[=====] 65 --.-K/s in 0s
2012-10-02 12:11:33 (2.66 MB/s) - wget-1.10.1.tar.gz.sig
FINISHED --2012-10-02 12:11:33--
Downloaded: 2 files, 437K in 7.6s (57.1 KB/s)

```

4. Read URL's from a file

You can store number of **URL**'s in text file and download them with **-i** option. Below we have created **tmp.txt** under wget directory where we put series of **URL**'s to download.

```

# wget -i /wget/tmp.txt
--2012-10-02 12:34:12-- http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
ftp://ftp.gnu.org/gnu/wget/wget-1.10.1.tar.gz.sig
Resolving ftp.gnu.org... 208.118.235.20, 2001:4830:134:3::b
Connecting to ftp.gnu.org|208.118.235.20|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 446966 (436K) [application/x-gzip]
Saving to: wget-1.10.1.tar.gz.sig
100%[=====]
=====] 446,966 35.0K/s in 10s
2012-10-02 12:34:23 (42.7 KB/s) - wget-1.10.1.tar.gz.sig
--2012-10-02 12:34:23--
http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-
LiveDVD.iso
Resolving mirrors.hns.net.in... 111.91.91.34, 2401:4800::111:91:91:34
Connecting to mirrors.hns.net.in|111.91.91.34|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1761607680 (1.6G) [application/octet-stream]
Saving to: CentOS-6.3-x86_64-LiveDVD.iso
45%[=====]
] 1,262,000 51.6K/s eta 8h 17m

```

5. Resume uncompleted download

In case of big file download, it may happen sometime to stop download in that case we can resume download the same file where it was left off with **-c** option. But when you start download file without specifying **-c** option **wget** will add **.1** extension at the end of file, considering as a fresh download. So, it's good practice to add **-c** switch when you download big files.

```

# wget -c http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-
LiveDVD.iso
--2012-10-02 12:46:57--
http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-
LiveDVD.iso
Resolving mirrors.hns.net.in... 111.91.91.34, 2401:4800::111:91:91:34
Connecting to mirrors.hns.net.in|111.91.91.34|:80... connected.
HTTP request sent, awaiting response... 206 Partial Content

```

```
Length: 1761607680 (1.6G), 1758132697 (1.6G) remaining [application/octet-stream]
Saving to: CentOS-6.3-x86_64-LiveDVD.iso
51% [=====]
] 3,877,262   47.0K/s eta 10h 27m ^
```

6. Download file with appended .1 in file name

When you start download without **-c** option **wget** add **.1** at the end of file and start with fresh download. If **.1** already exist **.2** append at the end of file.

```
# wget http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-LiveDVD.iso
--2012-10-02 12:50:49--
http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-LiveDVD.iso
Resolving mirrors.hns.net.in... 111.91.91.34, 2401:4800::111:91:91:34
Connecting to mirrors.hns.net.in|111.91.91.34|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1761607680 (1.6G) [application/octet-stream]
Saving to: CentOS-6.3-x86_64-LiveDVD.iso.1
18% [=====]
] 172,436      59.2K/s
```

See the example files with **.1** extension appended at the end of the file.

```
# ls -l CentOS*
-rw-r--r--. 1 root root 3877262 Oct  2 12:47 CentOS-6.3-x86_64-LiveDVD.iso
-rw-r--r--. 1 root root 181004 Oct  2 12:50 CentOS-6.3-x86_64-LiveDVD.iso.1
```

7. Download files in background

With **-b** option you can send download in background immediately after download start and logs are written in **/wget/log.txt** file.

```
# wget -b /wget/log.txt ftp://ftp.iinet.net.au/debian/debian-cd/6.0.5/i386/iso-dvd/debian-6.0.5-i386-DVD-1.iso
Continuing in background, pid 3550.
```

8. Restrict download speed limits

With Option **--limit-rate=100k**, the download speed limit is restricted to 100k and the logs will be created under **/wget/log.txt** as shown below.

```
# wget -c --limit-rate=100k /wget/log.txt
ftp://ftp.iinet.net.au/debian/debian-cd/6.0.5/i386/iso-dvd/debian-6.0.5-i386-DVD-1.iso
/wget/log.txt: Scheme missing.
--2012-10-02 13:16:21--  ftp://ftp.iinet.net.au/debian/debian-cd/6.0.5/i386/iso-dvd/debian-6.0.5-i386-DVD-1.iso
=> debian-6.0.5-i386-DVD-1.iso
```

```
esolving ftp.iinet.net.au... 203.0.178.32
Connecting to ftp.iinet.net.au|203.0.178.32|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done. ==> PWD ... done.
==> TYPE I ... done. ==> CWD (1) /debian/debian-cd/6.0.5/i386/iso-dvd ...
done.
==> SIZE debian-6.0.5-i386-DVD-1.iso ... 4691312640
==> PASV ... done. ==> REST 2825236 ... done.
==> RETR debian-6.0.5-i386-DVD-1.iso ... done.
Length: 4688487404 (4.4G), 4685662168 (4.4G) remaining (unauthoritative)
0% [
] 3,372,160 35.5K/s eta 28h 39m
```

9. Restricted FTP and HTTP downloads with username and password

With Options **--http-user=username**, **--http-password=password** & **--ftp-user=username**, **--ftp-password=password**, you can download password restricted **HTTP** or **FTP** sites as shown below.

```
# wget --http-user=narad --http-password=password
http://mirrors.hns.net.in/centos/6.3/isos/x86_64/CentOS-6.3-x86_64-
LiveDVD.iso
# wget --ftp-user=narad --ftp-password=password
ftp://ftp.iinet.net.au/debian/debian-cd/6.0.5/i386/iso-dvd/debian-6.0.5-i386-
DVD-1.iso
```

10. Find wget version and help

With Options **--version** and **--help** you can view **version** and **help** as needed.

```
# wget --version
# wget --help
```

In this article we have covered Linux **wget command** with options for daily administrative task. Do **man wget** if you wan to know more about it. Kindly share through our comment box or if we've missed out anything, do let us know.

REFERENCES

www.tecmint.com

www.howtogeek.com

www.binarytides.com

<https://www.ibm.com/developerworks/library/l-linuxboot/index.html>

<https://www.ibm.com/developerworks/library/l-linuxuniversal/index.html>