LINUX - THE PENGUIN MARCHES ON



Patrick McClanahan
San Joaquin Delta College

LibreTexts™

San Joaquin Delta College

Linux - The Penguin Marches On

Professor Patrick McClanahan
Professor Tobi West
Professor Skip Berry

This text is disseminated via the Open Education Resource (OER) LibreTexts Project (https://LibreTexts.org) and like the thousands of other texts available within this powerful platform, it is freely available for reading, printing, and "consuming."

The LibreTexts mission is to bring together students, faculty, and scholars in a collaborative effort to provide an accessible, and comprehensive platform that empowers our community to develop, curate, adapt, and adopt openly licensed resources and technologies; through these efforts we can reduce the financial burden born from traditional educational resource costs, ensuring education is more accessible for students and communities worldwide.

Most, but not all, pages in the library have licenses that may allow individuals to make changes, save, and print this book. Carefully consult the applicable license(s) before pursuing such effects. Instructors can adopt existing LibreTexts texts or Remix them to quickly build course-specific resources to meet the needs of their students. Unlike traditional textbooks, LibreTexts' web based origins allow powerful integration of advanced features and new technologies to support learning.



LibreTexts is the adaptable, user-friendly non-profit open education resource platform that educators trust for creating, customizing, and sharing accessible, interactive textbooks, adaptive homework, and ancillary materials. We collaborate with individuals and organizations to champion open education initiatives, support institutional publishing programs, drive curriculum development projects, and more.

The LibreTexts libraries are Powered by NICE CXone Expert and was supported by the Department of Education Open Textbook Pilot Project, the California Education Learning Lab, the UC Davis Office of the Provost, the UC Davis Library, the California State University Affordable Learning Solutions Program, and Merlot. This material is based upon work supported by the National Science Foundation under Grant No. 1246120, 1525057, and 1413739.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation nor the US Department of Education.

Have questions or comments? For information about adoptions or adaptions contact info@LibreTexts.org or visit our main website at https://LibreTexts.org.

This text was compiled on 11/25/2025

TABLE OF CONTENTS

Title Page

Licensing

ASCCC OERI

Module 01: Back to the Basics

- 01-A.1: Introduction to Linux
- 01-A.2: Open Source Software
- 01-A.3: Free Software vs Open Source
- 01-A.4: Different FOSS licensing...
- 01-A.5: The GNU Project
- 01-A.6: Unix - What's the Connection?
- 01-A.7: Finally - Linux
- 01-A.8: Advantages and Disadvantages of Linux
- 01-A.9: Distro List
- 01-A.10: How is Linux Used?
- 01-B.11: How the Shell and CLI Work
- 01-B.12: CLI - The Good and the Bad
- 01-B.13: Shells: What and Why
- 01-B.14: Doing bash
- 01-B.15: Bash Command How To
- 01-B.16: Let's Look at the Basics
- 01-B.17: File Basics: More Later
- 01-B.18: Who is superuser?
- 01-B.19: Leftovers, What Else Do We Need?
- 01-C.20: Help! Where can I find help in Linux?
- 01-C.21: Linux man pages
- 01-C.22: Man Page Format
- 01-C.23: Additional Linux System Help Choices
- 01-C.24: Sources of Online Linux Documentation
- 01-C.25: Asynchronous Linux Help Resources

Module 02: User & Group Administration

- 02-A.1: Users, and How to Acquire Superuser Power
- 02-A.2: Who Is the Superuser?
- 02-A.3: su: The Command
- 02-A.4: Other Options for Administrative Privileges
- 02-B.5: Users: Create, Modify, and Delete
- 02-B.6: Users: Create, Modify, and Delete (cont'd)
- 02-B.7: Users: Create, Modify and Delete (continued)
- 02-B.8: Users: Create, Modify, and Delete - passwd Command
- 02-B.9: Users: Create, Modify, and Delete - etc passwd and etc shadow
- 02-B.10: Users: Create, Modify, and Delete - Edit /etc/passwd File
- 02-C.11: Groups: Create, Modify, and Delete

- 02-C.12: Groups: Create, Modify, and Delete - Modify Group
- 02-D.13: User and Group Queries
- 02-E.14: User Account Profile Configuration
- 02-E.15: Example: Bash RC file
- 02-E.16: System Wide User Profiles
  - 02-E.16.1: System Wide User Profiles: /etc/profile
  - 02-E.16.2: System Wide User Profiles: The /etc/profile.d Directory
  - 02-E.16.3: System Wide User Profiles: The /etc/bashrc File

Module 03: Permission and Ownership Management

- 03-A.1: Modify Permissions: File and Directory
- 03-A.2: Modify Permissions: File and Directory (continued)
- 03-B.3: Modify Ownership of Files and Directories
- 03-C.4: Special Permission Types - The setuid Bit
  - 03-C.4.1: Special Permission Types - The setgid Bit
  - 03-C.4.2: Special Permission Types - The Sticky Bit
- 03-C.5: File Attributes
- 03-D.6: Permission Issues and How to Troubleshoot

Module 04: Managing Linux Storage

- 04-A.1: Storage Devices
- 04-A.2: Linux File Systems
- 04-A.3: Inodes
- 04-A.4 Journaling File Systems
- 04-A.5: Linux Virtual File System
- 04-A.6: Partitions and Labels
  - 04-A.6.1: Partitions and Labels continued
- 04-A.7: Disk Setup Process - fdisk
  - 04-A.7.1: Disk Setup Process - parted
  - 04-A.7.2: Disk Setup Process - mkfs & mount
- 04-A.8: The dev Directory
- 04-B.9: Logical Volume Management
  - 04-B.9.1: LVM and other stuff
  - 04-B.9.2: LVM Tools
- 04-C.10: Mounting Linux File Systems
- 04-D.11: File System Management
  - 04-D.11.1: File System Management - ext Filesystem Tools
  - 04-D.11.2: File System Management - xfs Tools
- 04-E.12: Linux Directory Structure
  - 04-E.12.1: Linux Directory Structure - Hierarchy
  - 04-E.12.2: Linux Directory Structure - Home Directory
  - 04-E.12.3: Linux Directory Structure - Paths
- 04-F.13: Storage Issues - Find and Fix
  - 04-F.13.1: Storage Issues - IO Scheduling
  - 04-F.13.2: Storage Issues - More IO Scheduling
  - 04-F.13.3: Storage Issues - iostat, ioping

- 04-F.13.4: Storage Issues - Quota Management

Module 05: File and Directory Management

- 05-A.1: Creating and Editing Text files
- 05-A.2: Creating and Editing Text files - Vim Editor
  - 05-A.2.1: Vim - Insert Mode
  - 05-A.2.2: Vim - Visual Mode
  - 05-A.2.3: Vim - Command-Line Mode
- 05-A.3: Nano Editor
- 05-A.4: Gedit and More
- 05-B.5: Searching for Files on Linux
  - 05-B.5.1: Searching for Files on Linux - find Command
  - 05-B.5.2: Searching for Files on Linux - which Command
  - 05-B.5.3: Searching for Files on Linux - whereis Command
- 05-C.6: Working with Files and Directories - cat Command
  - 05-C.6.1: Working with Files and Directories - head/tail Commands
  - 05-C.6.2: Working with Files and Directories - less/more Command
  - 05-C.6.3: Working with Files and Directories - cp/mv Command
  - 05-C.6.4: Working with Files and Directories - rm/touch Command
  - 05-C.6.5: Working with Files and Directories - ls/rmidir/mkdir Command
- 05-D.7: Handling Text Files - echo/printf
  - 05-D.7.1: Handling Text Files - tr/wc Command
  - 05-D.7.2: Handling Text Files - sort/diff Commands
  - 05-D.7.3: Handling Text Files - cut/paste
  - 05-D.7.4: Handling Text Files - grep Command
  - 05-D.7.5: Handling Text Files - awk Command
  - 05-D.7.6: Handling Text Files - sed Command
  - 05-D.7.7: Handling Text Files - ln Command
- 05-E.8: File Output Manipulation
  - 05-E.8.1: File Output Manipulation - pipe
  - 05-E.8.2: File Output Manipulation - xargs
  - 05-E.8.3: File Output Manipulation - tee & /dev/null

Module 06: Kernel Module Management

- 06-A.1: Examine the Linux Kernel
- 06-A.2: Linux Kernel
- 06-A.3: Linux Kernel Subsystem
- 06-B.4: Kernel Modules
- 06-B.5: Kernel Module Management - lsmod/modinfo Command
  - 06-B.5.1: Kernel Module Management - insmod/rmmod Command
  - 06-B.5.2: Kernel Module Management - modprobe/depmod Command
- 06-B.6: Kernel Modules - Configuration
- 06-B.7: Kernel Modules - Kernel Parameters
- 06-C.8: Kernel Module Monitoring
- 06-C.9: The dmesg Command

Module 07: Linux Boot Process

- 07-A.1: Configuring Boot Components
- 07-A.2: Configuring Boot Components (continued)
- 07-A.3: Configuring initrd
- 07-A.4: Boot Process
- 07-B.5: GRUB2 - Installation
- 07-B.6: GRUB2 - Configuration

Module 08: How to Manage System Components

- 08-A.1: Localization Options - Configuration
- 08-A.2: Localization - date/timdatectl Command
- 08-A.3: Localization - localectl Command
- 08-B.4: GUI Configuration
- 08-B.5: GUI Desktop Environments
- 08-B.6: Remote Desktop Access
- 08-B.7: Remote Console - SSH/X Forwarding
- 08-C.8: Managing Linux Services
  - 08-C.8.1: Managing Linux Services - systemctl
  - 08-C.8.2: Managing Linux Services - hostnamectl
  - 08-C.8.3: Managing Linux Services - SysVinit
  - 08-C.8.4: Managing Linux Services - inittab & init.d
- 08-D.9: Process Troubleshooting
  - 08-D.9.1: Process Troubleshooting - ps command
  - 08-D.9.2: Process Troubleshooting - top command
  - 08-D.9.3: Process Troubleshooting - lsof Command
  - 08-D.9.4: Process Troubleshooting - Process Priorities
  - 08-D.9.5: Process Troubleshooting - Background / Foreground Processes
  - 08-D.9.6: Process Troubleshooting - nohup and kill Commands
- 08-E.10: CPU and Memory Troubleshooting
  - 08-E.10.1: CPU and Memory Troubleshooting - sar & Memory Issues
  - 08-E.10.2: CPU and Memory Troubleshooting - free / vmstat Commands
  - 08-E.10.3: CPU and Memory Troubleshooting - OOM & swap

Module 09: Managing Linux Devices

- 09-A.1: Supported Linux Devices
  - 09-A.1.1: Supported Linux Devices (cont'd)
  - 09-A.1.2: Supported Linux Devices (continued)
- 09-B.2: Configuring Linux Devices
  - 09-B.2.1: Configuring Linux Devices - Printing
- 09-C.3: Monitoring Devices in Linux
- 09-D.4: Troubleshooting: Hardware Issues

Module 10: Network Management

- 10-A.1: TCP/IP Fundamentals
  - 10-A.1.1: TCP/IP Fundamentals - Identities
  - 10-A.1.2: TCP/IP Fundamentals - Addressing

- 10-A.1.3: TCP/IP Fundamentals - Ports and Segments
- 10-B.2: Server Roles in Linux
  - 10-B.2-3: Server Roles in Linux (continued)
  - 10-B.2.1: Server Roles in Linux - Continued
  - 10-B.2.4: Server Roles in Linux (continued)
- 10-C.3: Network Connection
- 10-D.4: Static VS Dynamic IP Addresses
- 10-D.5: DHCP Configuration
- 10-D.6: Name Resolution
- 10-E.7: Cloud Configuration
- 10-E.8: The Cloud and Virtualization
- 10-E.9: Cloud Bootstrapping and Storage
- 10-E.10: Configuring Networks and Virtualization Tools
- 10-F.11: How to Troubleshoot Networking Problems
  - 10-F.11.1: Network Troubleshooting Commands - netstat / ss
  - 10-F.11.2: Network Troubleshooting Commands - dig / nslookup / host
  - 10-F.11.3: Network Troubleshooting Commands - ip / route / nmap
  - 10-F.11.4: Network Troubleshooting Commands - wireshark
  - 10-F.11.5: Network Troubleshooting Commands - iftop / iperf / mtr
  - 10-F.11.6: Network Troubleshooting Commands - arp / whois

Module 11: Package Management

- 11-A.1: Who are the Package Managers
- 11-B.2: rpm / yum How To
- 11-C.3: dpkg / apt How To
- 11-D.4: Repository Configuration
- 11-E.5: Other Software Sources
- 11-F.6: Software Builds from Source
- 11-G.7: Software Dependency Troubleshooting

Module 12: Linux Systems Security

- 12-A.1: Best Cybersecurity Practices
  - 12-A.1.1: Best Cybersecurity Practices - Continued
- 12-A.2: chroot Jail
- 12-A.3: Encryption in Linux
- 12-A.4: Security Best Practices
- 12-B.5: Identity and Access Management
- 12-B.6: Pluggable Authentication Modules
- 12-B.7: TTY Security & PKI
- 12-B.8: OpenSSL / VPN
- 12-B.9: Authentication Issues
- 12-C.10: SELinux Configuration
- 12-C.11: AppArmor
- 12-D.12: Firewall Configuration in Linux
- 12-D.13: Access Control Lists / iptables
- 12-D.14: UFW & firewalld
- 12-D.15: Netfilter / IP Forwarding
- 12-E.16: Linux Logging

- 12-E.17: journalctl & last Commands
- 12-F.18: Backup and Restore in Linux
- 12-F.19: tar / cpio / dd
- 12-F.20: mirrorvg / rsync / scp
- 12-F.21: File Compression
- 12-F.22: Integrity Check and Checksums

Module 13: Working with Bash Scripts

- 13-A.1: How to Customize the Bash Shell
- 13-A.2: env / alias / PATH / time
- 13-B.3: Fundamentals of Linux Scripts
- 13-B.4: Shell Operators
- 13-B.5: Strings / Arrays / Functions
- 13-C.6: Writing and Running Linux Scripts
- 13-C.7: Shell Expansion / Variable Substitution / Command Substitution
- 13-C.8: Positional Parameters / exec Command / source Command
- 13-C.9: Script Permissions / Script Extensions
- 13-D.10: Shell Control Statements
- 13-D.11: Script Loops

Module 14: Task Automation

- 14-A.1: Job Scheduling - at Command
- 14-A.2: Cron
- 14-B.3: Git: Version Control
- 14-C.4: Orchestration

Module 15: Linux Installation

- 15-A.1: Linux Installation
- 15-A.2: Installation: Methods and Preparation

Ancillary Materials

- Lab Exercises
- Powerpoint Presentations

Index

Glossary

Detailed Licensing

Licensing

*A detailed breakdown of this resource's licensing can be found in **Back Matter/Detailed Licensing**.*

An Open Educational Resource Supported by the Academic Senate for California Community Colleges Open Educational Resources Initiative

The Academic Senate for California Community Colleges Open Educational Resources Initiative (OERI) was funded by the California legislature in trailer bill language during the summer of 2018. The OERI's mission is to reduce the cost of educational resources for students by expanding the availability and adoption of high quality Open Educational Resources (OER). The OERI facilitates and coordinates the curation and development of OER texts, ancillaries, and support systems. In addition, the OERI supports local OER implementation efforts through the provision of professional development, technical support, and technical resources.

The information in this resource is intended solely for use by the user who accepts full responsibility for its use. Although the author(s) and ASCCC OERI have made every effort to ensure that the information in this resource is accurate, openly licensed, and accessible at press time, the author(s) and ASCCC OERI do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause.

Please bring all such errors and changes to the resource to the attention of

Academic Senate for California Community Colleges
One Capitol Mall, Suite 230
Sacramento

CHAPTER OVERVIEW

Module 01: Back to the Basics

01-A.1: Introduction to Linux
01-A.2: Open Source Software
01-A.3: Free Software vs Open Source
01-A.4: Different FOSS licensing...
01-A.5: The GNU Project
01-A.6: Unix - What's the Connection?
01-A.7: Finally - Linux
01-A.8: Advantages and Disadvantages of Linux
01-A.9: Distro List
01-A.10: How is Linux Used?
01-B.11: How the Shell and CLI Work
01-B.12: CLI - The Good and the Bad
01-B.13: Shells: What and Why
01-B.14: Doing bash
01-B.15: Bash Command How To
01-B.16: Let's Look at the Basics
01-B.17: File Basics: More Later
01-B.18: Who is superuser?
01-B.19: Leftovers, What Else Do We Need?
01-C.20: Help! Where can I find help in Linux?
01-C.21: Linux man pages
01-C.22: Man Page Format
01-C.23: Additional Linux System Help Choices
01-C.24: Sources of Online Linux Documentation
01-C.25: Asynchronous Linux Help Resources

This page titled Module 01: Back to the Basics is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.1: Introduction to Linux

Module Introduction

Are you ready for this? You are about to step into the amazing world of the Linux operating system. What you are undertaking is an adventure into the vast world known as Linux. There is a lot of information and a lot to grasp, so in an attempt to help you, we are going to start slow and build as we go. We are going to start with some fundamental ideas, which will help you as we move along.

Objectives of this Module

In this lesson, you will:

- Understand the high-level concepts that guide the Linux operating system development.
- Use basic Linux shell commands to begin using and learning about the command-line interface (CLI).
- Learn where to find the various resources for help with using the Linux operating system.

This page titled 01-A.1: Introduction to Linux is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.2: Open Source Software

What is "open source"?

The term "open source" refers to something people can modify and share because it's design is publicly accessible.

The term originated in the context of software development to designate a specific approach to creating computer programs. Today, however, "open source" designates a broader set of values—what we call "the open source way." Open source projects, products, or initiatives embrace and celebrate principles of open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community-oriented development.

What is open source software?

Open source software is software with source code that anyone can inspect, modify, and enhance.

"Source code" is the part of software that most computer users don't ever see; it's the code computer programmers can manipulate to change how a piece of software—a "program" or "application"—works. Programmers who have access to a computer program's source code can improve that program by adding features to it or fixing parts that don't always work correctly.

Derived from: "What is open source?" by David Both is licensed under CC BY-SA 3.0

This page titled 01-A.2: Open Source Software is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.3: Free Software vs Open Source

Open Source?

Do you use "open source software" or "free software?" Although there are different rules for free software licenses (four freedoms) and open source licenses (Open Source Definition), what is not apparent from those two sets of rules is:

1. Both terms refer to essentially the same set of licenses and software.
2. Each term implies different underlying values.

In other words, although the terms "free software" and "open source software" refer to essentially the same set of licenses, they arrive at that set via different routes. (The results aren't perfectly identical, but the differences are unlikely to matter broadly.) And, even though the licenses are the same, a person's choice of terminology may imply a different emphasis in values.

The concept of "free software" was developed by Richard Stallman in the 1980's. The focus is on what the recipient of software is permitted to do with the software: Roughly, it means that the users have the freedom to run, copy, distribute, study, change, and improve the software.

"Open source" focuses on the practical consequences enabled by these licenses, which is surprisingly effective collaboration on software development. Free software came first. Later, it became apparent that free software was leading to remarkable collaboration dynamics. In 1997, Eric Raymond's seminal essay "The Cathedral and the Bazaar" focused attention on the implications that free software has for software development methodology.

In "Why Open Source Misses the Point of Free Software," Stallman explains that the two terms describe almost the same category of software, but he asserts they stand for views based on fundamentally different values. Stallman's view is that open source is a development methodology while free software is a social movement.

Different values? Yes. But not mutually exclusive. Rather than aligning with one or the other, many people find varying degrees of resonance with the values underlying each term.

Clearing the Confusion

What if someone wants to refer to this type of software without specifying underlying values? Awkwardly, there is no broadly accepted term that refers to the licenses or the software that's neutral about the values implied by each term. In other words, we lack a third term that refers to that same software and same set of licenses, but doesn't take sides with respect to why that software and those licenses are significant. It may be that "open source" was initially expected to be a neutral term; however, it has developed its own implied values.

The closest to a neutral term would be FOSS (free and open source software) or FLOSS (free/libre open source software), which have had limited success fulfilling that value-neutral role. Perhaps the existence of two such terms (with and without "L") may have diluted and thus diminished the ability of either to break out as a broadly used term.

This assortment of terms has contributed to confusion. Would a neutral term be useful? Or is the attempt to separate the associated values a flawed goal? Is a neutral term inappropriate because there are significant free software projects that would not be considered open source? Or the reverse?

This page titled 01-A.3: Free Software vs Open Source is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.4: Different FOSS licensing...

Popular Licenses

There are numerous free/open source software licenses that are all somewhat related, although they each have their own nuances. Below is a list of several of the more commonly used licenses. This list came from the opensource.org website - you can search for each one individually and find all the deep details if you need more information.

The following OSI-approved licenses are popular, widely used, or have strong communities:

- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- Creative Commons Zero
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License version 2.0

This page titled 01-A.4: Different FOSS licensing... is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.5: The GNU Project

GNU is not Unix

The GNU operating system was initiated by Richard Stallman in 1983. The goal was to bring a wholly free software operating system into existence. Stallman wanted computer users to be free to study the source code of the software they use, share software with other people, modify the behavior of software, and publish their own modified versions of the software. This philosophy was later published as the GNU Manifesto in 1985.

It was decided that development would move forward and the resulting system would be compatible with Unix. At the time, Unix was already a popular proprietary operating system. The design of Unix was modular, so it could be re-implemented piece by piece.

Much of the needed software had to be written from scratch. However, existing compatible third-party free software components were also used, including the Mach microkernel that forms the basis of the GNU Mach core of GNU. Most of GNU has been written by volunteers - some in their spare time; some paid by companies, educational institutions, and other non-profit organizations. In 1985, Stallman set up the Free Software Foundation (FSF). In the late 1980s and 1990s, the FSF hired software developers to write the software needed for GNU.

This page titled 01-A.5: The GNU Project is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.6: Unix - What's the Connection?

There Is a Connection Between Unix and Linux

Since the GNU project was, as stated previously, compatible with Unix, it made a certain amount of sense for GNU to follow the Unix philosophy. This philosophy, originated by one of the Unix creators, Ken Thompson, is a set of cultural norms and philosophical approaches to software design. Early Unix developers were instrumental in bringing the concepts of modularity and reusability into software engineering practice, spawning a "software tools" movement. Over time, the leading developers of Unix (and programs that ran on it) established a set of cultural norms for developing software, and these norms became as important and influential as the technology of Unix itself. This has been termed the "Unix philosophy."

In their award-winning Unix paper of 1974, Ritchie and Thompson quote the following design considerations:

- An easy ability to write, test, and run programs.
- Interactive use instead of batch processing.
- Economy and elegance of design due to size constraints ("salvation through suffering").
- Self-supporting system: all Unix software is maintained under Unix.

This page titled 01-A.6: Unix - What's the Connection? is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.7: Finally - Linux

Linux

In 1991, while attending the University of Helsinki, Linus Torvalds became curious about operating systems. Frustrated by the licensing of current operating systems, he began to work on his own system kernel, which eventually became the Linux kernel.

It was advantageous to use the freely available code, so the GNU utilities were incorporated into the fledgling operating system. Code licensed under the GNU GPL can be reused in other computer programs as long as they also are released under the same or a compatible license. Torvalds initiated a switch from his original license, which prohibited commercial redistribution, to the GNU GPL. Developers worked to integrate GNU components with the Linux kernel, making a fully functional and free operating system.

This page titled 01-A.7: Finally - Linux is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.8: Advantages and Disadvantages of Linux

Advantages of Using Linux

Here are some of the major advantages of using Linux:

- The FOSS essence promotes transparency.
- Its design highlights its simplicity and modularity.
- It is extremely customizable.
- It is exceptionally reliable and stable.
- It has a robust integration with the dominant programming languages, like C, C++, Python, Ruby, etc.
- It places a priority on security and privacy.
- It has a passionate community willing to provide support.

Disadvantages of Using Linux

- The main concern is there is no single, official version of Linux.
- It is a bit more difficult to find distributions of Linux where you have vendor support.
- Much of the commercial desktop software does not run on Linux.
- It has gotten better, but there is still a pretty steep learning curve to move to Linux from Windows, or even MacOS.

This page titled 01-A.8: Advantages and Disadvantages of Linux is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.9: Distro List

Linux Distributions

Currently there are over 600 distros, but only about 500 of them are currently active. Here are just a few distros and their positive points

Linux distribution	Description
Mint	An Ubuntu distro with a new set of applications and an updated desktop theme. Since its beginnings, there have been a variety of graphical tools for enhanced usability.
Ubuntu	One of the most popular distros, it is based on Debian. Has both a desktop distro and a server distro.
Slackware	The oldest distro. Considered highly stable, clean and largely bug-free, with a strong adherence to UNIX principles.
Arch Linux	Excellent software management infrastructure. Unparalleled customization and tweaking options. Superb on-line documentation.
CentOS	Extremely well-tested, stable and reliable. Free to download and use. Comes with 7+ years of free security updates.
Kali Linux	A collection of security and forensics tools. It features timely security updates, support for the ARM architecture, a choice of four popular desktop environments, and seamless upgrades to newer versions.
Fedora	Highly innovative, with outstanding security features, a large number of supported packages, and strict adherence to the free software philosophy. Availability of live spins featuring many popular desktop environments.
Debian	Very stable, and remarkable quality control. Includes over 30,000 software packages and supports more processor architectures than any other Linux distribution.
openSUSE	Comprehensive and intuitive configuration tool, large repository of software packages, excellent web site infrastructure and printed documentation, Btrfs with boot environments by default.
Gentoo	Highly flexible and endlessly customizable. Able to use a range of compile-time configurations, init systems, and run on many architectures.

On the DistroWatch web site they track the distros who get the most hits - interesting collection of data here.

Last 12 months			Last 6 months			Last 3 months			Last 1 month		
1	MX Linux	4740▲	1	MX Linux	4585▼	1	MX Linux	4652▼	1	MX Linux	4192▲
2	Manjaro	2786▼	2	Manjaro	2934▲	2	Manjaro	3154–	2	Manjaro	3023▲
3	Mint	2275▲	3	Mint	2501▲	3	Mint	2592▲	3	Mint	2738▲
4	Debian	1599▲	4	Debian	1672–	4	Ubuntu	1725▲	4	Ubuntu	1816▲
5	Ubuntu	1517▲	5	Ubuntu	1635▲	5	Debian	1687▲	5	Debian	1629–
6	elementary	1395–	6	elementary	1472▲	6	elementary	1543▲	6	elementary	1480▲
7	Solus	1172–	7	Solus	1270▲	7	Solus	1396▲	7	Zorin	1398▲
8	Fedora	1021▼	8	Zorin	1093▲	8	Zorin	1116▲	8	Solus	1286▲
9	Zorin	961▲	9	Fedora	1041▲	9	Fedora	1057–	9	Fedora	1132▲
10	deepin	897▲	10	deepin	952–	10	KDE neon	959▲	10	Ubuntu Kylin	1037▲
11	antiX	790▲	11	KDE neon	878▲	11	deepin	917▼	11	KDE neon	1008▲
12	KDE neon	765▲	12	antiX	862▲	12	antiX	857▲	12	antiX	966▲
13	openSUSE	744▼	13	Pop!_OS	770▲	13	Pop!_OS	803▲	13	deepin	906▲
14	CentOS	703–	14	ArcoLinux	737▲	14	Arch	785▲	14	Pop!_OS	862▲
15	ArcoLinux	673▲	15	PCLinuxOS	724–	15	CentOS	784–	15	Arch	710▲
16	PCLinuxOS	667–	16	CentOS	721▼	16	PCLinuxOS	728▲	16	PCLinuxOS	686▼
17	Pop!_OS	665▲	17	Arch	702–	17	Ubuntu Kylin	700▲	17	ArcoLinux	679▲
18	Arch	652▲	18	openSUSE	696–	18	ArcoLinux	694–	18	openSUSE	675▼
19	Kali	515–	19	Kali	583–	19	openSUSE	686–	19	CentOS	643–
20	ReactOS	450▼	20	Puppy	478▲	20	Kali	619–	20	Bodhi	554▲

Figure **1.9.1**: This shows a portion of DistroWatch's home page listing showing the top 275 most popular Linux distros over the last 12 months, the last 6 months, over the last 3 months and over the last 1 month. (Copyright (C) 2001 - 2020 Atea Ataroa Limited. All rights reserved)

This table shows a great deal of information, which is constantly changing. Up to date information can be found at the above referenced DistroWatch web site. It is interesting to notice that MX Linux is the most popular across all four of the time frames, followed by Manjaro distribution. The DistroWatch Page Hit Ranking statistics are a light-hearted way of measuring the popularity of Linux distributions and other free operating systems among the visitors of this website. They correlate neither to usage nor to quality and should not be used to measure the market share of distributions. They simply show the number of times a distribution page on DistroWatch was accessed each day, nothing more.

This page titled 01-A.9: Distro List is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-A.10: How is Linux Used?

One of the cool things about Linux is that we can change it in whatever manner we desire. As such, we can alter it to run on a variety of platforms. Here is a list of some of the devices Linux runs:

https://eng.libretexts.org/@go/page/26786

Where is Linux Being Used	Description
NASA	You might not be aware that Linux powers NASA. NASA's Pleiades supercomputer runs Linux. The International Space Station switched from Windows to Linux six years ago due to the operating system's reliability. NASA even recently deployed three "Astrobee" robots—which run Linux—to the International Space Station.
eReaders	If you use any of Amazon's services—from Amazon Elastic Compute Cloud (Amazon EC2) to Fire TV—you are running on Linux. When you ask Alexa what time it is, or for the score of your favorite sports team, you are also using Linux, since Alexa is powered by Fire OS (an Android-based operating system). In fact, Android was developed by Google as Linux for mobile handsets, and powers 76% of today's mobile phones.
Television	TiVo, Roku and Chromecast all run Linux. LG uses webOS, which is based on the Linux kernel. Panasonic uses Firefox OS, which is also based on the Linux kernel. Samsung, Phillips, and many more use Linux-based operating systems to power their devices.
Smartwatches & laptops	If you own a smartwatch, it's likely running Linux. School systems outfit their students with a Chromebook - these lightweight laptops use Chrome OS, which is based on Linux.
Automobiles	Automotive-Grade Linux has enlisted manufacturers like Toyota, Mazda, Mercedes-Benz, and Volkswagen in a project that will see Linux as the standard code base for automobiles. It is also likely that your in-vehicle infotainment system runs Linux. The GENIVI Alliance develops "standard approaches for integrating operating systems and middleware present in the centralized and connected vehicle cockpit," according to its website.
Social media	As you're scrolling and commenting, you may realize what a lot of work these platforms are doing. Perhaps then it's not so surprising that Instagram, Facebook, YouTube, and Twitter all run on Linux.
Business and government	The New York Stock Exchange runs on Linux, as does the Pentagon. The Federal Aviation Administration handles over sixteen million flights a year, and they operate on Linux. The Library of Congress, House of Representatives, Senate, and White House all use Linux.
Retail	That entertainment system in the seat back on your latest flight is likely running on Linux. It's possible that the point of sale at your favorite store is running Linux. Tizen OS, which is based on Linux, powers an array of smart home and other smart devices. Many public libraries now host their integrated library systems on Evergreen and Koha. Both of those systems run on Linux.
Apple	If you are an iOS user who uses iCloud, then you, too, are using a system that runs on Linux. Apple Computer's company website runs on Linux. If you would like to know what other websites are run on Linux, be sure to use Netcraft and check the results of "What's that site running?"

Adapted from:

"11 surprising ways you use Linux every day" by Don Watkins, OpenSource.com is licensed under CC BY-SA 4.0

This page titled 01-A.10: How is Linux Used? is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.11: How the Shell and CLI Work

GUI vs CLI

"Under Linux there are GUIs (graphical user interfaces), where you can point and click and drag, and hopefully get work done without first reading lots of documentation. The traditional Unix environment is a CLI (command line interface), where you type commands to tell the computer what to do. That is faster and more powerful, but requires finding out what the commands are." -- from man intro(1).

There are also many graphical user interfaces (GUIs), but each of them works differently and there is little standardization between them. Experienced users who work with many different Linux distributions therefore find it easier to learn commands that can be used in all varieties of Ubuntu and, indeed, in other Linux distributions as well. However, it is important to note that even experienced users often cut and paste commands (from a guide or manual) into the terminal; they do not memorize them.

It is important, of course, to know how to use the command line interface (CLI) in the Linux terminal - and anyone who can manage typing, backspacing, and cutting and pasting will be able to use the terminal (it is not any more difficult than that). Many Linux distros do not even include the GUI with their server package, so it is imperative that you are comfortable with and efficient at using the command line interface.

"UsingTheTerminal" by Chris Perry, Ubuntu Community Help Wiki is licensed under CC BY-SA 3.0

This page titled 01-B.11: How the Shell and CLI Work is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.12: CLI - The Good and the Bad

The Good of Using the Command Line

CLI preceded GUI as a computer interface. Since GUIs are common now, why should we still care about CLI? Some advantages of using the command line are:

- It can save you time.
- It can help when you are unable to use the GUI, such as a system crash or a configuration issue.
- It can enable you to use Linux in ways that using a GUI exclusively can not (such as scripting repetitive tasks).

For example, let's say you have been told by the systems administrator that you have used too much space. You want to quickly work out where the most space is used, so using a graphical interface, start your timer - go. Now, go to a command line and type: `du | sort -n` (we will describe more later). See? It is faster to do some things on the command line (and other times, easier for graphical).

The following video is a brief introduction to the command line interface. The various components of the interface are introduced, along with some tips on using the interface. More in depth information about using the CLI is in following sections of this course.



The Bad of Using the Command Line

But...there is also the "bad" to using the CLI:

- It can be more difficult to learn than a GUI since we can't really "explore" like we can with a GUI.
- Linux commands have many options which are not always consistent.
- There are many command-line environments among the Linux, Unix, macOS, and Windows platforms and they are all different.

This page titled 01-B.12: CLI - The Good and the Bad is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.13: Shells: What and Why

The Shell

A **Linux shell** is a command-line interpreter or shell that provides a command line user interface for Linux operating systems. The shell is both an interactive command language and a scripting language, which can be used by the operating system to control the execution of the system using shell scripts.

Users typically interact with a Linux shell using a terminal emulator; however, direct operation via serial hardware connections or Secure Shell are common for server systems. All Unix shells provide filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.

The shell sits between the Linux applications, including the Linux terminal app/keyboard, and the kernel (think of operating system). So when a command is entered in a terminal window, the shell interprets what is being requested and hands the request off to the kernel, which then interfaces with the hardware if necessary. If the commands are in a script file (we will get to this later), the shell reads each line of the script, executing the commands and presenting the output as necessary

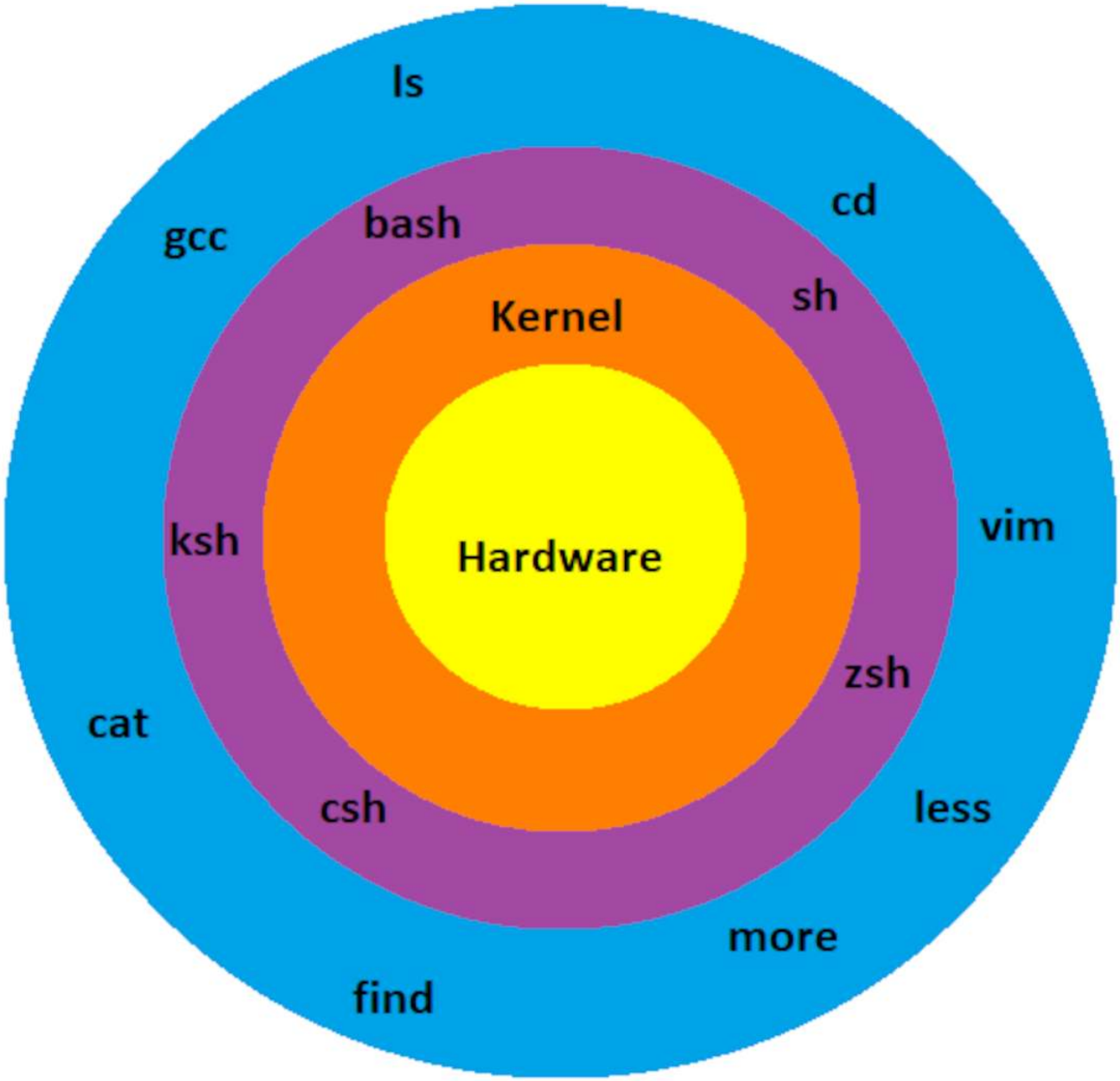


Figure **1.13.1**:Linux Shell. ("Linux Shell" by Patrick McClanahan is licensed under CC BY-SA 4.0)

There is actually a great deal going on in this image. First of all, the hardware is at the center, and surrounded by the Linux Kernel. The shell must interact with the kernel in order to interface with the hardware. The purple ring represents the various shells that are available to the Linux user: bash, k shell, c shell, z shell and the original shell. The shells all interface with the kernel. The blue ring shows only some of the commands available to a user. Linux commands interface with the shell, which passes the requests to the kernel which then performs the necessary operations, including any interactions with the hardware if required.

There are numerous shells available; below we list several of them to give you an idea of the diversity.

Shell	Details
Bourne Shell - denoted as sh	Written by Steve Bourne at AT&T Bell Labs. It is the original UNIX shell. It is faster and more preferred, but lacks features for interactive use like the ability to recall previous commands.
C shell - denoted as csh	Created by Bill Joy at the University of California at Berkeley. It incorporated features such as aliases and command history and includes helpful programming features like built-in arithmetic and C-like expression syntax.
Korn shell - denoted as ksh	Written by David Korn at AT&T Bell Labs. It is a superset of the Bourne shell and supports everything in the Bourne shell. It has interactive features. It includes features like built-in arithmetic and C-like arrays, functions, and string-manipulation facilities. It is faster than C shell and compatible with script written for C shell.
Bash shell - denoted as bash	The Bourne Again Shell was originally released in 1989. Bash has grown to easily become the most used shell across the Linux world, and it is commonly found in other unix-like operating systems as well.
Tcsh - denoted as tcsh	A derivative of csh, the Berkely Unix C shell, which sports a very long lineage back to the early days of Unix and computing itself.
zsh	zsh originated in the early 90's and sports a number of useful features, including spelling correction, theming, namable directory shortcuts, sharing your command history across multiple terminals, and various other slight tweaks from the original Bourne shell.

This page titled 01-B.13: Shells: What and Why is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.14: Doing bash

Bash? CLI?

It used to be that there was no graphical user interface (GUI) - **everything** was done via the command line interface (CLI).

The CLI is a non-graphical, text-based interface to the computer system, where the user types in a command and the computer then successfully executes it. The Terminal is the platform or the IDE that provides the command line interface (CLI) environment to the user. Since most users work on Linux systems that run a GUI, the app that is often used is a terminal emulator. However, some Linux server distros do not have a GUI loaded, and these systems truly run in a terminal environment.

Linux uses the Terminal app as the default terminal emulator. However, there is also the XTerm app, KDE Konsole, xterm, and gnome-terminal to name a few of the other more popular terminal emulator apps.

```
pbmac@pbmac-server $
```

In looking at this particular example we see the user's command line prompt. The components of this command prompt are:

pbmac - The username of the current user

pbmac-server - The hostname of the server

$ - The prompt symbol. The user's keyboard input appears following this symbol.

The user can customize their command prompt using config files, which we will talk about in a later portion of this chapter. Some of the common customizations are adding the display of the current working directory; showing only the user name; showing the fully qualified domain name...you can decide for yourself what you would like it to be.

The CLI terminal accepts whatever is typed on the keyboard and passes it to the currently running shell. The shell then receives and interprets what the user has typed and attempts to determine if the text is within the syntax of the shell's "language." If there is some error in what the user entered, the shell sends an error message back to the terminal. If there is no error in the user's input, the shell passes the command/arguments/options to Linux to execute the command. The shell then displays the results it receives back from the operating system (OS).

This page titled 01-B.14: Doing bash is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.15: Bash Command How To

Enter Commands in Bash

Just like a programming language the bash shell has a syntax — that is the manner in which a command is entered on the keyboard. If commands, options and arguments are not entered in the proper manner the user will receive an error. Most commands have a variety of options that may or may not be optional. Take the ls command that is shown in the example on the left. This command will allow the user to simply enter the command without any options or arguments; it simply assumes the user is asking for a short list of the current directory. (a short list is simply the file/directory names). If you enter options, it still assumes those options pertain to the listing of the current directory. Only if the user enters one or more arguments does it list something other then the current directory. So, how do you know which commands require options, or arguments? It simply takes practice and actually working on a Linux system.

If you fail to enter a command in the proper syntax, the shell will return an error message of some type. These messages are not always as easy to decipher as we would like, but often they are pretty straight forward. The most common one is probably "command not found", which is the result of a typo on the command line. Another common error is "no such file or directory", again, often caused by a typo, or perhaps an incorrect path name.

In the following example, the user enters a command, followed by an option, and then the argument(s). Options are **always** proceeded with a dash in Linux. You can string the options together using just a single dash. Be aware, that you do not need to use options, or arguments on many commands, but on other commands arguments are required. In this command we see the user enters the ls command, which provides a list of the files in the specified directory. The user also provides 2 options: 1) the a option says to show **all** the files, even the hidden ones; and 2) the l option which provides a long listing, giving all the file details for each file. The last part of the command is the argument, in this case the user enters /usr, specifying the system should list the files in the /usr directory. Again — time and experience will help you with all of this.

```
pbmac@pbmac-server $ ls –al /usr
total 196
drwxr-xr-x  12 root root   274 Dec  7  2020 .
drwxrwxrwx  21 root root   550 May 24 17:45 ..
dr-xr-xr-x   2 root root 59808 Dec 26  2020 bin
drwxr-xr-x   2 root root    37 Aug 12  2020 games
drwxr-xr-x  98 root root  8023 Dec 19  2020 include
dr-xr-xr-x  40 root root  1214 Dec 19  2020 lib
dr-xr-xr-x 106 root root 70319 Dec 19  2020 lib64
drwxr-xr-x  40 root root  2771 Dec 19  2020 libexec
drwxr-xr-x  12 root root   252 Dec  7  2020 local
dr-xr-xr-x   2 root root 16054 Dec 26  2020 sbin
drwxr-xr-x 197 root root  4832 Dec 31  1969 share
drwxr-xr-x   4 root root    83 Dec  7  2020 src
lrwxrwxrwx   1 root root    10 Aug 12  2020 tmp -> ../var/tmp
```

Figure **1.15.1**: Command line: options and arguments ("Linux Shell" by Patrick McClanahan is licensed under CC BY—SA 4.0)

This table provides some examples of commands, commands with options, commands with arguments, and commands with options and arguments. Any of these combinations are valid.

Command line entered	Explanation	Does what?
`ls`	Command only — no options, no arguments.	Returns a list of the files/directory names from the current directory.
`ls –al`	Command with 2 options. Linux allows you to enter: ls —a —l, or combine the options in one: ls —al.	The —a shows the hidden files — those files that begin with a period — in the example you use a file called . and one called .. The —l is a "long" listing that lists the details of each file/directory, as shown in the example above.
`ls /usr`	Command with an argument but no options. Linux knows this is an argument because it is NOT preceded by a dash.	The output is the default listing of the /usr directory — which is simply a listing of the files/directories in the /usr directory.
`ls –al /usr`	Command with both options and arguments.	This produces a long listing of all files/directories — including hidden files/directories — in the /usr directory.

This page titled 01-B.15: Bash Command How To is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.16: Let's Look at the Basics

Some Basic Commands

There are some basic commands that are useful when one is new to the Linux realm. Some of these commands will be covered in greater detail as we move through this journey. We discuss some of the basics here.

https://eng.libretexts.org/@go/page/26792

Command Name	Command Description	Examples
~ (tilde)	This symbol stands for your home directory.	If you are pbmac, then the tilde (~) stands for /home/*pbmac*
pwd	The **pwd** command will allow you to know in which directory you're located (**pwd** stands for "print working directory").	In some instances **"pwd"** in the Desktop directory will show "~/Desktop". The GNOME Terminal also displays this information in the title bar at the top of its window.
ls	The **ls** command will show you ('list') the files in your current directory. Used with certain options, you can see sizes of files, when files were made, and permissions of files.	"**ls ~**" will show you the files that are in your home directory. "**ls /var/log**" will show you the files in the /var/log directory
cd	The **cd** command will allow you to change directories. When you open a terminal you will be in your home directory. To move around the file system you will use **cd**.	• To navigate into the root directory, use "**cd /**" • To navigate to your home directory, use "**cd**" or "**cd ~**" • To navigate up one directory level, use "**cd ..**" • To navigate to the previous directory (or back), use "**cd -**" • To navigate through multiple levels of directory specify the full directory path that you want to go to. "**cd /var/www**" will take you to the /www subdirectory of /var/. "**cd ~/Desktop**" will move you to the Desktop subdirectory inside your home directory.
cp	The **cp** command will make a copy of a file.	"**cp bar foo**" will make an exact copy of "bar" and name it "foo", but the file "bar" will still be there. If you are copying a directory, you must use "c**p -r directory foo**" (copy recursively). ("recursively" means to copy the directory and all its files and subdirectories and all their files and subdirectories of the subdirectories and all their files, and on and on, "recursively")
mv	The **mv** command will move a file to a different location or will rename a file - the original file will be removed.	"**mv bar foo**" will rename the file "bar" to "foo." "**mv foo ~/Desktop**" will move the file "foo" to your Desktop directory, but it will not rename it. You

Command Name	Command Description	Examples
		must specify a new file name to rename a file.
rm	Use this command to remove or delete a file in your directory.	"**rm foo**" will delete the file "foo"
rmdir	The **rmdir** command will delete empty directories.	To delete a directory and all of its contents recursively, "**rm -r directoryname**"
mkdir	The **mkdir** command will allow you to create directories.	"**mkdir music**" will create a directory called "music" in the current directory.

This is by no means an exhaustive list of basic commands, but this will get you started so you can begin to navigate around the system a bit.

"UsingTheTerminal" by Chris Perry, Ubuntu Community Help Wiki is licensed under CC BY-SA 3.0

This page titled 01-B.16: Let's Look at the Basics is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.17: File Basics: More Later

Linux and Files

We will spend a great deal of time in later chapters looking at and working with commands to view the contents of text files. We must learn how to manage text files since there are many Linux configuration files that are text files. There are two types of commands that we will use when working with text files: 1) those commands that simply display the contents of the file and 2) those commands which let us edit the files in order to make necessary changes.

File Viewing Commands

There are a few commands that simply display the contents of a text file to the screen. The most basic of these commands is the **`cat`** command. It takes a single argument, which is a file name. So, the command **`cat file1`** will print the contents of the file to the screen.The problem with the **`cat`** command is that if the contents of the file are more than a single screen full, the contents of the file simply scrolls past, without any way to view the contents.

To resolve this issue, Linux provides the **`less`** command (and the **`more`** command - but we will talk more about that later). The **`less`** command displays a screen full of the file contents at a time. You can then control the content being displayed. You have access to the PageUp and PageDown keys that will display the previous screen full, or the next screen full. The space bar displays the next line in the file, scrolling everything else up one line on the screen.

File Editing Commands

We will discuss the file editing capabilities of Linux as we progress, and you will have many opportunities to work with the editor of your choice. In the terminal mode there are only a very few editors to choose from. The two most popular are **`vi`** and **`nano`**. The current iteration of vi in Linux is actually **`vim`** (**`vi`** improved). The problem with **`vi`** is that it has been around a long time. The good thing about **`vi`** is that it has been around a long time. Having been invented in the Unix days, it is a rather archaic tool, although it is quite powerful if you take the time to learn how to use it. There have been a lot of changes that have upgraded the capabilities, and there are several excellent web sites that provide help if learning to use **`vi`**. The other editor, **`nano`**, is a very simple to use tool, but most certainly can handle basic text file operations. It does have a few quirks of its own, but overall is simple and usable.

There is also the graphical tool **`gedit`**, a simple graphical interface that makes using it a simple job.

This page titled 01-B.17: File Basics: More Later is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.18: Who is superuser?

Superuser

In Linux there is a term - superuser. This refers to any user that has administrator privileges. The user of the administrative user is "root." Anyone with root access has the ability to do whatever they want, which is why we need to be careful as to who gets root access on our systems. Historically, you could log in to a Linux system as root by providing the appropriate password. Most distros do not allow root login anymore due to the security concerns associated with such access.

So, the question is, how does one gain access to administrative privileges? There are a couple of methods we can use. Using the **`su`** command a user can switch to any user of the system. The **`su`** command stands for "switch user," "substitute user," and even "super user"...it depends on where you look. Issuing the command **`su - pbmac`** will ask you for pbmac's password. If you can provide the proper password, then your userid will be set to pbmac. If you do not provide a userid as in **`su -`**, or if you provide the userid of root - **`su - root`**, then you must provide the root password to become "superuser," or the root user.

Be careful about changing to the root user as you have total authority and privileges on the system. As root it is possible to mess up the system really quickly to the point that your system is unusable, especially by inadvertently removing files or directories. So, be very careful about using su to become root for extended work.

A better solution is to use the **`sudo`** command. Sudo allows a permitted user to execute a command as the superuser, or even as another user. We will talk more about the **`sudo`** command in a future lesson.

This page titled 01-B.18: Who is superuser? is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-B.19: Leftovers, What Else Do We Need?

Before We Move On

Just a few more points to cover before we move on.

Shutdown/Reboot

Those who come from the world of Windows systems are used to having to reboot the system at various times. While Linux does not suffer from the frequency of reboots that Windows seems to, there are still times when Linux need to be powered off, or simply rebooted. Current distros are better at dealing with being shut down by a user hitting the power off button on their computer, BUT, that is never the recommended method. It is best to perform an orderly shutdown, and for that there is the **`shutdown`** command and a **`reboot`** command.

A couple of the more common invocations of shutdown:

- **shutdown -h**
  - The -h option tells shutdown to halt the system, that is, to power it off after an orderly shutdown.
- **`shutdown -h -t 60`**
  - The system will shut down after the specified time delay - in this example the delay is 60 seconds
- **shutdown -r**
  - If a reboot is desired, the -r option is available. You can also issue the **`reboot`** command from the terminal to achieve the same resultsBash History

The bash shell comes with a very powerful command called “history.” This command is used to extract the history information about commands that have been executed by the user including all previous login sessions. Linux stores the command history for each user in a file ~/.bash_history by default.

As each command is issued, it is recorded in the history file. The **`history`** command reads the history file and displays the contents to the terminal screen. You can see an example of the output in the image to the left. This history is set up to record 2000 entries, so you will be able to view the previous 2000 commands you enter, and the history file is persistent between login sessions.

```
pbmac@pbmac-server $ history
 2000  ./"HW-01"
 2001  journalctl
 2002  ls /var/log/journal/
 2003  ls -l /var/log/journal/
 2004  ls -l /var/log/journal/75d57d2bcd2831ebf58bcc1a000008d4/
 2005  journalctl -x
 2006  journalctl
 2007  journalctl -x
 2008  last
 2009  lastlog
 2010  man dar
 2011  cd /usr/share/man
 2012  ls
 2013  cd man8
 2014  cd ../man1
 2015  ls tar*
```

To access previous commands you can use the UpArrow and DownArrow to find the command you wish to re-issue. Often it is a matter of a typo - use the arrow keys to find the command, then use the left/right arrow keys to move your cursor and use the

backspace or delete keys to remove characters and type in the correct input.

There is much more that can be done with history in regards to command recall and command editing. There are several web sites and blogs that discuss this topic in great detail. Search for "bash history" and find one that works for you.

Shell Tricks

TAB completion saves typing. The *completion* feature in Bash permits typing a partial command, then pressing the **[Tab]** key to auto-complete the command sequence. If multiple completions are possible, then **[Tab]** lists them all. Let's see how it works.

```
bash$ xtra[Tab]
xtraceroute       xtrapin           xtrapproto
xtraceroute.real  xtrapinfo         xtrapreset
xtrapchar         xtrapout          xtrapstats

bash$ xtrac[Tab]
xtraceroute       xtraceroute.real

bash$ xtraceroute.r[Tab]
xtraceroute.real
```

There is a `clear` command, which clears your terminal window. Sometimes things get cluttered, and to ease reading the terminal you can use the clear command to clean up your workspace.

Use your history - see the previous section.

Adapted from:
"Appendix J. An Introduction to Programmable Completion" by David Merrill is licensed under CC BY-SA 3.0

This page titled 01-B.19: Leftovers, What Else Do We Need? is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

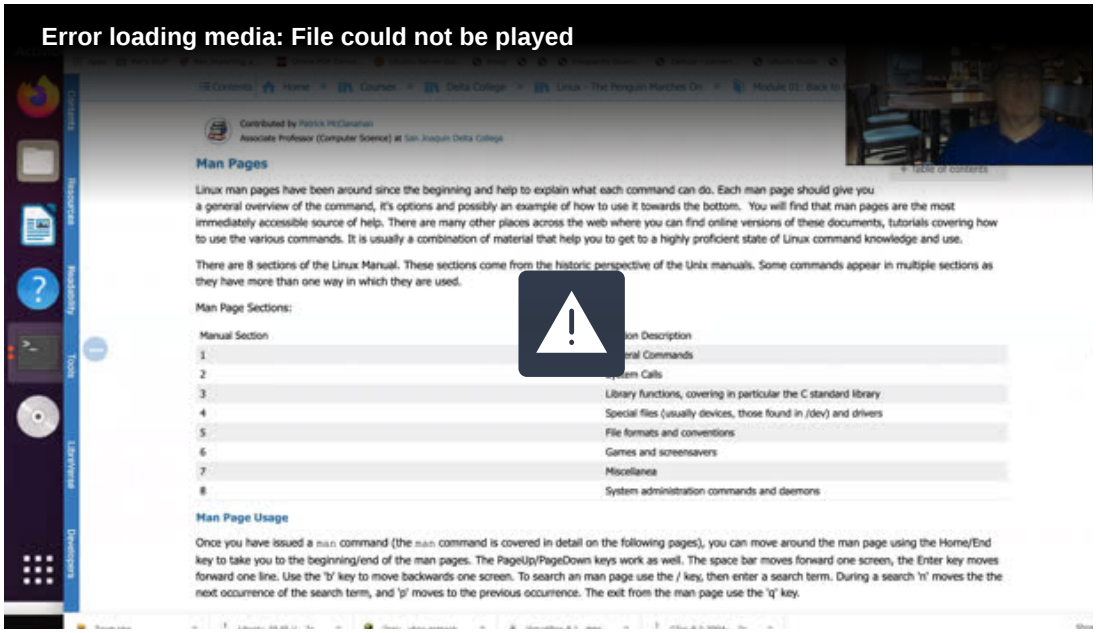
01-C.20: Help! Where can I find help in Linux?

Where is Help?

As always, documentation is a necessity with any complex project. Documentation is meant to help Linux users, no matter their responsibility or experience level, to perform a variety of activities and resolve the diversity of issues that will arise. There is not a single, authoritative source of Linux system documentation. Documentation can be found across multiple sources that appear in multiple forms, each one suited to a particular context.

- Linux manual pages
- Help commands that are part of the software
- Online Linux documentation projects
- Question and answer websites
- Online forums and social media
- Books and other print resources
- Usenet newsgroups
- Internet mailing lists

This video discusses the use of Linux manual pages, shortened to man pages, which are a great source of assistance in addition to the fact that they are readily available on most Linux systems. It is up the whoever installs the operating system whether they install the man pages.



This page titled 01-C.20: Help! Where can I find help in Linux? is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-C.21: Linux man pages

Man Pages

Linux man pages have been around since the beginning and help to explain what each command can do. Each man page should give you a general overview of the command, it's options, and possibly an example of how to use it towards the bottom. You will find that man pages are the most immediately accessible source of help. There are many other places across the web where you can find online versions of these documents, and tutorials covering how to use the various commands. It is usually a combination of material that helps you to get to a highly proficient state of Linux command knowledge and use.

There are 8 sections of the Linux Manual. These sections come from the historic perspective of the Unix manuals. Some commands appear in multiple sections as they have more than one way in which they are used.

Man Page Sections are as follows:

1. General Commands
2. System Calls
3. Library functions, covering in particular the C standard library
4. Special files (usually devices, those found in /dev) and drivers
5. File formats and conventions
6. Games and screensavers
7. Miscellanea
8. System administration commands and daemons

Man Page Usage

Once you have issued a **`man`** command (the **`man`** command is covered in detail on the following pages), you can move around the man page using the Home/End key to take you to the beginning/end of the man pages. The PageUp/PageDown keys work as well. The space bar moves forward one screen, the Enter key moves forward one line. Use the 'b' key to move backwards one screen. To search any man page use the / key, then enter a search term. During a search 'n' moves to the next occurrence of the search term, and 'p' moves to the previous occurrence. To exit from the man page use the 'q' key.

Adapted from:
"man command in Linux with Examples" by msdeep14, Geeks for Geeks is licensed under CC BY-SA 4.0

This page titled 01-C.21: Linux man pages is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-C.22: Man Page Format

Man Pages and The Man Command

Man pages are all formatted the exact same way. Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES, VERSIONS, CONFORMING TO, NOTES, BUGS, EXAMPLE, AUTHORS, and SEE ALSO. However not every man page contains every one of these sections.

To access a man page a user enters the man command. The **man** command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command. To display the man page for the **man** command we would enter:

```
# The first man is the command, the second is the specific page we want to view
pbmac@pbmac-server $ man man

# Optionally we can specify the section number - there are 8 sections as described on
pbmac@pbmac-server $ man -s 1 man
```

The man page shown below is broken up in order to be able to discuss the sections - normally the output is all together.

At the top of every man page, in the left and right corners, is the name of the command (in this case the man command). There is the **NAME** section, followed by the **SYNOPSIS** section which lists all of the options and arguments.

```
MAN(1)                                          Manual pager utils

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m s
       list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--n
       prompt]  [-7]  [-E  encoding]  [--no-hyphenation]  [--no-justification]  [-p  s
       [-H[browser]] [-X[dpi]] [-Z] [[section] page[.section] ...] ...
       man -k [apropos options] regexp ...
       man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
       man -f [whatis options] page ...
       man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-
       [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file
       man -w|-W [-C file] [-d] [-D] page ...
       man -c [-C file] [-d] [-D] page ...
       man [-?V]
```

This is immediately followed by a detailed **DESCRIPTION** of the command, its purpose, and other information about the command.

```
DESCRIPTION
       man  is  the  system's  manual pager.  Each page argument given to man is norma
       utility or function.  The manual page associated with each of these arguments
```

```
       played.  A section, if provided, will direct man to look only in that section o
       action is to search in all of the available sections following a pre-defined or
       3pm  3perl 3am 5 4 9 6 7" by default, unless overridden by the SECTION directiv
       and to show only the first page found, even if page exists in several sections
```

There are often **EXAMPLES** of how to use the command. Not every single option is shown, but sufficient examples to help a user do what is being attempted, or whether to turn to some other resource for further assistance.

```
EXAMPLES
       man ls
           Display the manual page for the item (program) ls.

       man man.7
           Display the manual page for macro package man from section 7.

       man -a intro
           Display,  in  succession, all of the available intro manual pages contained
           possible to quit between successive displays or skip any of them.
```

The **OPTIONS** section does include each one of the options for a given command. Sometimes, even that doesn't provide the necessary clarity as to how or what a certain option does. Thank goodness for the plethora of web resources.

```
OPTIONS
       Non  argument  options that are duplicated either on the command line, in $MANO
       ful.  For options that require an argument, each duplication will override the

   General options
       -C file, --config-file=file
              Use this user configuration file rather than the default of ~/.manpath.

       -d, --debug
              Print debugging information.
```

Right near the end of the man page is the **SEE ALSO** section. This often provides valuable information about other commands that might be similar and might even be better to accomplish the user's goal. If you can't seem to figure out how to make a command behave the way you want, try looking at the SEE ALSO section and look up some of these commands as well.

```
SEE ALSO
       apropos(1), groff(1), less(1), manpath(1), nroff(1), troff(1), whatis(1), zsoel
       path(5), ascii(7), latin1(7), man(7), catman(8), mandb(8), the man-db package r
```

Adapted from:
"man command in Linux with Examples" by AshwinGoel, Geeks for Geeks is licensed under CC BY-SA 4.0

This page titled 01-C.22: Man Page Format is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-C.23: Additional Linux System Help Choices

Other Helping Commands

Linux has commands that provide additional choices for a user to find assistance with finding and using commands from the CLI.

https://eng.libretexts.org/@go/page/26799

Choice	Description	Example
info	In addition to the man pages, you can read the Info pages about a command, using the **info** command. These usually contain more recent information and are somewhat easier to use. The man pages for some commands refer to the Info pages. The info command at times refers to the man pages.	**info** - prints default page **info info** - opens info page about how to use info **info man** - depends on the distro - but usually opens the same page as **man man** command.
apropos	This command searches the NAME section of the man pages for the keyword the user enters. The output is the one line in the name section, but it also includes which section of the man pages is the command found in.	pbmac@pbmac-server $ apropos email gpgparsemail (1) - Parse a mail message into an annotated format mailto.conf (5) - configuration file for cups email notifier nautilus-sendto (1) - convenience application to send a file via email uuencode (1) - encode a file into email friendly text X509_check_email (3ssl) - X.509 certificate matching X509_VERIFY_PARAM_set1_email (3ssl) - X509 verification parameters xdg-email (1) - command line tool for sending mail using the...
whatis	This command behaves similar to apropos, but it gives you the command, and what it does. You would use this command if you know a command name, but you are wanting to figure out what the command does.	pbmac@pbmac-server $ whatis ls ls (1) - list directory contents pbmac@pbmac-server $ whatis cat cat (1) - concatenate files and print on the standard output
--help	Many of the Linux commands have been updated to include a **--help option**. The user simply enters the command in question followed by this option. Often the output is quite verbose.	A partial output example: pbmac@pbmac-server $ cat --help Usage: cat [OPTION]... [FILE]... Concatenate FILE(s) to standard output. **With no FILE, or when FILE is -, read standard input.**

Choice	Description	Example

		`-A, --show-all equivalent to -vET`
		`-b, --number-nonblank number nonempty output lines, overrides -n`
		`-e equivalent to -vE`

`/usr/share/doc/`	The files and subdirectories contain a lot of additional documentation for libraries, system libraries and other software packages installed on the system. There is not really a utility to read this documentation...using CLI commands like **`cat`** or **`less`** is how to do it.	Some of the subdirectories contain README files, or other text files along with various gzipped files.

This page titled 01-C.23: Additional Linux System Help Choices is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-C.24: Sources of Online Linux Documentation

Online Help Sources

With the advent of the web, we now have a plethora of additional places we can find help with almost any topic, including using and administering Linux. There are several sites that provide better information than others; it pretty much comes down to what sort of information you are looking for. For Linux documentation there are several sources that are the 'go to' sites.

Organization	Description	Link
The Linux Documentation Project	**TLDP** is a loosely knit team of volunteers who provide documentation for many aspects of Linux. There are several forms of documentation: Guides, HOWTOs, man pages, and FAQs. A guide is typically a longer book with broader coverage of a subject; for instance, the Network Administration or User Guide. The intent is to understand the whole subject, as opposed to performing only one task. If you want to have a broader look at some aspect of Linux, then the guides should be very handy. A HOWTO is usually a step-by-step guide that describes, in detail, how to perform a specific task. For example, you can use the Linux Installation HOWTO to help you install Linux on a system, but it does not cover how to set up a Web server so that you can focus on a particular task.	The TLPD web site
The Linux man-pages project	The Linux man-pages project documents the Linux kernel and C library interfaces that are employed by user-space programs. With respect to the C library, the primary focus is the GNU, C library (glibc), although, where known, documentation of variations on other C libraries available for Linux is also included.	man-pages project web site
Distro documentation	Most of the distros maintain a documentation page on their web site that is specific to the distro. This documentation is usually linked on a distros home page. Another option would be: 1) go to the Distro Watch website 2) select the desired distro 3) scroll down - on the left side is a listing for the distros documentation page.	Online docs for Ubuntu Distro Watch page for Ubuntu, scroll down and click on the Documentation link found on the left side
GNU Core Utilities (GNU coreutils)	The GNU Core Utilities are the basic file, shell and text manipulation utilities of the GNU operating system. These are the core utilities which are expected to exist on every operating system.	Core Utilities page includes the link to the documentation

This page titled 01-C.24: Sources of Online Linux Documentation is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

01-C.25: Asynchronous Linux Help Resources

https://eng.libretexts.org/@go/page/26801

Newsgroups and Forums

Source Name	Description	Where is it?
Usenet Newsgroups	**Usenet** is a worldwide distributed discussion system available on computers. It was developed from the general-purpose Unix-to-Unix Copy (UUCP) dial-up network architecture. Tom Truscott and Jim Ellis conceived the idea in 1979, and it was established in 1980. Users read and post messages (called *articles* or *posts*, and collectively termed *news*) to one or more categories, known as newsgroups. Usenet resembles a bulletin board system (BBS) in many respects and is the precursor to Internet forums that are widely used today. Discussions are threaded, as with web forums and BBSs, though posts are stored on the server sequentially. The name comes from the term "users network." "Usenet" Wikimedia Foundation is licensed under CC BY-SA 4.0	Newsgroups are usually a subscription service. You can find a LOT of options if you search on the web for "usenet newsgroup providers." These groups can also be found on Google groups.
Mailing Lists	A mail list is a discussion group where users communicate via email with others who have subscribed to a particular topic. All users who belong to a mailing list receive every message posted to that list via e-mail as users post questions/answers. Linux topics can be found on several mail lists.	The Linux Foundation hosts a Linux specific mail list. Example: Ubuntu hosts such a distro specific list
Online Communities	Question and answer websites have become a place to get feedback regarding questions on a particular topic. A user will post a question and other users are allowed to comment on the question. Many of these sites have the capability for users to "upvote" questions/responses they feel are useful, or that offer a good solution. The most popular Q&A site for IT topics is Stack Exchange, which has a Unix & Linux category available at https://unix.stackexchange.com/.	Stack Exchange is one of the most popular Linux Q&A sites. .
Online Forums/Social Media	An Internet forum, or message board, is an online discussion site where people can hold conversations in the form of posted messages. They differ from chat rooms in that messages are often longer than one line of text, and are at least temporarily archived. Also, depending on the access level of a user or the forum set-up, a posted message might need to be approved by a	Linux Questions is an example of a forum. Reddit, a network of communities based on people's interests, has a Linux community.

Source Name	Description	Where is it?
	moderator before it becomes publicly visible. "Internet Forum" Wikimedia Foundation is licensed under CC BY-SA 4.0	

This page titled 01-C.25: Asynchronous Linux Help Resources is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

CHAPTER OVERVIEW

Module 02: User & Group Administration

- 02-A.1: Users, and How to Acquire Superuser Power
- 02-A.2: Who Is the Superuser?
- 02-A.3: su: The Command
- 02-A.4: Other Options for Administrative Privileges
- 02-B.5: Users: Create, Modify, and Delete
- 02-B.6: Users: Create, Modify, and Delete (cont'd)
- 02-B.7: Users: Create, Modify and Delete (continued)
- 02-B.8: Users: Create, Modify, and Delete - passwd Command
- 02-B.9: Users: Create, Modify, and Delete - etc passwd and etc shadow
- 02-B.10: Users: Create, Modify, and Delete - Edit /etc/passwd File
- 02-C.11: Groups: Create, Modify, and Delete
- 02-C.12: Groups: Create, Modify, and Delete - Modify Group
- 02-D.13: User and Group Queries
- 02-E.14: User Account Profile Configuration
- 02-E.15: Example: Bash RC file
- 02-E.16: System Wide User Profiles
  - 02-E.16.1: System Wide User Profiles: /etc/profile
  - 02-E.16.2: System Wide User Profiles: The /etc/profile.d Directory
  - 02-E.16.3: System Wide User Profiles: The /etc/bashrc File

---

This page titled Module 02: User & Group Administration is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-A.1: Users, and How to Acquire Superuser Power

EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Module Introduction

As a Linux admin it will be necessary to create user accounts and be able to complete a LOT of duties in order to keep the system running at peak efficiency. To be able to accomplish all of this you will need to assume the identity of superuser: the root user, the system administrator that is allowed to perform any task.

Objectives of this Module

In this lesson, you will:

- Learn how to use superuser privileges as required.
- Learn and practice the creation, modification, and deletion of user accounts.
- Learn and practice the creation, modification, and deletion of group accounts.
- Use the proper commands to query user and group accounts.
- Understand what it means to configure user account profiles.

This page titled 02-A.1: Users, and How to Acquire Superuser Power is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-A.2: Who Is the Superuser?

Superuser

As well as the files that a user can create, Linux systems have a number of files which are needed by the system for it to work properly. If these important system files are changed incorrectly they can cause various things to break, so they are protected from changes by default. Certain applications also modify important parts of the system, and so are also protected.

The way that they are protected is by only allowing users with administrative privileges to change the files or use the applications. In day-to-day use, users will not need to change any system files or use these applications, so by default users do not have administrative privileges.

Sometimes users do need to use these applications, so they may be able to temporarily get administrative privileges to allow them to make the changes. If an application needs administrative privileges, it will ask for a password. For example, if a user wants to install some new software, the software installer (package manager) will ask for your administrator password so it can add the new application to the system. Once it has finished, your administrative privileges will be taken away again.

Administrative privileges are associated with certain user accounts. Administrator users are allowed to have these privileges while standard users are not. Without administrative privileges you will not be able to install software. Some user accounts (for example, the "root" account) have permanent administrative privileges. Users should not use administrative privileges all of the time, because it is possible to accidentally change something that was not intended to (like delete a needed system file, for example).

In summary, administrative privileges allow users to change important parts of the system when needed but prevent users from doing it accidentally.

What Does "Superuser" Mean?

A user with administrative privileges is sometimes called a superuser. This is simply because that user has more privileges than normal users. You might see people discussing things like su and sudo; these are programs for temporarily giving you "superuser" (administrative) privileges.



Important Concept: Principle of Least Privilege

The principle of least privilege is an important concept in information security. It is the practice of limiting access rights for users to the bare minimum permissions they need to perform their work. Linux employs this principle by restricting user permission to read, write or execute only the files or resources they need to do their jobs. In other words, users are given the least amount of privilege necessary. So on Linux, if a user needs to be able to read a file, but should not have the ability to write to the file, they are given read permission, but not write permission. If a certain user needs to be able to reset a user password, but that user should not be allowed to add a new user, they are given the proper permission to do only those tasks that are necessary to complete their job.

Why are Administrative Privileges Useful?

Requiring users to have administrative privileges before important system changes are made is useful because it helps to prevent your system from being broken, intentionally or unintentionally.

If you had administrative privileges all of the time, you might accidentally change an important file, or run an application which changes something important by mistake. Only getting administrative privileges temporarily, when you need them, reduces the risk of these mistakes happening.

Only certain trusted users should be allowed to have administrative privileges. This prevents other users from messing with the computer and doing things like uninstalling applications that you need, installing applications that you don't want, or changing important files. This is useful from a security standpoint.

"How do administrative privileges work?" by Ubuntu Documentation Team is licensed under CC BY-SA 4.0

This page titled 02-A.2: Who Is the Superuser? is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-A.3: su: The Command

The **su** Command

Some Linux distros have made the decision to disable the administrative root account by default in all installations. This does not mean that the root account has been deleted or that it may not be accessed. It merely has been given a password which matches no possible encrypted value, therefore may not log in directly by itself. This simply means that root cannot gain access to the system via the login prompt.

Not to fear, there are still tools available to allow users to gain root user privileges. One method is to use the **su** command. Although **su** can be used to change the ownership of a session to any user, it is most commonly employed to change the ownership from an ordinary user to the root (i.e., administrative) user, thereby providing access to all parts of and all commands on the computer or system. For this reason, it is often referred to (although somewhat inaccurately) as the superuser command. It is also sometimes called the switch user command.

To use the su command one simply issues:

```
pbmac@pbmac-server $ su username
OR
pbmac@pbmac-server $ su - username
```

where username is the user that you desire to switch to. If you do NOT specify a username, it is assumed you are attempting to switch to the root user.

The default behavior of **su** is to maintain the current directory and the environmental variables of the original user (rather than switch to those of the new user). Although the shell account likewise remains that of the original user, any new, unprivileged user (i.e., users other than root and others with some system privileges) does not gain automatic access to the directories or files of the former owner of the session.

So, a user is logged in as pbmac, and the user issues the command: **su bob** It will change the UID to that of bob, but it will still be operating under the environment of pbmac's login (notice it did not change the pwd).

BUT

if the users issues the command:

```
pbmac@pbmac-server $ su - bob
```

it changes the pwd, because the environment is now set according to bob's login. We will talk more later about a user's environment.

```
pbmac@pbmac-server $ pwd
/home/pbmac
pbmac@pbmac-server $ su bob
Password:
bob@pbmac-server:/home/pbmac$ pwd
/home/pbmac
bob@pbmac-server:/home/pbmac$ whoami
bob
bob@pbmac-server:/home/pbmac$ exit
exit
pbmac@pbmac-server $ su - bob
Password:
```

```
bob@pbmac-server:/home/pbmac$ pwd
/home/bob
bob@pbmac-server:/home/pbmac$ whoami
bob
bob@pbmac-server:/home/pbmac$ exit
logout
pbmac@pbmac-server $
```

Sometimes an administrator needs to log in to a user's account as that user. At times, the most efficient way to solve a user's problem is to log into that user's account in order to reproduce or debug the problem.

However, in many situations it is not desirable, or it can even be dangerous, for the root user to be operating from an ordinary user's shell account and with that account's environmental variables rather than from its own. While inadvertently using an ordinary user's shell account, root could install a program or make other changes to the system that would not have the same result as if they were made while using the root account. For instance, a program could be installed that could give the ordinary user power to accidentally damage the system or gain unauthorized access to certain data.

Thus, it is advisable that administrative users, as well as any other users that are authorized to use **su** (of which there should be very few, if any), acquire the habit of always following the su command with a space and then a hyphen. The hyphen has two effects: (1) it switches the current directory to the home directory of the new user (e.g., to */root* in the case of the root user) and (2) it changes the environmental variables to those of the new user. That is, if the first *argument* to **su** is a hyphen, the current directory and environment will be changed to what would be expected if the new user had actually logged on to a new session (rather than just taking over an existing session).

This page titled 02-A.3: su: The Command is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-A.4: Other Options for Administrative Privileges

The sudo Command

The **su** command provides a powerful tool to allow users to be able to perform system administrative tasks. However, this method gives more power to users than is really required. A user operating in root's shell has the capability of accidently breaking things. There is a better way to allow users to perform certain functions: this is done through the **sudo** command. With the **sudo** command we can allow users to run commands with administrative privileges, without giving them root's password. We can also allow certain users to run only certain commands with root permissions. We will use sudo in many of the labs, and we will continue to see it used as we go through this course.

To allow a user permission to use the **sudo** command, they must be a member of the sudo group as specified in the /**etc/group** file. Some distros allow for any user in the admin group to have **sudo** permission. It is best though, to keep your configuration file, **/etc/sudoers,** file as simple as possible. It is also possible for individual users or groups to have an entry in the **/etc/sudoers** file that allows them access to the **sudo** command.

To run the sudo command you simply provide any options, if any, and the command you desire to run as root: **sudo options command**

The sudoedit Command

There are some files in the Linux world that have permissions set in such a manner that you need root privileges to edit them. This can be accomplished by using the **sudo** command with any of the editors; however, the preferred solution is to use the **sudoedit** command. Using sudo with an editor can have the unwanted side effect of changed ownership and permissions on some of your own configuration files.

To use the sudoedit command a user must have an entry in the /etc/sudoers file. For instance if you are a member of the group www - that is, your username is listed in the /etc/group file in the group www, then the following line would allow you to access the specified file with administrative privileges:

%www ALL = sudoedit /path/to/file

To run the **sudedit** command: **sudoedit options filename**

The more a Linux administrator makes use of these techniques and tools, the more secure the system will be.

The visudo Command

While we use the **sudoedit** command to make changes to many system files, we have the **visudo** command specifically to edit the **/etc/sudoers** file. This file is necessary for a user to gain root access on a Linux system. If it contains any errors, it would be impossible for any user

```
pbmac@pbmac-server $ sudo visudo
>>> /etc/sudoers: syntax error near line 28 <<<
What now? ?
Options are:
   (e)dit sudoers file again
   e(x)it without saving changes to sudoers file
   (Q)uit and save changes to sudoers file (DANGER!)

what now? x
pbmac@pbmac-server $
```

to be able to use the sudo command to gain administrative privileges. Therefore, there is a special command that not only allows the editing of the file but checks for errors before writing any changes back to the file. If there are errors, the user will get a notice,

and will be asked what they want to do. They can return to edit mode, they can exit the edit session losing their changes, or they can save the changes (NOT recommended).

The Wheel Group

The wheel group is a special user group used on some Unix systems (mostly BSD systems) to control access to the su or sudo command, which allows a user to masquerade as another user (usually the super user). Debian-like operating systems creates a group called sudo with similar purpose to wheel group.

In some distros of Linux the wheel group is necessary to use the **`sudo`** command. On these systems you run the **`visudo`** command to edit the privileges of the wheel group. New users can also be added to the wheel group as necessary to enable them to operate with elevated privileges.

Adapted from:
"Wheel (computing)" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

This page titled 02-A.4: Other Options for Administrative Privileges is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-B.5: Users: Create, Modify, and Delete

EXAM OBJECTIVES COVERED

2.2 Given a scenario, manage users and groups.

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Create Users

If we are to use Linux, we must be able to create new user accounts. To do this we use the **`useradd`** command. In addition this command allows us to assign basic settings for a user. Many of the Debian distros use the **`adduser`** command instead, although the **`useradd`** command is still available.

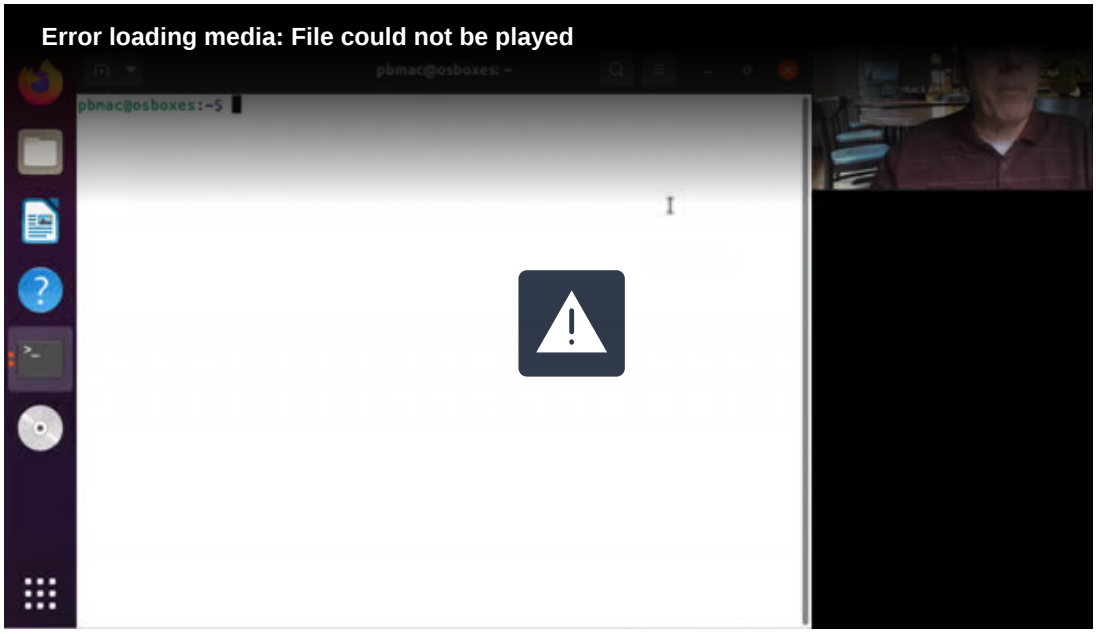
`adduser` is not a standard Linux command. It's essentially a Perl script that uses the **`useradd`** command in the background. This high-level utility is more efficient in properly creating new users on Linux. Default parameters for all new users can also be set through the **`adduser`** command. Since we are focusing on standard Linux, we will discuss the **`useradd`** command.

There are several files that are used in the creation of a new user:

- /etc/passwd - User account information.
- /etc/shadow - Secure user account information.
- /etc/group - Group account information.
- /etc/gshadow - Secure group account information.
- /etc/default/useradd - Default values for account creation.
- /etc/skel/ - Directory containing default files.
- /etc/subgid - Per user subordinate group IDs.
- /etc/subuid - Per user subordinate user IDs.
- /etc/login.defs - Shadow password suite configuration.

Depending on the contents and configuration of the **`/etc/login.defs`** file, the **`useradd`** command may not actually create a home directory. The new user's shell is set to the default **`/bin/sh`** and a password will not be created for the new user.

This video walks through the process of adding a user in detail and shows examples.



This page titled 02-B.5: Users: Create, Modify, and Delete is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-B.6: Users: Create, Modify, and Delete (cont'd)

The useradd Command

The **useradd** command is used to create user accounts and configure basic settings. The basic usage of **useradd** is quite simple: A user can be added just by providing their username.

Syntax:

```
useradd [ OPTIONS ] USER_LOGIN
```

Command Options

Options	Meaning
-b, --base-dir BASE_DIR	The default base directory for the system if -d HOME_DIR is not specified. BASE_DIR is concatenated with the account name to define the home directory. If the -m option is not used, BASE_DIR must exist. If this option is not specified, useradd will use the base directory specified by the HOME variable in /etc/default/useradd, or /home by default.
-c, --comment COMMENT	Any text string. It is generally a short description of the login, and is currently used as the field for the user's full name.
-d, --home HOME_DIR	The new user will be created using HOME_DIR as the value for the user's login directory. The default is to append the LOGIN name to BASE_DIR and use that as the login directory name. The directory HOME_DIR does not have to exist but will not be created if it is missing.
-D, --defaults	See below, the subsection "Changing the default values."
-e, --expiredate EXPIRE_DATE	The date on which the user account will be disabled. The date is specified in the format YYYY-MM-DD.
-s, --shell SHELL	The name of the user's login shell. The default is to leave this field blank, which causes the system to select the default login shell specified by the SHELL variable in /etc/default/useradd, or an empty string by default.

As an example, to add a new user, pbmac, the following command is used:

```
$ sudo useradd pbmac
```

In this example, the **useradd** command creates an account called pbmac. A group with the same name is also created, and the user pbmac is placed in this new group and is the primary group for this user. There are other parameters, such as language and shell, that are applied according to defaults and values set in the configuration files **/etc/default/useradd** and **/etc/login.defs**. This is generally sufficient for a single, personal system or a small, one-server business environment.

While the two files above govern the behavior of **useradd**, user information is stored in other files found in the **/etc** directory.

File	Description	Fields (bold—set by useradd)
/etc/passwd	Stores user account details	**username**:unused:**uid**:**gid**:**comment**:**home dir**:**shell**
/etc/shadow	Stores user account security details	**username**:password:lastchange:minimum: maximum:warn:**inactive**:**expire**:unused
/etc/group	Stores group details	**groupname**:unused:**gid**:**members**

The **useradd** command line allows customization for times when an administrator needs finer control, such as to specify a user's ID number.

(Starting in this next section, the **getent** command produces output in a specific format. We will talk more about this when we discuss the password command.)

User and Group ID Numbers

By default, **useradd** tries to use the same number for the user ID (UID) and primary group ID (GID), however, this is NOT guaranteed. Although it's not necessary for the UID and GID to match, it's easier for administrators to manage them when they do. (**ATTENTION** - memorize UID and GID, these are **important** acronyms for Linux admins)

Suppose we add another account, this time for santiago. Comparing the two users, pbmac and santiago, shows that both users and their respective primary groups were created by using the **getent** command.

```
$ getent passwd pbmac santiago
pbmac:x:1001:1002:pbmac:/home/pbmac:/bin/bash
santiago:x:1002:1003::/home/santiago:/bin/bash

$ getent group pbmac santiago
pbmac:x:1002:
santiago:x:1003:
```

Unfortunately, neither users' UID nor primary GID match. This is because the default behavior is to assign the next available UID to the user and then attempt to assign the same number to the primary group. However, if that number is already used, the next available GID is assigned to the group. To understand what happened, we can assume that a group with GID 1001 already exists - we can enter a command to confirm:

```
$ getent group 1001
book:x:1001:brandon
```

The group book with the ID 1001 has caused the GIDs to be off by one. This is an example where a system administrator would need to take more control of the user-creation process. To resolve this issue, you must first determine the next available user and group ID that will match. The commands **getent group** and **getent passwd** will be helpful in determining the next available number. This number can be passed with the **-u** argument.

```
$ sudo useradd -u 1004 maida

$ getent passwd maida; getent group maida
maida:x:1004:1004::/home/maida:/bin/bash
maida:x:1004:
```

"Intro to the Linux useradd command" by Alan Formy-Duval, opensource.com is licensed under CC BY-SA 4.0

This page titled 02-B.6: Users: Create, Modify, and Delete (cont'd) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-B.7: Users: Create, Modify and Delete (continued)

More `useradd` Customization

Even though the useradd command is very basic, it is possible for options to be specified as a new user is created. Let's take a look at some brief examples of the most common customizations used.

Comment

The comment option is a plain-text field for providing a short description or other information using the -c argument.

```
$ sudo useradd -c "Sean is cool" sean
$ getent passwd sean
sean:x:1011:1011:sean is cool:/home/sean:/bin/bash
```

Groups

A user can be assigned one primary group and multiple secondary groups. The -g argument specifies the name or GID of the primary group. If it's not specified, **`useradd`** creates a primary group with the user's same name (as demonstrated above). The -G (uppercase) argument is used to pass a comma-separated list of groups that the user will be placed into; these are known as secondary groups.

```
$ sudo useradd -G mygrp,fgroup,libvirt emma
$ id emma
uid=1012(emma) gid=1012(emma) groups=1012(emma),981(libvirt),4000(fgroup),3000(tgroup)
```

Home directory

The default behavior of **`useradd`** is to create the user's home directory in /home. However, different aspects of the home directory can be overridden with the following arguments. The -b sets another directory where user homes can be placed. For example, /home2 instead of the default /home.

```
$ sudo useradd -b /home2 hallie
$ getent passwd hallie
hallie:x:1013:1013::/home2/hallie:/bin/bash
```

The **-d** lets you specify a home directory with a different name from the user.

```
$ sudo useradd -d /home/ben nick
$ getent passwd nick
nick:x:1014:1014::/home/ben:/bin/bash
```

The skeleton directory

The -k instructs the new user's new home directory to be populated with any files in the /etc/skel directory. These are usually shell configuration files, but they can be anything that a system administrator would like to make available to all new users.

Shell

The -s argument can be used to specify the shell. The default is used if nothing else is specified. For example, in the following, shell bash is defined in the default configuration file, but david has requested zsh.

```
$ grep SHELL /etc/default/useradd
SHELL=/bin/bash
```

```
$ sudo useradd -s /usr/bin/zsh david
$ getent passwd david
david:x:1004:1004::/home/david:/usr/bin/zsh
```

Security

Security is an essential part of user management, so there are several options available with the **useradd** command. A user account can be given an expiration date, in the form YYYY-MM-DD, using the -e argument.

```
$ sudo useradd -e 20191231 sammy
$ sudo getent shadow sammy
sammy:!!:18171:0:99999:7::20191231:
```

An account can also be disabled automatically if the password expires. The -f argument will set the number of days after the password expires before the account is disabled. Zero is immediate.

```
$ sudo useradd -f 30 seth
$ sudo getent shadow seth
seth:!!:18171:0:99999:7:30::
```

"Intro to the Linux useradd command" by Alan Formy-Duval, opensource.com is licensed under CC BY-SA 4.0

This page titled 02-B.7: Users: Create, Modify and Delete (continued) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-B.8: Users: Create, Modify, and Delete - passwd Command

The **passwd** Command

Changing user password in Linux is quite simple using the **passwd** command.

Syntax:

```
passwd [ OPTIONS ] username
```

Command Options:

Options	Meaning
-u	This is the reverse of the -l option - it will unlock the account password by removing the ! prefix. This option is available to root only. By default passwd will refuse to create a passwordless account (it will not unlock an account that has only "!" as a password). The force option -f will override this protection.
-d	This is a quick way to delete a password for an account. It will set the named account passwordless. Available to root only.
-e	This is a quick way to expire a password for an account. The user will be forced to change the password during the next login attempt. Available to root only.

If a user want to change their current password (assuming the user's id is pbmac), simply run this command in a terminal:

```
passwd
OR
passwd <username> - where you replace <username> with a valid username to change the p
```

The system will ask the user to enter their current password and the new password twice.

The user won't see anything on the screen while typing the password. This is perfectly normal behavior for Linux.

```
$ passwd
Changing password for pbmac.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Since this is an admin account, the **sudo** password just got changed without even realizing it.

If you want to change the password for some other user, you can do that as well with the **passwd** command. But in this case, you'll have to use **sudo**.

```
$ sudo passwd santiago
Changing password for santiago.
(current) UNIX password:
Enter new UNIX password:
```

```
Retype new UNIX password:
passwd: password updated successfully
```

This page titled 02-B.8: Users: Create, Modify, and Delete - passwd Command is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-B.9: Users: Create, Modify, and Delete - etc passwd and etc shadow

The /etc/passwd File

The /etc/passwd file stores information about all user accounts on the system. It is a colon-separated file that contains important information:

- Username - the name the user logs into the system with.
- Encrypted password - in the /etc/passwd file this is usually represented by an 'x,' the encrypted password is stored elsewhere.
- User ID number (UID) - the unique user ID number, the system uses this number to represent the user.
- User's group ID number (GID) - the unique group ID number that signifies the user's primary group.
- Full name of the user (GECOS) - this field was historically used to contain specific user information such as name, office, telephone number, etc.
- User home directory - the absolute path of the user's home directory.
- Login shell - the absolute path to the file that gets executed when this user logs in.

A typical /etc/passwd file is shown below. Notice: 1) the password field is an 'x' for every ID; 2) more of these users have /usr/sbin/nologin as their shell - they are NOT allowed to login using this userid 3) syslog has a login shell of /bin/false, and the root user and pbmac have /bin/bash as their login.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
syslog:x:101:103::/home/syslog:/bin/false
pbmac:x:1000:1000:pbmac,,,:/home/pbmac:/bin/bash
```

Looking at the userid pbmac, the entry is broken down as:

```
  pbmac:x:1000:1000:,,,:/home/pbmac:/bin/bash
  [---] - [--] [--] [-] [---------] [-------]
    |   |   |    |   |       |          |
    |   |   |    |   |       |       Login shell
    |   |   |    |   |  Home directory
    |   |   |    | GECOS
    |   |   |   GID
    |   |  UID
    | Password
 Username
```

The /etc/shadow file contains the password information for the system's accounts and optional aging information. This file must not be readable by regular users if password security is to be maintained. Each line of this file contains 9 fields, separated by colons.

Here is a brief description of each field:

- Username – the name of the user.
- Encrypted password – the password in encrypted form.
- Last password change – the date of the last password change. This date is stored as the number of days since January 1, 1970.
- Minimum – the number of days before a password change is allowed. The value of 0 means the password can be changed any time.
- Maximum – the number of days before the password must be changed. The value 99999 means the user's password never expires.
- Warning – the number of days before a password is going to expire during which the user will be warned.
- Disabled – the number of days after a password has expired until the user account is disabled. No entry in this field means that the account is disabled immediately after the password expires.
- Disabled date – the number of days since January 1, 1970 that the account has been disabled. No entry in this field means the account is not disabled.

```
pbmac:$6$kgBH9Jca$6Jf7V1qc27CYSAHiSUuRGGOxT5g0VaXYZUDf5ySgDGm5.2PsNZLdqdw43jlnyGHNb/:1
[---] [-------------------------------------------------------------------------------] |
username                                  hashed value of user's password

                                                                                      s:

                                                                                      ch



                                                                                      c
```

This page titled 02-B.9: Users: Create, Modify, and Delete - etc passwd and etc shadow is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-B.10: Users: Create, Modify, and Delete - Edit /etc/passwd File

Modify /etc/passwd and /etc/Group Files

The **vipw** Command

The **vipw** command edits the files /etc/passwd. The command sets appropriate locks to prevent corruption by others attempting to make changes to the file during the editing session. The programs attempt to determine the proper editor, by first trying the environment variable $VISUAL, then the environment variable $EDITOR, and finally the default editor which is vi.

Syntax:

```
vipw [ OPTION ]
```

NOTICE: the user does NOT specify a filename...there are no arguments.

Command Options:

Options	Option Meaning
-g, --group	Edit group database.
-h, --help	Display help message and exit.
-p, --passwd	Edit passwd database.
-q, --quiet	Quiet mode.
-R, --root CHROOT_DIR	Apply changes in the CHROOT_DIR directory and use the configuration files from the CHROOT_DIR directory.
-s, --shadow	Edit shadow or gshadow database.

To safely edit the /etc/password file, simply run:

```
pbmac@pbmac-server $ sudo vipw
```

The password will not get updated until you exit the vipw command. This ensures that other users can not inadvertently or intentionally make any changes in /etc/passwd file while you're editing them as root. The vipw command also ensures that once the /etc/passwd is closed, the shadow password file is automatically updated successfully.

You can also login as normal user and try to change the the user's password in another Terminal window. The password will not change until you close the vipw command.

You can't even create any new users while editing /etc/passwd file vipw command.

As stated above, the **vipw** command will lock the /etc/passwd file and prevent other users from making any changes. It is also possible to make the necessary changes using the **useradd**, the **usermod**, and the **userdel** commands to add, change, and delete users on the system. We have previously talked about the **useradd** command.

The **usermod** Command

The **usermod** command allows for certain changes and modifications be made to a user's information from the command line.

Syntax:

```
usermod [ OPTIONS ] USER_LOGIN
```

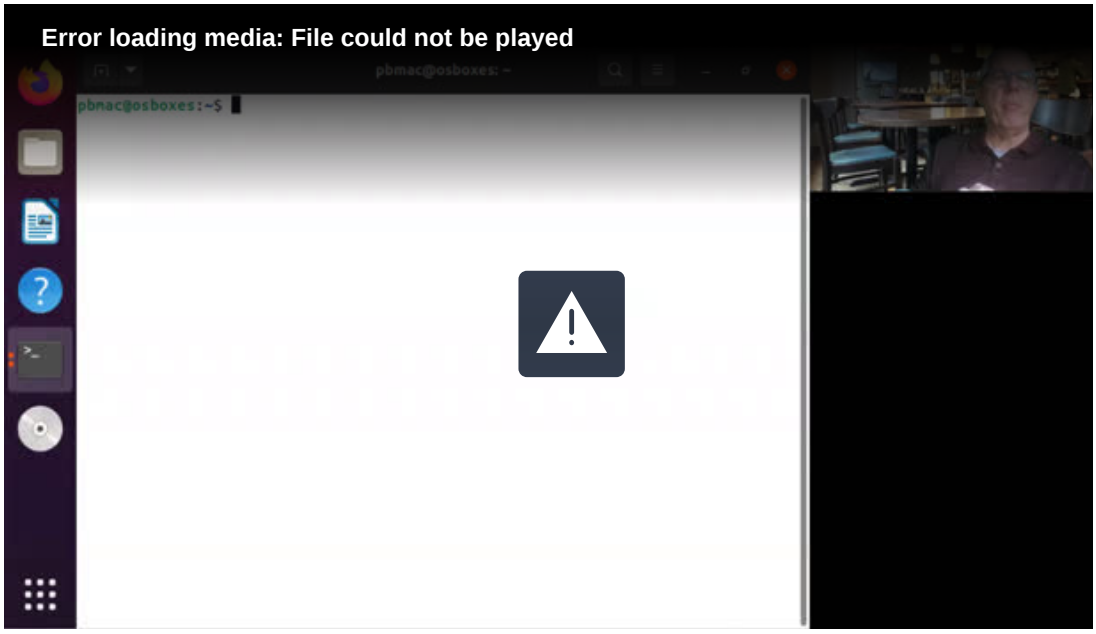
Command Options:

Options	Option Meaning
-d, --home HOME_DIR	The user's new login directory. If the -m option is given, the contents of the current home directory will be moved to the new home directory, which is created if it does not already exist.
-e, --expiredate EXPIRE_DATE	The date on which the user account will be disabled. The date is specified in the format YYYY-MM-DD.
-f, --inactive INACTIVE	The number of days after a password expires until the account is permanently disabled. A value of 0 disables the account as soon as the password has expired, and a value of -1 disables the feature.
-g, --gid GROUP	The group name or number of the user's new initial login group. The group must exist.
-G, --groups GROUP1[,GROUP2,...[,GROUPN]]]	A list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given with the -g option.

A simple example would be the changing of a user's home directory:

```
pbmac@pbmac-server $ usermode -d /home/manav test_user
```

The following video discusses the use of the usermod command, with some examples of how it can be used from the command line.



The `userdel` Command

The userdel command in Linux system is used to delete a user account and related files. This command basically modifies the system account files, deleting all the entries which refer to the username LOGIN. It is a low-level utility for removing the users.

Syntax:

```
userdel [ OPTIONS ] USER_LOGIN
```

Command Options:

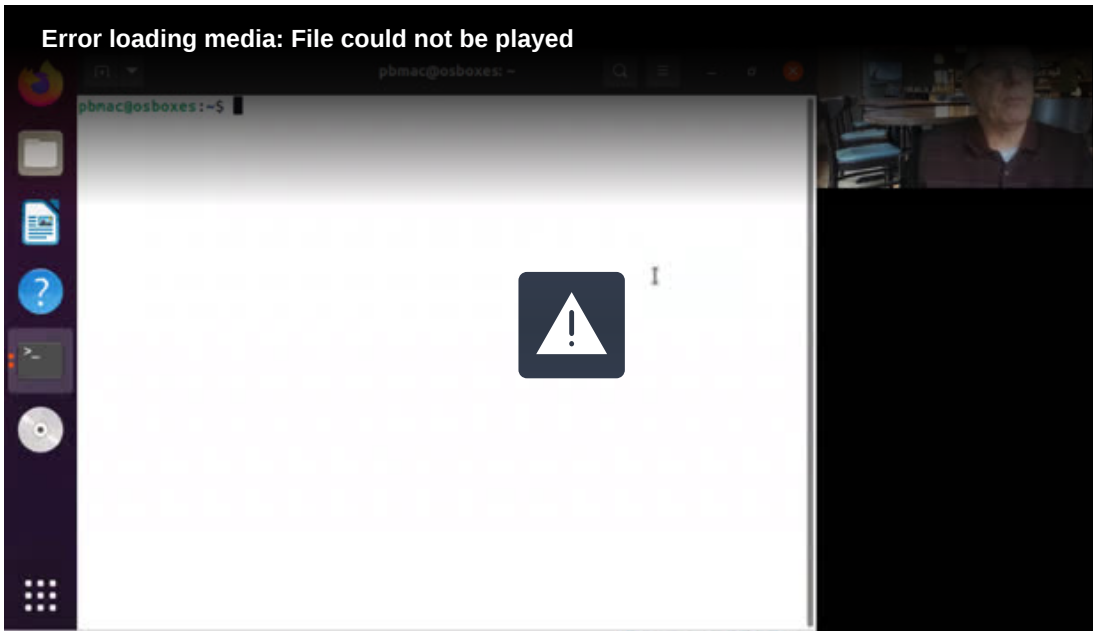
Options	Option Meaning
`-f, --force`	This option forces the removal of the user account, even if the user is still logged in.
`-r, --remove`	Files in the user's home directory will be removed along with the home directory itself and the user's mail spool. Files located in other file systems will have to be searched for and deleted manually.

With few options the command is quite simple:

```
pbmac@pbmac-server $ userdel -f pbmac
```

This will force the user pbmac to be removed from the system.

A discussion with examples regarding the use of the userdel command. The implications of not deleting users is covered in this video as well.



Adapted from:
"The Right Way To Edit /etc/passwd And /etc/group Files In Linux" by SK, OSTECHNIX is licensed under CC BY-NC 4.0
"usermod command in Linux with Examples" by manav014, Geeks for Geeks is licensed under CC BY-SA 4.0
"userdel command in Linux with Examples" by rahulkumarmandal, Geeks for Geeks is licensed under CC BY-SA 4.0

This page titled 02-B.10: Users: Create, Modify, and Delete - Edit /etc/passwd File is shared under a not declared license and was authored, remixed, and/or curated by Patrick McClanahan.

02-C.11: Groups: Create, Modify, and Delete

EXAM OBJECTIVES COVERED

2.2 Given a scenario, manage users and groups.

Linux Groups

In the previous task we discussed Linux users. There is another concept in Linux, that of groups, which is basically a logical mechanism to manage a collection of users. The idea of groups is to make it easy to manage several users that need access and permissions to a group of files, or commands on the system. The admin can assign each user to that group and allow them to have group access to files and commands. This is where the GID comes into use: it signifies which group(s) a user belongs to, and thereby, which access and permissions that user has.

The /etc/group File

This file, like the /etc/passwd file, is a colon separated file. It is much simpler than the passwd file, as it only has 4 fields:

- **Groupname** - Contains the name assigned to the group.
- **Group-password** (x) - x in this field indicates that shadow passwords are used.
- **GID** - Contains the group's GID number.
- **Username-list** - Comma separated list of users that are members of the group.

A typical /etc/groups file is shown below:

```
security-team:x:133:pbmac, santiago, maida
faculty:x:134:lisa, kathleen, salvador
administration:x:135:laura, james
```

Here is a breakdown of the fields from the first line in the example above.

```
security-team:x:133:pbmac, santiago, maida
[-----------] - [-] [---------------------]
     |        |  |            |
     |        |  |            |
     |        |  |    userid for those members of this group - there may be NO userids
     |        | GID
     |   group password
  group name
```

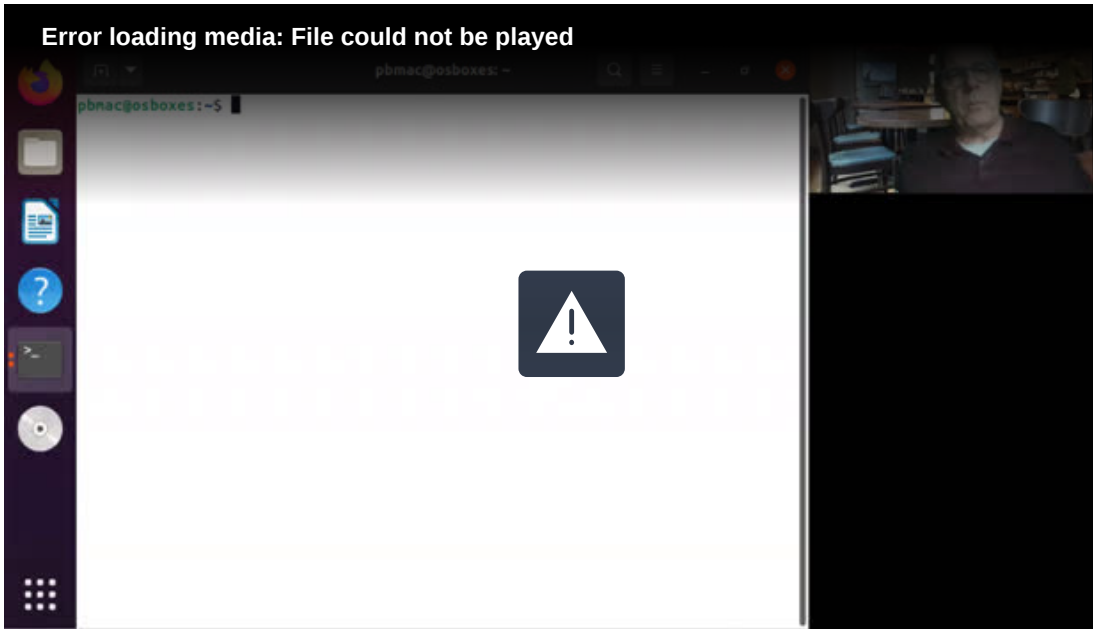
This page titled 02-C.11: Groups: Create, Modify, and Delete is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-C.12: Groups: Create, Modify, and Delete - Modify Group

Modify /etc/group File

The proper way to edit **/etc/group** file is using **`vigr`** commands in Linux. We CANNOT simply edit the groups file with a text editor. The **`vigr`** command is used to safely edit the files /etc/group file. It will put appropriate locks when editing /etc/group file to prevent other users from making any changes in the file and avoid file corruption. When you run this command with **-s** flag, it will edit the /etc/gshadow file.

As we have the ability to add, modify and delete users, we also have the ability to add, modify and delete groups. This video discusses these capabilites and shows examples of each.



The `vigr` Command

The **`vigr`** command edits the /etc/groups file in the same manner that the vipw command edits the /etc/passwd file. The discussion can be found on the preceding page.

To safely edit /etc/groups file, simply run:

```
pbmac@pbmac-server $ sudo vigr
```

Now **`vigr`** command will lock the /etc/group files and prevent other users from making any changes. The password will not get updated until you exit the **`vigr`** command. So the other users may not inadvertently or intentionally make any changes in /etc/group file while you're editing them as root.

Modify /etc/group File From the Command Line

Just like with **`useradd/usermod/userdel`** commands for users, there are similar commands for groups: **`groupadd`**, **`groupmod`**, and **`groupdel`**. These commands allow any admin to add, modify or delete groups.

The `groupadd` Command

As the name says, this command creates a group. The default for **`groupadd`** is that the group will not have any members. NOTE: The **`groupadd`** command does NOT add users to a group; for that task use the **`useradd`** or **`usermod`** command.

Syntax:

```
groupadd [ OPTIONS ] GROUP_NAME
```

Command Options:

Option	How is it used...	Show me
-f	The command exits with a status of "success" if the specified group exists. When used with the -g, and the specified GID exists, another unique GID is chosen.	`groupadd -f security-team`
-g	Assign a specific group id, if not specified groupadd uses the next in the sequence.	`groupadd -g 217 security-team`
-o	Allows for the creation of a non-unique GID.	`groupadd -o -g 217 sales`

To add a new group named developers:

```
pbmac@pbmac-server $ groupadd -f developers
```

The `groupmod` Command

The **`groupmod`** command gives system administrators the ability to change the group's own attributes. The changes are made to the /etc/group.

Syntax:

```
groupmod [ OPTIONS ] GROUP_NAME
```

Command Options:

Option	How is it used...	Show me
-g	The group id will be changed to the given value for the specified group.	`groupmod -g 999 security-team`
-n	The name of the specified group will be changed to the new group name.	`groupmod -n teamsec security-team`
-o	Allows for the creation of a non-unique GID.	`groupmod -o -g 217 sales`

To change the name of our developers group to dev_team we use a simple command using th e-n option:

```
pbmac@pbmac-server $ groupmod -n dev_team developers
```

The groupdel Command

The **`groupdel`** command modifies the system account files, deleting all entries that refer to the specified group. The named group must exist.

Syntax:

```
groupdel [ OPTIONS ] GROUP_NAME
```

Command Options:

Options	Option Meaning
-g, --gid GID	The group ID of the given GROUP will be changed to GID. Any files that have the old group ID and must continue to belong to GROUP must have their group ID changed manually.
-n, --new-name NEW_GROUP	The name of the group will be changed from GROUP to NEW_GROUP name.
-o, --non-unique	When used with the -g option, allows to change the group GID to a non-unique value.
-p, --password PASSWORD	The encrypted password, as returned by crypt(3).

An example of deleting the existing group sales:

```
pbmac@pbmac-server $ groupdel dev_team
```

Be careful when deleting groups as it may cause users to have issues accessing resources. Also, take note that this command does NOT delete the users associated with the deleted group.

"The Right Way To Edit /etc/passwd And /etc/group Files In Linux" by SK, OSTECHNIX is licensed under CC BY-NC 4.0
"groupmod command in Linux with examples" by Samdare B, Geeks for Geeks is licensed under CC BY-SA 4.0
"groupadd command in Linux with examples" by NaveenJangid, Geeks for Geeks is licensed under CC BY-SA 4.0
"groupdel command in Linux with examples" by Samdare B, Geeks for Geeks is licensed under CC BY-SA 4.0

This page titled 02-C.12: Groups: Create, Modify, and Delete - Modify Group is shared under a not declared license and was authored, remixed, and/or curated by Patrick McClanahan.

02-D.13: User and Group Queries

EXAM OBJECTIVES COVERED

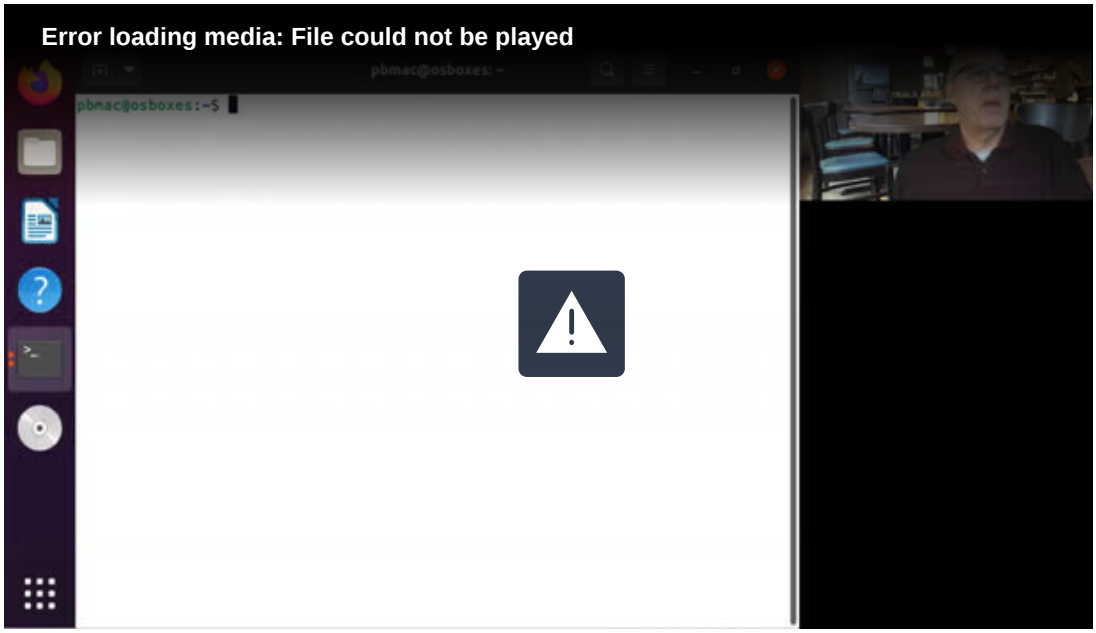
2.2 Given a scenario, manage users and groups.

User and Group Queries

Adding users and groups is only a part of the job. Now that you have users/groups created you need to administer them all. As an administrator you have a set of commands that you will get very familiar with which will help you care for your users.

These commands shown here do have options, but are seldom used with those options, so there is not a more in-depth description of them.

To see how these commands work, and when you might want to use them take a look at this video.



The `whoami` Command

This command prints the username that is associated with the current effective user ID.

Notice in the image to the right that user **pbmac** issues the **whoami** command and it prints **pbmac**. The user then issues a **sudo su** command to switch to the root user and issues the **whoami** command again - now it responds with **root**.

```
pbmac@pbmac-server $ whoami
pbmac
pbmac@pbmac-server $ sudo su
[sudo] password for pbmac:
root@pbmac-server:/home/pbmac/Documents# whoami
root
```

Identification Via the Prompt

Also, notice that on this system the prompt shows the user name - **pbmac**, then after the **su** command was entered the prompt shows **root.** This is a configurable parameter, and so this might not be the same on all systems you work on. The prompt also shows the current directory when the user is logged in as root; it changes from **/home/pbmac** to **/etc** after the **cd** command.

The `who` Command

Use this command to find out which users are currently logged into the system. The **who** command has several options but is usually entered without using any. The command shows the username, the name of the system where the user is logged in from - in this case **pbmac** is directly logged in, and **santiago** is logged in from 192.168.1.7 onto the psuedo port 3 - and the date and time that user was logged into the system.

```
pbmac@pbmac-server $ who
pbmac    :0           2020-11-16 09:48 (:0)
santiago pts/3        2020-11-16 16:17 (192.168.1.7)
```

The `w` Command

The **w** command displays information about the users currently on the machine and their processes. The header shows, in this order, the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

The following entries are displayed for each user: login name, the tty name, the remote host, login time, idle time, JCPU, PCPU, and the command line of their current process.

The JCPU time is the time used by all processes attached to the tty. It does not include past background jobs, but does include currently running background jobs.

The PCPU time is the time used by the current process, named in the "what" field.

```
pbmac@pbmac-server $ w
 17:21:23 up 25 days,  7:34,  1 user,  load average: 0.31, 1.00, 1.72
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
pbmac    :0       :0               16Nov20 ?xdm?   1:50m  0.00s /usr/lib/gdm
santiago pts/3    192.168.1.7      16:26    21:00s 0.08s  0.08s -bash
```

The `id` Command

A simple command to show the user and group information for the specified user, or if the username is omitted, it will show information for the current user.

```
pbmac@pbmac-server $ id
uid=1000(pbmac) gid=1000(pbmac) groups=1000(pbmac),4(adm),20(dialout),24(cdrom),27(sudo),46(plugd
107(lpadmin),108(sambashare),109(admin),131(wireshark),133(vboxusers),148(lxd),1001(samba),1004(c
```

The `last` Command

The **last** command in Linux is used to display the list of all the users logged in and out since the file **/var/log/wtmp** was created. One or more usernames can be given as an argument to display their login in (and out) time and their host-name.

```
pbmac@ubuntu $ last
pbmac    tty2         tty2             Tue Jul 12 15:28   still logged in
reboot   system boot  5.4.0-1065-raspi Wed Dec 31 16:00   still running
pbmac    tty2         tty2             Mon Jul 11 18:28 - down   (21:00)
reboot   system boot  5.4.0-1065-raspi Wed Dec 31 16:00 - 15:28 (19185+22:28)
pbmac    tty2         tty2             Mon Jul 11 09:59 - down   (08:27)
reboot   system boot  5.4.0-1065-raspi Wed Dec 31 16:00 - 18:27 (19185+01:27)
santiago tty4         tty4             Sat Jul  9 17:31 - still logged in
reboot   system boot  5.4.0-1065-raspi Wed Dec 31 16:00 - 09:58 (19184+16:58)
santiago tty4         tty4             Tue Jul  5 16:47 - down  (4+00:12)
pbmac    :0           :0               Tue Jul  5 16:21 - 16:46  (00:25)
```

"last command in Linux with Examples" by basilmohamed, Geeks for Geeks is licensed under CC BY-SA 4.0

This page titled 02-D.13: User and Group Queries is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-E.14: User Account Profile Configuration

What Defines a User's Profile?

In the Linux world each user can define what their working environment looks like, to an extent. There are certain files that get run when a user logs in to a system. We can break the files into system profiles and individual profiles. There are some files in the /etc directory: /etc/profile, /etc/bash.bashrc (some systems use /etc/bashrc) that will set system wide preferences. Check your man pages for the Bash shell (man bash) and it should give you specific details about your distribution and how it is configured.

What Do /etc/profile and ~/.profile Do?

Profiles mainly are used to load environment variables. Since profiles are loaded by login shells, and login shells are the initial entry point into a system, that's the time when setting up the environment makes the most sense. One of the biggest environment variables is the PATH variable. When a login shell is initiated, the PATH is set. Other environment variables also can be set in the system-wide profile or individual user profiles, but just know that the profile system is where most variables are set.

The order with which profile information is loaded is very important, because if you want to override the system-wide default profile information, you can do so by specifying environment variables in your personal user profile script. For instance, the PATH variable is usually modified by the user's profile script on login. Usually, the .profile (or .bash_profile, etc., see above) script will add ~/bin to the PATH variable if users have their own bin folder inside their home directory. Because user profiles are loaded after the system-wide profile, user settings take precedent and override system-wide settings.

What Do /etc/bash.bashrc and .bashrc Do?

The system-wide bashrc file and then the individual user's .bashrc script usually set personal preferences for the command line. If a user wants to customize their prompt, or prefer a specific color scheme, the bashrc system is where that would be set. Like the profile file described above, the user's .bashrc file overrides the system-wide /etc/bash.bashrc (or /etc/bashrc) settings. That means a user can customize the behavior of their command line in any manner they desire without impacting other users on the system.

The most common customization inside the .bashrc file is to add aliases. An alias is sort of like text expansion, in that it substitutes your defined alias with whatever command you specify. For example, here's a snippet from a .bashrc file in the user's folder:

```
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
```

The aliases make it so that if the user types `ll` on the command line, the system will execute `ls -alF` instead. It's a great way to make shortcuts for commands with cryptic options or shortcuts for commands you type often.

This page titled 02-E.14: User Account Profile Configuration is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-E.15: Example: Bash RC file

EXAM OBJECTIVES COVERED

2.2 Given a scenario, manage users and groups.

The `~/.bashrc` file

This file allows the user to modify their environment without impacting other users on the system. Every user's home directory has a **`.bashrc`** file - that is why it is often referred to with the tilde: **`~/.bashrc.`** Users can edit this file and add their own customizations, even overriding system settings that are set in the /etc/bash.bashrc file.

```
# ~/.bashrc: executed by bash(1) for non-login shells.

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

export PATH="$HOME/scripts:$PATH"
export EDITOR=/usr/bin/vi
```

Looking at the pieces:

Some environment variables are set:

- HISTSIZE is the number of lines or commands that are stored in memory in a history list while your bash session is ongoing.
- HISTFILESIZE is the number of lines or commands that (a) are allowed in the history file at startup time of a session, and (b) are stored in the history file at the end of your bash session for use in future sessions.

There are 3 aliases set. Aliases allow users to customize commands to save on typing. In this file there is a command the user creates called **`ll`** which actually executes as **`ls -alF .`** A user can basically create their own commands based on existing Linux commands and the command options.

An if statement behaves just like an if statement in any programming language. This particular if statement says, 'if the file ~/.bash_aliases exists then we execute that file.' If it does NOT exist, nothing happens.

The last two entries are setting environment variables to a new value. The first one is adding the directory /home/pbmac/scripts to the PATH variable, which is the list of directories searched when I issue a command. This allows me to create my own customer scripts to run and be able to run them easily from the command line. The other export is for the EDITOR variable - which is used by the system when I do things like edit the password file. This value tells the system I want to use the vi editor.

The ~/bash_profile

This file is slightly different from the **~/.bashrc** in that it is only executed when the user initially logs into the system. This file is NOT always present, in which case the **~/.bashrc** file is always used.

What is Difference Between ~/.bashrc and ~/.bash_profile

The **.bash_profile** is read and executed when Bash is invoked as an interactive login shell (meaning when the user logs into the system from the login prompt), while the **.bashrc** is executed for an interactive non-login shell, such as when the user runs a script that they have written.

A good rule of thumb then is to use .**bash_profile** to run commands that should run only once, such as customizing the $PATH environment variable.

Put the commands that should run every time the user launches a new shell in the **.bashrc** file. This would include a user's aliases, customizations to the prompt, history customizations, etc.

Typically **~/.bash_profile** checks for the existence of the .~/.bashrc file, and if it is present it runs it. We saw this in the example above with the if statement and the .bash_aliases file.

```
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
```

Most Linux distributions are using the **~/.profile** instead of **~/.bash_profile** because the ~/.profile is read by all shells, while **~/.bash_profile i**s only used by the Bash shell.

This page titled 02-E.15: Example: Bash RC file is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-E.16: System Wide User Profiles

Global Profiles

In Linux, settings that impact all users or the entire system are referred to as global. So, user profiles that affect all users are known as global user profiles. These files allow administrators to set system wide settings that will be the default setting for all users. Through their individual profiles, users can change or edit these settings to customize their environment.

We are going to take a look at these global profiles, what files are involved, and what they provide for the users.

A Quick Sidebar

When we are discussing scripts in Linux there are two methods of using a script.

Executing (or Running the Script)

This is similar to executing/running any compiled program in Linux - like executing the file to open up a web browser or a word processor.

...BUT...

Executing a script will run each of the commands in the script file in a **new** shell process, which means it will be a new process id with a new environment.

To clarify this:

```
./somescript
```

will **execute** somescript provided that this script file is executable and located in the current directory. We know it is looking for the file in the current directory since the command specifically includes the leading ./ - the . specifies this directory. Specifying in this fashion is necessary because the current directory is usually not (and usually should not be) in the user's $PATH .

Likewise, the command:

```
somescript
```

will **execute** somescript if the file is executable and located in any of the directories listed in this user's $PATH .

Source a Script

Alternatively, **sourcing** a script will run each of the commands contained in the script file within the **current** shell process. So, the command:

```
source somescript
```

will **source** somescript . The file need not be executable but it must be a valid shell script. The file can be in current directory or in a directory in $PATH .

The command

```
. somescript
```

will also **source** somescript . Bash defines the command source as an alias to dot, so the above two commands are basically the same.

This page titled 02-E.16: System Wide User Profiles is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-E.16.1: System Wide User Profiles: /etc/profile

The /etc/profile File

The /etc/profile contains Linux system wide environment and other startup scripts. Usually the default command line prompt is set in this file. It is used for all users logging in to the bash, ksh, or sh shells. This is usually where the PATH variable, user limits, and other settings are defined for users. This file is only run for login shell and therefore does not run when a script is executed. Part of this file is checking for the existence of the /etc/profile.d directory (we will discuss more about this in a moment).

The administrator can make some minor changes in this file if they desire to customize the system. However, to make large changes or application specific changes, create a separate script in the /etc/profile.d directory.

Below is an example of the /etc/profile

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
if [ "$PS1" ]; then
   if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
   fi
fi


if [ -x /usr/bin/id ]; then
   USER="`id -un`"
   LOGNAME=$USER
   MAIL="/var/spool/mail/$USER"
fi

export USER LOGNAME MAIL

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
     . $i
    fi
done
```

The script:

- checks to see if the prompt variable, $PS1, is defined - if so then this is an interactive login
- checks for the existence of /etc/bash.bashrc - if it exists the system sources the file
- checks to see if the file /usr/bin/id is executable, if so then we get the userid, the loginname and point to the email file
- checks for existence of the /etc/profile.d directory, if it exists we source each file in that directory

"The Bash Shell Startup Files" by Multiple contributors, Beyond Linux® From Scratch (System V Edition) is licensed under CC BY-NC-SA 2.0

This page titled 02-E.16.1: System Wide User Profiles: /etc/profile is shared under a CC BY-NC-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-E.16.2: System Wide User Profiles: The /etc/profile.d Directory

/etc/profile.d

This directory contains files configuring system-wide behavior of specific programs and site-specific environment customization. As seen previously, these scripts are run from the /etc/profile script.

This would be where the administrator can place customization scripts of their own.

Depending on which distribution of Linux you are using, you will find a variety of files in this directory.

```
pbmac@pbmac-server $ ls -la /etc/profile.d
total 72
drwxr-xr-x   2 root root  4096 Jun  1 15:06 .
drwxr-xr-x 219 root root 16384 Jun  4 13:11 ..
-rw-r--r--   1 root root    96 Jun  1 15:06 01-locale-fix.sh
-rw-r--r--   1 root root    40 Apr 15  2014 appmenu-qt5.sh
-rw-r--r--   1 root root   824 Jun  1 15:06 apps-bin-path.sh
-rw-r--r--   1 root root   664 Jun  1 15:06 bash_completion.sh
-rw-r--r--   1 root root  1003 Jun  1 15:06 cedilla-portuguese.sh
-rw-r--r--   1 root root   813 Feb 13 07:30 flatpak.sh
-rw-r--r--   1 root root   652 Apr  3  2019 input-method-config.sh
-rw-r--r--   1 root root   385 Aug 27  2018 kaltura-base.sh
-rw-r--r--   1 root root   108 Dec 23  2017 sysstat.sh
-rw-r--r--   1 root root  1941 Jun  1 15:06 vte-2.91.sh
-rw-r--r--   1 root root  1947 Jun  1 14:18 vte.sh
-rw-r--r--   1 root root   954 Jun  1 15:06 xdg_dirs_desktop_session.sh
-rw-r--r--   1 root root  1557 Apr 14  2016 Z97-byobu.sh
```

"The Bash Shell Startup Files" by Multiple contributors, Beyond Linux® From Scratch (System V Edition) is licensed under CC BY-NC-SA 2.0

This page titled 02-E.16.2: System Wide User Profiles: The /etc/profile.d Directory is shared under a CC BY-NC-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

02-E.16.3: System Wide User Profiles: The /etc/bashrc File

The /etc/bashrc File

The /etc/bashrc is referred to as the /etc/bash.bashrc on some Linux distros. It contains system-wide functions and aliases including other configurations that apply to all system users.

If the system administrator allows users to select from the numerous different shells for their login shell, it is a good idea to put all bash-specific system configurations in this file.

This page titled 02-E.16.3: System Wide User Profiles: The /etc/bashrc File is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

CHAPTER OVERVIEW

Module 03: Permission and Ownership Management

- 03-A.1: Modify Permissions: File and Directory
- 03-A.2: Modify Permissions: File and Directory (continued)
- 03-B.3: Modify Ownership of Files and Directories
- 03-C.4: Special Permission Types - The setuid Bit
  - 03-C.4.1: Special Permission Types - The setgid Bit
  - 03-C.4.2: Special Permission Types - The Sticky Bit
- 03-C.5: File Attributes
- 03-D.6: Permission Issues and How to Troubleshoot

This page titled Module 03: Permission and Ownership Management is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

03-A.1: Modify Permissions: File and Directory

EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Module Introduction

As a Linux admin it will be necessary to set user and/or group permissions to control access to directories and files. This includes setting file and directory permissions, context-based permissions, and controlling privilege escalation.

Objectives of this Module

In this lesson, you will:

- Modify file and directory permissions.
- Modify file and directory ownership.
- Configure file and directory special permissions and attributes.
- Troubleshoot permissions and ownership problems.

This page titled 03-A.1: Modify Permissions: File and Directory is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

03-A.2: Modify Permissions: File and Directory (continued)

Changing File and Directory Permissions

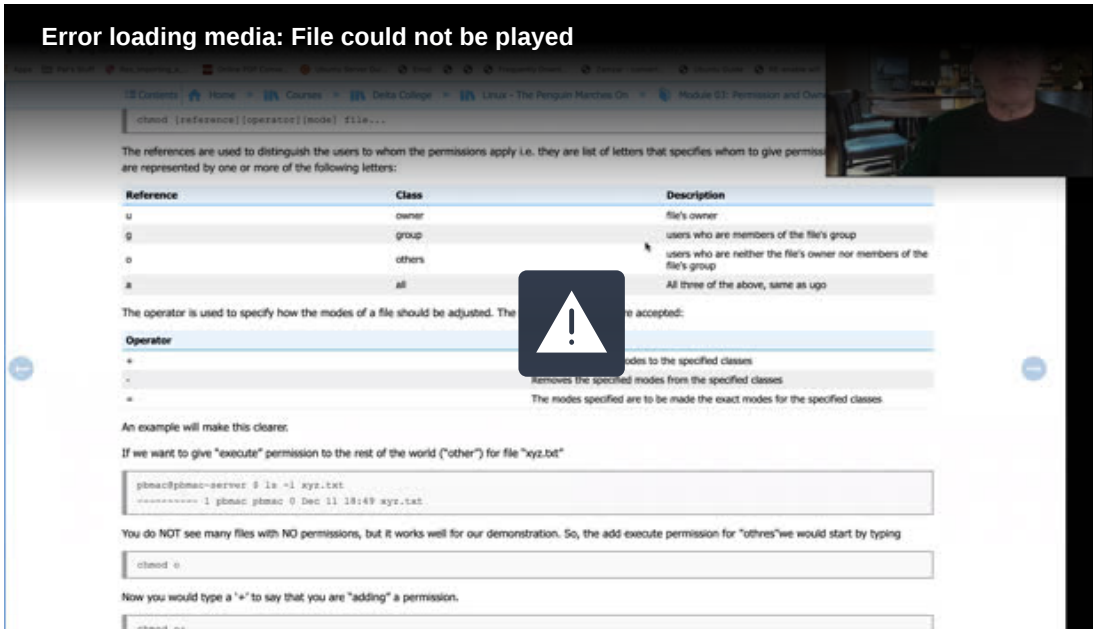
The **chmod** Command

The **chmod** command is used to change the access mode of a file. The name is an abbreviation of change mode.

Syntax :

```
chmod [reference][operator][mode] file...
```

This video discusses file permissions, how to set them, change them and understand them. The author inadvertently says "Unix", instead of "Linux" at the beginning of this video...we **are** talking about Linux



The references are used to distinguish the users to whom the permissions apply, i.e. they are a list of letters that specifies whom to give permissions. The references are represented by one or more of the following letters:

Reference	Class	Description
u	owner	file's owner
g	group	users who are members of the file's group
o	others	users who are neither the file's owner nor members of the file's group
a	all	All three of the above, same as ugo

The operator is used to specify how the modes of a file should be adjusted. The following operators are accepted:

Operator	Description
+	Adds the specified modes to the specified classes
-	Removes the specified modes from the specified classes
=	The modes specified are to be made the exact modes for the specified classes

An example will make this clearer.

If we want to give “execute” permission to the rest of the world (“other”) for file “xyz.txt”

```
pbmac@pbmac-server $ ls -l xyz.txt
---------- 1 pbmac pbmac 0 Dec 11 18:49 xyz.txt
```

You do NOT see many files with NO permissions, but it works well for our demonstration. So, the add execute permission for "others" we would start by typing:

```
chmod o
```

Now you would type a '+' to say that you are "adding" a permission.

```
chmod o+
```

Then you would type an 'x' to say that you are adding "execute" permission.

```
chmod o+x
```

Finally, specify which file you are changing:

```
chmod o+x xyz.txt
```

The change can be seen in the below - the xyz.txtfile now has execute permission - 'x' - on the "others" permission level.

```
pbmac@pbmac-server $ chmod o+x xyz.txt
pbmac@pbmac-server $ ls -l xyz.txt
---------x 1 pbmac pbmac 0 Dec 11 18:49 xyz.txt
```

It is possible to change multiple permissions at once. For example, to give ALL permissions for everyone, the command would be

```
chmod ugo+rwx xyz.txt
```

This command affects the user (u), the group (g) and others (o) and adds (-+) the read (r), the write (w) and the execute (x) permission to the file xyz.txt.

```
pbmac@pbmac-server $ chmod ugo+rwx xyz.txt
pbmac@pbmac-server $ ls -l xyz.txt
-rwxrwxrwx 1 pbmac pbmac 0 Dec 11 18:49 xyz.txt
```

NOTICE - it did NOT add any value to the first position; it is still a dash.

Another example - we will start with the file abc.mp4:

```
pbmac@pbmac-server $ ls -l abc.mp4
-rw-r--r-x 1 pbmac pbmac 0 Dec 11 18:57 abc.mp4
```

Using a more complex **`chmod`** command we change several permissions at different levels:

```
pbmac@pbmac-server $ chmod o+x abc.mp4
pbmac@pbmac-server $ ls -l abc.mp4
-rw-rw-r-x 1 pbmac pbmac 0 Dec 11 18:57 abc.mp4
```

This command adds (+) the read (r) and write (w) permission to both user (u) and group (g) and removes (-) the execute (x) permission from others (o) for the file abc.mp4 (this file does NOT exist in our images).

The command

```
chmod ug=rx,o+r abc.c
```

assigns read (r) and execute (x) permission to both user (u) and group (g) and adds read permission to others for the file abc.c.

```
pbmac@pbmac-server $ ls -l abc.c
-rw-r----- 1 pbmac pbmac 0 Dec 11 19:07 abc.c
pbmac@pbmac-server $ chmod ug=rx,o+r abc.c
pbmac@pbmac-server $ ls -l abc.c
-r-xr-xr-- 1 pbmac pbmac 0 Dec 11 19:07 abc.c
```

There can be numerous combinations of file permissions that can add, remove and assign.

The Octal Notations

Each of these digits is the sum of its component bits in the binary numeral system. As a result, specific bits add to the sum as it is represented by a numeral:

- The read bit adds 4 to its total (in binary 100),
- The write bit adds 2 to its total (in binary 010), and
- The execute bit adds 1 to its total (in binary 001).

These values never produce ambiguous combinations; each sum represents a specific set of permissions. More technically, this is an octal representation of a bit field – each bit references a separate permission, and grouping 3 bits at a time in octal corresponds to grouping these permissions by user, group, and others.

These are the examples from the symbolic notation section given in octal notation:

Symbolic notation	Numeric notation	English
`----------`	0000	no permissions
`-rwx------`	0700	**read, write, & execute only for owner**
`-rwxrwx---`	0770	read, write, & execute for owner and group
`-rwxrwxrwx`	0777	read, write, & execute for owner, group and others
`---x--x--x`	0111	execute
`--w--w--w-`	0222	write
`--wx-wx-wx`	0333	write & execute
`-r--r--r--`	0444	read
`-r-xr-xr-x`	0555	read & execute
`-rw-rw-rw-`	0666	read & write
`-rwxr-----`	0740	owner can read, write, & execute; group can only read; others have no permissions

Using the table above we can figure this out. The following two commands are the same.

```
chmod ugo+rwx [file_name]
chmod 777 [file_name]

So - this is:
421  421  421
rwx  rwx  rwx
 7    7    7
```

Both provide read, write and execute permission (the octal value of 7) to all three levels. The first number, on the left, represents the user, the middle number represents the group, and the last number, on the right, represents the others.

Another example:

```
chmod u=r,g=wx,o=rx [file_name]
chmod 435 [file_name]

Looking at this one:
 421   421   421
 r--   -wx   r-x
  4     3     5
```

Both commands give read (an octal value of 4) permission to user, write and execute (an octal value of 3) for group and read and execute (an octal value of 5) for others.

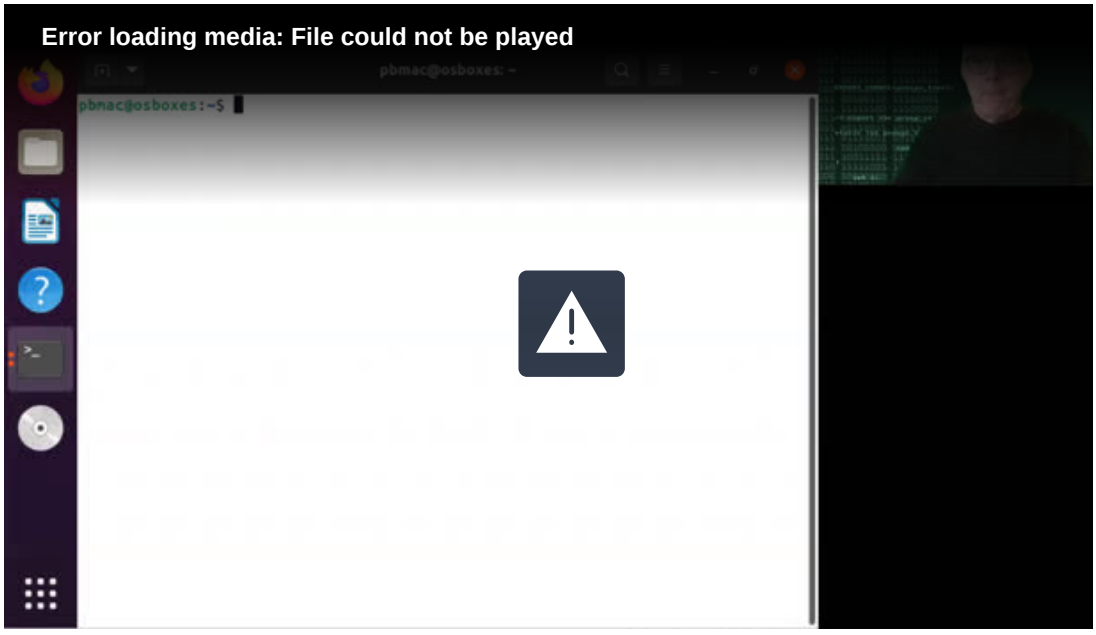
And even this…

```
chmod 775 [file_name]
chmod ug+rwx,o=rx [file_name]

421  421  421
rwx  rwx  r-x
 7    7    5
```

Both the commands give all permissions (an octal value of 7) to user and group, read and execute (an octal value of 5) for others.

A brief video about the concept of umask, and what it means to Linux.



Adapted from:
"File-system permissions" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

03-A.2: Modify Permissions: File and Directory (continued) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

03-B.3: Modify Ownership of Files and Directories

EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

File and Directory Ownership

Every file and directory on a Linux system is owned by someone, and the owner has complete control to change or delete the files they own. In addition to having an owning user, a file has an owning group. In the output below of the **`ls -l`** command, we can see these three directories are owned by the user santiago and by the group santiago.

```
santiago@pbmac-server:~$ ls -l
drwxrwxr-x 2 santiago santiago  4096 Jun  4 14:18 Desktop
drwxrwxr-x 2 santiago santiago  4096 Jun  4 14:18 Documents
drwxrwxr-x 2 santiago santiago  4096 Jun  4 14:18 Downloads
-rw-r--r-- 1 santiago santiago  8980 Apr 18 16:30 examples.desktop
```

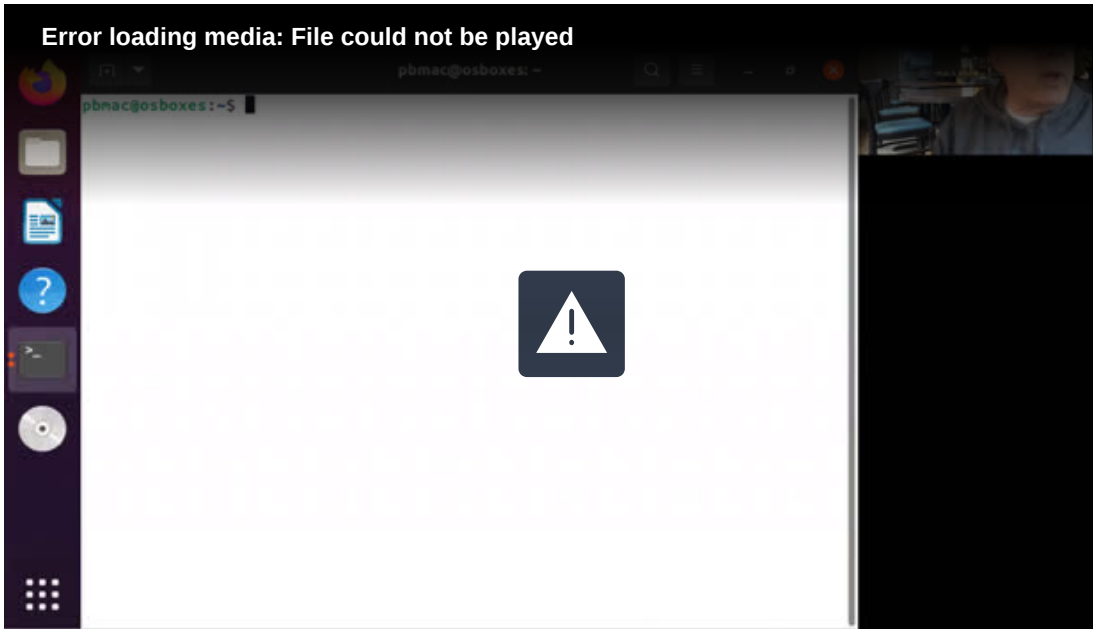
The ownership settings are set by the **`chmod`** command and controls who is allowed to perform read, write, or execute actions on a file or directory. Ownership can be changed with the **`chown`** command.

It is often necessary to change ownership. Files and directories can live a long time on a system, but users can come and go. Ownership may also need to change when files and directories are moved around the system or from one system to another.

If there is a group of users that work on personnel files, and the company hires a new employee, we need to make sure that this employee can view all the necessary personnel files.

```
pbmac@pbmac-server $ ls -la
drwxr-xr-x 2 pbmac pbmac  4096 Jun  4 17:58 .
drwxrwxrwx 3 pbmac pbmac 45056 Jun  4 17:58 ..
-rw-r--r-- 1 pbmac pbmac     0 Jun  4 17:51 PersonnelFile-1.txt
```

File ownership is an important concept in Linux. The following video goes into more depth on this topic.



The **chown** Command

The **`chown`** command is used to change the file owner or group.

Syntax:

```
chown [ OPTION ] [OWNER][:[GROUP]] FILE…
chown [ OPTION ] –reference=RFILE FILE…
```

Command Options:

Options	Option Meaning
-f, --silent, --quiet	suppress most error messages
--reference=RFILE (RFILE is simply some other file on the system)	use RFILE's owner and group rather than specifying OWNER:GROUP values
-R, --recursive	operate on files and directories recursively

Notice the PersonnelFile-1.txt is owned by user pbmac with the group pbmac. Unless a user can login as pbmac, or is in the group pbmac, they can do nothing other than read this file (notice the 'r' permissions for group and others). In this example the new employee is going to get ownership of both user and group of this file.

```
pbmac@pbmac-server $ chown bob:personnel PersonnelFIle-1.txt
pbmac@pbmac-server $ ls -l
-rw-r--r-- 1 bob    personnel     0 Jun  4 17:51 PersonnelFile-1.txt
```

With the **chown** command we can change both the owner, bob, and the group, personnel, with a single command by simply separating the user and group with a colon. If for some reason we want to change back to user pbmac and group pbmac there is a shortcut:

```
pbmac@pbmac-server $ chown pbmac: PersonnelFIle-1.txt
pbmac@pbmac-server $ ls -l
-rw-r--r-- 1 pbmac pbmac     0 Jun  4 17:51 PersonnelFile-1.txt
```

We do not have to change the owner - to change just the group put a colon in front of the group. Now members of the **personnel** group can edit the file
(we also added write permissions to the file for the group):

```
pbmac@pbmac-server $ chmod g+x PersonnelFIle-1.txt
pbmac@pbmac-server $ chown :personnel PersonnelFIle-1.txt
pbmac@pbmac-server $ ls -l
-rw-r--r-- 1 pbmac    personnel     0 Jun  4 17:51 PersonnelFile-1.txt
```

A few additional arguments to **chown** can be useful at both the command line and in a script. Just like many other Linux commands, **chown** has a recursive argument (**-R**) which tells the command to descend into the directory to operate on all files inside. Without the **-R** flag, only the permissions of the folder are changed, leaving the files inside it unchanged. In this example, assume that the intent is to change permissions of a directory and all its contents. Here I have added the **-v** (verbose) argument so that **chown** reports what it is doing:

```
pbmac@pbmac-server $ ls -l . Personnel/
.:
total 4
drwxr-xr-x 2 pbmac pbmac 4096 Jun  4 17:58 Personnel

Personnel/:
total 0
-rw-r--r-- 1 pbmac pbmac 0 Jun  4 17:51 PersonnelFile-1.txt

pbmac@pbmac-server $ chown -vR bob:personnel Personnel/
```

```
changed ownership of 'Personnel/PersonnelFile-1.txtl' from pbmac:pbmac to  bob:personnel
changed ownership of 'Personnel' from rom pbmac:pbmac to  bob:personnel
```

Depending on your role, you may need to use **sudo** to change ownership of a file.

The `chgrp` Command

There is also a **`chgrp`** command which is used to change the group ownership of a file.

Syntax:

```
chgrp [ OPTION ] GROUP FILE…
chgrp [ OPTION ] –reference=RFILE FILE…
```

Command Options:

Options	Options Meaning
-f, --silent, --quiet	suppress most error messages
--reference=RFILE (RFILE is simply some other file on the system)	use RFILE's owner and group rather than specifying OWNER:GROUP values
-R, --recursive	operate on files and directories recursively

If our PersonnelFile-1.txtfile has both owner and group of pbmac, and we desire to change the group to personnel we can do that.

```
pbmac@pbmac-server $ ls -l
-rw-r--r-- 1 pbmac pbmac 0 Jun  4 17:51 PersonnelFile-1.txt
pbmac@pbmac-server $ chgrp personnel PersonnelFile-1.txt
pbmac@pbmac-server $ ls -l
-rw-r--r-- 1 pbmac personnel     0 Jun  4 17:51 PersonnelFile-1.txt
```

Adapted from:
"Introduction to the Linux chown command" by Alan Formy-Duval, opensource.com is licensed under CC BY-SA 4.0
"chown command in Linux with Examples" by Madhusudan_Soni, Geeks for Geeks is licensed under CC BY-SA 4.0
"chgrp command in Linux with Examples" by Kaustav kumar Chanda, Geeks for Geeks is licensed under CC BY-SA 4.0

03-B.3: Modify Ownership of Files and Directories is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

03-C.4: Special Permission Types - The setuid Bit

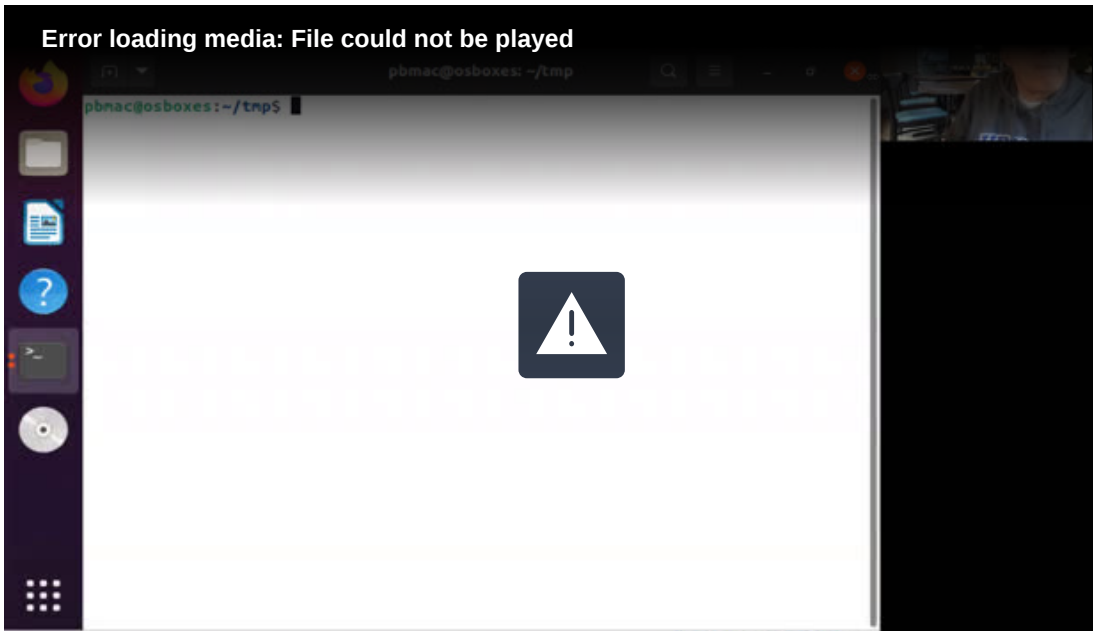
EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Linux Special Permissions

Linux uses some special permissions to allow certain capabilities that go beyond the basic rwx. By using special permissions, users with less privilege are allowed to execute a file and assume the privileges of the file's owner or group. This enables the user to temporarily have higher level privileges in a very limited and controlled manner.

Special permissions is a new topic for many Linux users. This video goes into detail about what these permissions are, how to set modify and remove them from a file's permissions.



The setuid Bit

The set user id bit (**`setuid`**) is present for files which have executable permissions. The **`setuid`** bit simply indicates that when running the executable, it will set its permissions to that of the owner, instead of setting it to the user who launched it. Similarly, there is a **`setgid`** bit which does the same for the gid, which we will touch on in a moment.

To locate the **`setuid`**, look for an ‘s’ instead of an ‘x’ in the executable bit of the file permissions.

The following are a few of the Linux commands that use the SUID bit to give the command elevated privileges when run by a regular user:

```
pbmac@pbmac-server $ ls -l /bin/su
-rwsr-xr-x 1 root root 44664 Mar 22  2019 /bin/su
pbmac@pbmac-server $ ls -l /bin/ping
-rwsr-xr-x 1 root root 64424 Jun 28  2019 /bin/ping
pbmac@pbmac-server $ ls -l /bin/mount
-rwsr-xr-x 1 root root 43088 Jan  8 10:31 /bin/mount
pbmac@pbmac-server $ ls -l /bin/umount
-rwsr-xr-x 1 root root 26696 Jan  8 10:31 /bin/umount
pbmac@pbmac-server $ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59640 Mar 22  2019 /usr/bin/passwd

You can locate ALL setuid files with:
pbmac@pbmac-server $ sudo find / -perm -4000
```

You can see that for each of these executable files, the 'x' has been replaced with an 's' for the owner's (root) permission. When any user executes one of these files that process is given the permissions of the owner - in these cases it is root. So, if my user, pbmac, were to execute the **`/usr/bin/passwd`** command I temporarily have root privileges, which is what allows me to actually alter the /etc/passwd file.

To set the setuid bit, use the **`chmod`** command as shown below - where filesname is the name of the file that you desire to add the setuid bit:

```
pbmac@pbmac-server $ chmod u+s filename
```

To remove the setuid bit, use the following command:

```
pbmac@pbmac-server $ chmod u-s filename
```

The setuid bit works for files owned by any user, not just root.

Adapted from: "SetUID, SetGID, and Sticky Bits in Linux File Permissions" by Anannya Uberoi 1, Geeks for Geeks is licensed under CC BY-SA 4.0

03-C.4: Special Permission Types - The setuid Bit is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

03-C.4.1: Special Permission Types - The setgid Bit

The setgid Bit

The setgid affects both files as well as directories. When used on a file, it executes with the privileges of the group of the user who owns it instead of executing with those of the group of the user who executed it.

When the bit is set for a directory, the set of files in that directory will have the same group as the group of the parent directory, and not that of the user who created those files. This is used for file sharing since they can now be modified by all the users who are part of the group of the parent directory.

To locate the setgid bit, look for an ‘s’ in the group section of the file permissions. We show a **`find`** command that will search your entire system looking for setgid files, and the output for the **`wall`** command.

```
pbmac@pbmac-server $ sudo find / -perm -2000

-rwxr-sr-x 1 root tty 30800 Jan  8 10:31 /usr/bin/wall
```

To set the setgid bit, use the following command - where filesname is the name of the file that you desire to set or remove the setgid bit:

```
pbmac@pbmac-server $ chmod g+s filename
```

To remove the setgid bit, use the following command:

```
pbmac@pbmac-server $ chmod g-s filename
```

Security Risks

The setuid bit is indeed quite useful in various applications, however, the executable programs supporting this feature should be carefully designed so as to not compromise on any security risks that follow, such as buffer overruns and path injection. If a vulnerable program runs with root privileges, the attacker could gain root access to the system through it. To dodge such possibilities, some operating systems ignore the setuid bit for executable shell scripts.

Adapted from: "SetUID, SetGID, and Sticky Bits in Linux File Permissions" by Anannya Uberoi 1, Geeks for Geeks is licensed under CC BY-SA 4.0

03-C.4.1: Special Permission Types - The setgid Bit is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

03-C.4.2: Special Permission Types - The Sticky Bit

The Sticky Bit

The sticky bit was initially introduced to 'stick' an executable program's text segment in the swap space even after the program has completed execution, in order to speed up the subsequent runs of the same program. However, these days the sticky bit means something entirely different.

When a directory has the sticky bit set, its files can be deleted or renamed only by the file owner, directory owner and the root user. The command below shows how the sticky bit can be set.

```
chmod +t filename
```

Simply look for a 't' character in the file permissions to locate the sticky bit. The snippet below shows how we can set the sticky bit for some directory "Gatos," and how it prevents the new user from deleting a file in the directory.

Here is a directory called Personnel. It has a permission of 777 (**rwxrwxrwx**)- the user, the group and all the rest of the world have read, write and execute permissions. The directory has 2 files in it. The xyz.txt file also has wide open permissions.

```
pbmac@pbmac-server $ ls -l
drwxrwxrwx 2 pbmac pbmac 4096 Jun  5 16:23 Personnel
pbmac@pbmac-server $ ls -l Personnel/
-rw-r--r-- 1 pbmac pbmac 0 Jun  4 17:51 PersonnelFile-1.txt
-rwxrwxrwx 1 pbmac pbmac 0 Jun  5 16:23 xyz.txt
```

Use the **chmod** command to add the sticky bit, and an ls -l confirms the sticky bit is now set. Switch users to santiago, and cd into the Personnel directory, and attempt to remove the xyz.txt file - it fails due to the sticky bit being set on the directory where this file lives. Exit from the user santiago.

```
pbmac@pbmac-server $ chmod +t Personnel/
pbmac@pbmac-server $ su santiago
Password:
santiago@pbmac-server $ cd Personnel/
santiago@pbmac-server $ rm xyz.txt
rm: cannot remove 'xyz.txt': Operation not permitted
santiago@pbmac-server $
```

As user pbmac, the sticky bit is removed using the **chmod** command. Switch users to santiago, and **cd** into the Personnel directory, and attempt to remove the xyz.txt file - success, because the directory has **rwxrwxrwx** as does the file.

```
pbmac@pbmac-server $ chmod -t Personnel/
pbmac@pbmac-server $ ls -l
drwxrwxrwx 2 pbmac pbmac 4096 Jun  5 16:23 Personnel
pbmac@pbmac-server $ su santiago
Password:
santiago@pbmac-server $ cd Personnel/
santiago@pbmac-server $ rm xyz.txt
santiago@pbmac-server $ ls
PersonnelFile-1.txt
```

As shown above, to remove the sticky bit, simply use the following command:

```
chmod -t filename
```

Since deleting a file is controlled by the write permission of the file, practical uses of the sticky bit involve world-writable directories such as '/tmp' so that the delete permissions are reserved only for the owners of the file.

Note that each of these bits can also be set in octal format with SUID=4, SGID=2, and Sticky=1.

```
Set the SETUID bit
pbmac@pbmac-server $ chmod 4744 filename
Set the SGID bit
pbmac@pbmac-server $ chmod 2644 filename
Set the Sticky bit
pbmac@pbmac-server $ chmod 1755 filename
```

Uppercase or Lowercase?

If you are setting the special bits and see an uppercase **S** or **T** instead of lowercase (as we've seen until this point), it is because the underlying execute bit is not present. To demonstrate, the following example creates a file with the sticky bit set. We can then add/remove the execute bit to demonstrate the case change.

```
pbmac@pbmac-server $ chmod 1755 cap-ST-demo
pbmac@pbmac-server $ ls -l cap-ST-demo
-rwxr-xr-t  1 root root 0 Jun 12 06:16 cap-ST-demo
Now - remove the execute permission from the other and see what happens
pbmac@pbmac-server $ chmod o-x cap-X-demo
pbmac@pbmac-server $ ls -l cap-X-demo
-rwxr-xr-T  1 root root 0 Jun 12 06:16 cap-ST-demo
```

Setting the Execute Bit Conditionally

To this point, we've set the **execute** bit using a lowercase **x**, which sets it without asking any questions. We have another option: using an uppercase **X** instead of lowercase will set the **execute** bit only if it is already present somewhere in the permission group. This can be a difficult concept to explain, but the demo below will help illustrate it. Notice here that after trying to add the **execute** bit to the group privileges, it is not applied.

```
pbmac@pbmac-server $ ls -l cap-X-file
-rw-r--r-- 1 root root 0 Jun 12 06:31 cap-X-file
pbmac@pbmac-server $ chmod g+X cap-X-file
Notice - the file does NOT accept the addition of the X permission
pbmac@pbmac-server $ ls -l cap-X-file
-rw-r--r-- 1 root root 0 Jun 12 06:31 cap-X-file
```

In this similar example, we add the execute bit first to the group permissions using the lowercase **x** and then use the uppercase **X** to add permissions for all other users. This time, the uppercase **X** sets the permissions.

```
pbmac@pbmac-server $ ls -l cap-X-file
-rw-r--r-- 1 root root 0 Jun 12 06:31 cap-X-file
Add x permission to the group
```

```
pbmac@pbmac-server  chmod g+x cap-X-file
pbmac@pbmac-server $ ls -l cap-X-file
-rw-r-xr-- 1 root root 0 Jun 12 06:31 cap-X-file
Now try with the upper case X
pbmac@pbmac-server $ chmod o+X cap-X-file
pbmac@pbmac-server $ ls -l cap-X-file
-rw-r-xr-x 1 root root 0 Jun 12 06:31 cap-X-file
```

Adapted from:

"Linux permissions 101" by Alex Juarez , OpenSource.com is licensed under CC BY-SA 4.0

"SetUID, SetGID, and Sticky Bits in Linux File Permissions" by Anannya Uberoi 1, Geeks for Geeks is licensed under CC BY-SA 4.0

03-C.4.2: Special Permission Types - The Sticky Bit is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

03-C.5: File Attributes

EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

File Attributes

Apart from the file mode bits that control **user and group** read, write and execute permissions, several **file systems** support file attributes that enable further customization of allowable file operations. This section describes some of these attributes and how to work with them.

The `chattr` Command

The **`chattr`** command in Linux is a file system command which is used for changing the attributes of a file in a directory. The primary use of this command is to make several files unable to alter for users other than the superuser.

Synopsis:

```
chattr [ OPTIONS ] [ -v version ] [ mode ] files...
```

At the beginning of a mode string, one of the following operators must appear:

- '+' : Adds selected attributes to the existing attributes of the files.
- '–' : Causes selected attributes to be removed.
- '=' : Causes selected attributes to be the only attributes that the files have.

The format of symbolic mode is:

```
{+|-|=}[aAcCdDeijsStTu]
```

Command Options:

Options	Option Meaning
-R	It is used to display the list attributes of directories and their contents recursively.
-a	Used to list all the files of a directory which also includes those whose name starts with a Period('.').
-d	This option will list the directories as regular files instead of listing their contents.

File Attributes:

There is a detailed explanation of these attributes on the **`chattr`** manual page, but this is a brief listing of Linux file attributes.

File Attribute	Attribute Description
a - append only	This attribute allows a file to be added to, but not to be removed. It prevents accidental or malicious changes to files that record data, such as log files.
c - compressed	This causes the kernel to compress data written to the file automatically and uncompress it when it's read back.
d - no dump	It makes sure the file is not backed up in backups where the dump utility is used.
e - extent format	It indicates that the file is using extents for mapping the blocks on disk.
i - immutable	It makes a file immutable, which goes a step beyond simply disabling write access to the file. The file can't be deleted, links to it can't be created, and the file can't be renamed.
j - data journaling	It ensures that on an Ext3 file system the file is first written to the journal and only after that to the data blocks on the hard disk.
s - secure deletion	It makes sure that recovery of a file is not possible after it has been deleted.
t - no tail-merging	Tail-merging is a process in which small data pieces at a file's end that don't fill a complete block are merged with similar pieces of data from other files.
u - undeletable	When a file is deleted its contents are saved, which allows a utility to be developed that works with that information to salvage deleted files.
A - no atime updates	Linux won't update the access time stamp when you access a file.
D - synchronous directory updates	It makes sure that changes to files are written to disk immediately, and not to cache first.
S - synchronous updates	The changes on a file are written synchronously on the disk.
T - and top of directory hierarchy	A directory will be deemed to be the top of directory hierarchies for the purposes of the Orlov block allocator.

Users can issue the **`chattr`** command to change a file's attributes on a Linux filesystem. A user can set or remove an attribute from the command line:

```
The file has the 'extent format' attribute set (see the list above)
pbmac@pbmac-server $ lsattr PersonnelFile-1.txt
--------------e--- PersonnelFile-1.txt

Only root can set the immutable flag - The command here made the file named file.txt i
are possible on this file until the attributes of the file are changed again.
pbmac@pbmac-server $ chattr +i PersonnelFile-1.txt
chattr: Operation not permitted while setting flags on PersonnelFile-1.txt

Only root can set an immutable flag
pbmac@pbmac-server $ sudo chattr +i PersonnelFile-1.txt
```

```
[sudo] password for pbmac:

Now, we see the immutable attribute set with 'chattr +i filename'
pbmac@pbmac-server $ lsattr PersonnelFile-1.txt
----i---------e--- PersonnelFile-1.txt

Remove the immutable attribute
pbmac@pbmac-server $ sudo chattr -i PersonnelFile-1.txt
pbmac@pbmac-server $ lsattr PersonnelFile-1.txt
--------------e--- PersonnelFile-1.txt
```

NOTE: The concept of immutable is important. If a file or directory has the immutable attribute set NO ONE - no, not even root - can change that file/directory. It is possible to mix mutable and immutable files and directories. A mutable directory MAY have immutable files in it - but new files or sub-directories may not be created in an immutable directory.

The `lsattr` Command

The **`lsattr`** command lists the file attributes on a second extended file system. See chattr explanation above for a description of the command options, the file attributes and what they mean.

Access Control Lists(ACL) in Linux

What is an ACL ?

Access Control List (ACL) provides an additional, more flexible permission mechanism for file systems. ACLs are often used for situations where the traditional file permission concept does not quite offer enough options. ACLs enable you to assign permissions to individual users or groups even when the user or group does not correspond to the Linux owner or group.

Use of ACL:

Think of a scenario in which a particular user is not a member of a group created by you, but you still want to give some read or write access. How can you do it without making that user a member of a group? ACLs will help us make this happen. Linux gives us two commands to administer ACLs: the **`setfacl`** command to set ACLs, and the **`getfacl`** command, to get the current settings for a file or directory.

The `setfacl` Command

The **`setfacl`** command is used to set Access Control Lists (ACLs) of files and directories. On the command line, a sequence of commands is followed by a sequence of files.

Syntax: (the syntax can be confusing)

```
setfacl [ OPTIONS ] [{-m|-x} acl_spec] [{-M|-X} acl_file] file
```

Command Options:

Options	Option Meaning
-b, --remove-all	Remove all extended ACL entries. The base ACL entries of the owner, group and others are retained.
-m	Modify the existing ACL of an object.
-R, --recursive	Apply operations to all files and directories recursively. This option cannot be mixed with "--restore."
-x	Remove entries from an existing ACL.
-v, --version	Print the version of setfacl, and exit.

The `getfacl` Command

The **`getfacl`** command displays the file name, owner, the group, and the Access Control List (ACL). If a directory has a default ACL, **`getfacl`** also displays the default ACL. Non-directories cannot have default ACLs.

Syntax:

```
getfacl [-aceEsRLPtpndvh] file
```

Command Options:

https://eng.libretexts.org/@go/page/26829

Options	Option Meaning
-a, --access	Display the file access control list.
-d, --default	Display the default access control list.
-c, --omit-header	Do not display the comment header (the first three lines of each file's output).
-e, --all-effective	Print all effective rights comments, even if identical to the rights defined by the ACL entry.
-E, --no-effective	Do not print effective rights comments.
-s, --skip-base	Skip files that only have the base ACL entries (owner, group, others).
-R, --recursive	List the ACLs of all files and directories recursively.
-L, --logical	Logical walk, follow symbolic links to directories. The default behavior is to follow symbolic link arguments, and skip symbolic links encountered in subdirectories. Only effective in combination with -R.
-P, --physical	Physical walk, do not follow symbolic links to directories. This also skips symbolic link arguments. Only effective in combination with -R.
-t, --tabular	Use an alternative tabular output format. The ACL and the default ACL are displayed side by side. Permissions that are ineffective due to the ACL mask entry are displayed capitalized. The entry tag names for the ACL_USER_OBJ and ACL_GROUP_OBJ entries are also displayed in capital letters, which helps in spotting those entries.
-p, --absolute-names	Do not strip leading slash characters ('/'). The default behavior is to strip leading slash characters.
-n, --numeric	List numeric user and group IDs

The **`getfacl`** command also has several options, and you can view them in the **`getfacl`** man page. Probably the most useful is the -R, which is the recursive option that walks through the specified directory looking at all files and subdirectories.

Below is a short example of using **`setfacl`** and **`getfacl`**:

```
pbmac@pbmac-server $ ls -l PersonnelFile-1.txt
-rw-r--r-- 1 pbmac pbmac 0 Jun  4 17:51 PersonnelFile-1.txt
```

```
getfacl shows the normal Linux permission settings shown in the line above
pbmac@pbmac-server $ getfacl PersonnelFile-1.txt
# file: PersonnelFile-1.txt
# owner: pbmac
# group: pbmac
user::rw-
group::r--
other::r--

Allow user santiago to have user lever rwx permissions on the specified file
pbmac@pbmac-server $ setfacl -m u:santiago:rwx PersonnelFile-1.txt
pbmac@pbmac-server $ ls -l PersonnelFile-1.txt
-rw-rwxr--+ 1 pbmac pbmac 0 Jun  4 17:51 PersonnelFile-1.txt
Notice the output on the line above - there is a '+' after the permissions signifying
getfacl shows that santiago does indeed have user priveleges.
pbmac@pbmac-server $ getfacl PersonnelFile-1.txt
# file: PersonnelFile-1.txt
# owner: pbmac
# group: pbmac
user::rw-
user:santiago:rwx
group::r--
mask::rwx
other::r--

We can remove a specific ACL entry with the -X argument
pbmac@pbmac-server $ setfacl -x u:santiago PersonnelFile-1.txt
pbmac@pbmac-server $ ls -l PersonnelFile-1.txt
-rw-r--r--+ 1 pbmac pbmac 0 Jun  4 17:51 PersonnelFile-1.txt
Notice in the line above that even though we removed the entry the '+' is still preser
To remove the ACL altogether we need to use the -b option, notice on the last line - t
pbmac@pbmac-server $ setfacl -b PersonnelFile-1.txt
pbmac@pbmac-server $ ls -l PersonnelFile-1.txt
-rw-r--r-- 1 pbmac pbmac 0 Jun  4 17:51 PersonnelFile-1.txt
```

Adapted from:
"chattr command in Linux with examples" by atharvakango, Geeks for Geeks is licensed under CC BY-SA 4.0
"Access Control Lists(ACL) in Linux" by msdeep14, Geeks for Geeks is licensed under CC BY-SA 4.0

03-C.5: File Attributes is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

03-D.6: Permission Issues and How to Troubleshoot

EXAM OBJECTIVES COVERED

4.3 Given a scenario, analyze and troubleshoot user issues.

Troubleshooting

Troubleshooting is a form of problem solving often applied to repair failed products or processes on a machine or a system. It is a logical, systematic search for the source of a problem in order to solve it and make the product or process operational again. Troubleshooting is needed to identify the symptoms. Determining the most likely cause is a process of elimination—eliminating potential causes of a problem. Finally, troubleshooting requires confirmation that the solution restores the product or process to its working state. In general, troubleshooting is the identification or diagnosis of "trouble" in the management flow of a system caused by a failure of some kind. The problem is initially described as symptoms of malfunction, and troubleshooting is the process of determining and remedying the causes of these symptoms.

There is a very handy cheat sheet, created by Setyh Kenlon, that is available by following this link. It is a pdf file which you can download and keep handy as you hone your Linux skills.

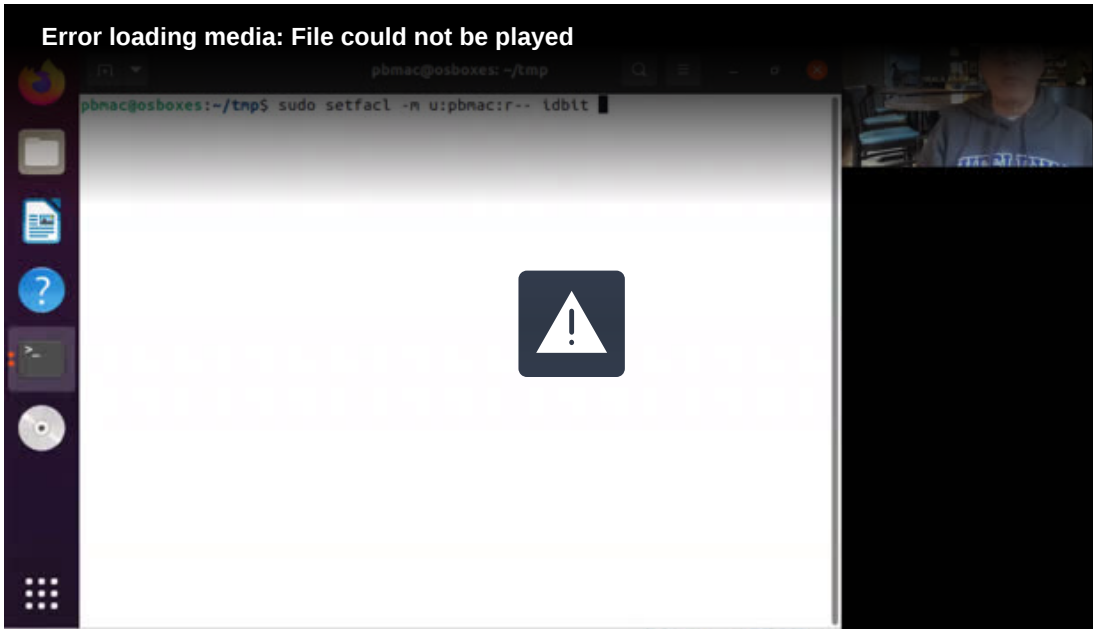
There are numerous web sites and "cheat sheets" that you can find to help you resolve problems with permissions. Although, in the previous several pages you have been introduced to the simple concepts...there is so much more that you will learn as you gain experience.

There are some basic things to remember about permissions:

- Always look at the existing permissions - and make sure that the reported issue would agree with these permissions. If need be you can used **`chmod`** to alter the permissions.
- Look to see if the SUID or SGID bit is set. Is it a lowercase or uppercase - that will tell you whether the underlying permission is set.
- Is the sticky bit set? This will change certain user's ability to remove the file.
- Check to see if an ACL is enabled on the file, since these controls do not show up in the normal permissions.
- **REMEMBER** -Directory Permissions: Read, write and execute permissions are set for directories as well as files. Read permission means that the user may see the contents of a directory (e.g. use the **`ls`** command for this directory.) Write permission means that a user may create files in the directory. Execute permission means that the user may enter the directory (i.e. make it his current directory.)
- Look at the parent directory as it may have some permission set that controls the files and sub-directories.

As you gain experience this will get easier.

Troubleshooting the setfacl and getfacl commands can be a bit daunting. This video covers the basics of those commands to help you get started in the proper direction.



Adapted from: "Troubleshooting" by Various Contributors, Wikipedia is licensed under CC BY-SA 4.0

03-D.6: Permission Issues and How to Troubleshoot is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 04: Managing Linux Storage

- 04-A.1: Storage Devices
- 04-A.2: Linux File Systems
- 04-A.3: Inodes
- 04-A.4 Journaling File Systems
- 04-A.5: Linux Virtual File System
- 04-A.6: Partitions and Labels
  - 04-A.6.1: Partitions and Labels continued
- 04-A.7: Disk Setup Process - fdisk
  - 04-A.7.1: Disk Setup Process - parted
  - 04-A.7.2: Disk Setup Process - mkfs & mount
- 04-A.8: The dev Directory
- 04-B.9: Logical Volume Management
  - 04-B.9.1: LVM and other stuff
  - 04-B.9.2: LVM Tools
- 04-C.10: Mounting Linux File Systems
- 04-D.11: File System Management
  - 04-D.11.1: File System Management - ext Filesystem Tools
  - 04-D.11.2: File System Management - xfs Tools
- 04-E.12: Linux Directory Structure
  - 04-E.12.1: Linux Directory Structure - Hierarchy
  - 04-E.12.2: Linux Directory Structure - Home Directory
  - 04-E.12.3: Linux Directory Structure - Paths
- 04-F.13: Storage Issues - Find and Fix
  - 04-F.13.1: Storage Issues - IO Scheduling
  - 04-F.13.2: Storage Issues - More IO Scheduling
  - 04-F.13.3: Storage Issues - iostat, ioping
  - 04-F.13.4: Storage Issues - Quota Management

This page titled Module 04: Managing Linux Storage is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

04-A.1: Storage Devices

EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.

2.3 Given a scenario, create, modify, and redirect files.

4.1 Given a scenario, analyze system properties and remediate accordingly.

Objectives of this Module

In this lesson, you will:

- Learn how to partition a drive in Linux.
- Learn to mount and manage Linux file systems.
- Identify the command line tools that can be used to perform disk setup in Linux.
- Differentiate between the purposes for journaling file systems and virtual file systems.

Storage in Linux

Like most other computing platforms, Linux is capable of working with most of the storage devices that are available on the market. The common types of storage include:

- hard disk drive (HDD) - they continue to hold greater quantities of date and are dropping in price;
- solid state devices (SSD) - a solid-state storage device that uses integrated circuit assemblies to store data persistently, typically using flash memory;
- USB flash drive - a data storage device that includes flash memory with an integrated USB interface;
- external storage devices - this includes the wide variety of optical devices - CD/DVD/Blu-ray as well as older magnetic tape;
- Secure Digital (SD) cards are a tiny flash memory card designed for high-capacity memory and various portable devices.

So, no worries about whether your Linux system can make use of any of the latest technology. In fact Linux is often quicker to pick up on newer technology than some of the commercial companies.

Linux: Block vs Character Devices

Character devices

Character special files or character devices provide unbuffered, direct access to the hardware device. They do not necessarily allow programs to read or write single characters at a time; that is up to the device in question. The character device for a hard disk, for example, will normally require that all reads and writes are aligned to block boundaries and most certainly will not allow reading a single byte.

Character devices are sometimes known as raw devices, to avoid the confusion surrounding the fact that a character device for a piece of block-based hardware will typically require programs to read and write aligned blocks.

Block devices

Block special files or block devices provide buffered access to hardware devices, and provide some abstraction from their specifics. [5] Unlike character devices, block devices will always allow the programmer to read or write a block of any size (including single characters/bytes) and any alignment. The downside is that because block devices are buffered, the programmer does not know how long it will take before written data is passed from the kernel's buffers to the actual device, or indeed in what order two separate writes will arrive at the physical device. Additionally, if the same hardware exposes both character and block devices, there is a risk of data corruption due to clients using the character device being unaware of changes made in the buffers of the block device.

Device Files

Device files are also known as device special files. Device files are employed to provide the operating system and users an interface to the devices that they represent. All Linux device files are located in the /dev directory, which is an integral part of the root (/) filesystem because these device files must be available to the operating system during the boot process.

One of the most important things to remember about these device files is that they are most definitely not device drivers. They are more accurately described as portals to the device drivers. Data is passed from an application or the operating system to the device

file which then passes it to the device driver which then sends it to the physical device. The reverse data path is also used, from the physical device through the device driver, the device file, and then to an application or another device.

Let's look at the data flow of a typical command to visualize this.

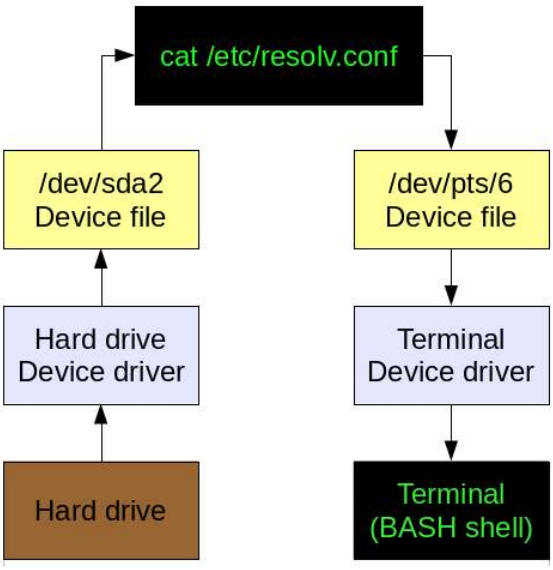


Figure **4.1.1**: Simple data flow for a typical command. ("Simple data flow for a typical command." by David Both is licensed under CC BY-SA 4.0)

In Figure 1, above, a simplified data flow is shown for a common command. Issuing the **`cat /etc/resolv.conf`** command from a GUI terminal emulator such as Konsole or xterm causes the resolv.conf file to be read from the disk with the disk device driver handling the device specific functions such as locating the file on the hard drive and reading it. The data is passed through the device file and then from the command to the device file and device driver for pseudo-terminal 6 where it is displayed in the terminal session.

Of course, the output of the **`cat`** command could have been redirected to a file in the following manner, **`cat /etc/resolv.conf > /etc/resolv.bak`** in order to create a backup of the file. In that case, the data flow on the left side of Figure 1 would remain the same while the data flow on the right would be through the /dev/sda2 device file, the hard drive device driver and then onto the hard drive itself.

These device files make it very easy to use standard streams (STD/IO) and redirection to access any and every device on a Linux or Unix computer. Simply directing a data stream to a device file sends the data to that device.

Classification

Device files can be classified in at least two ways. The first and most commonly used classification is that of the data stream commonly associated with the device. For example, tty (teletype) and serial devices are considered to be character based because the data stream is transferred and handled one character or byte at a time. Block type devices such as hard drives transfer data in blocks, typically a multiple of 256 bytes.

If you have not already, go ahead and as a non-root user in a terminal session, change the present working directory (PWD) to /dev and display a long listing. This shows a list of device files with their file permissions and their major and minor identification numbers. For example, the following device files are just a few of the ones in the /dev/directory on my Fedora 24 workstation. They represent disk and tty type devices. Notice the leftmost character of each line in the output. The ones that have a "b" are block type devices and the ones that begin with "c" are character devices.

```
brw-rw----   1 root disk        8,   0 Nov  7 07:06 sda
brw-rw----   1 root disk        8,   1 Nov  7 07:06 sda1
brw-rw----   1 root disk        8,  16 Nov  7 07:06 sdb
brw-rw----   1 root disk        8,  17 Nov  7 07:06 sdb1
```

```
brw-rw----   1 root disk        8,  18 Nov  7 07:06 sdb2
crw--w----   1 root tty         4,   0 Nov  7 07:06 tty0
crw--w----   1 root tty         4,   1 Nov  7 07:07 tty1
crw--w----   1 root tty         4,  10 Nov  7 07:06 tty10
crw--w----   1 root tty         4,  11 Nov  7 07:06 tty11
```

The more detailed and explicit way to identify device files is using the device major and minor numbers. The disk devices have a major number of 8 which designates them as SCSI block devices. Note that all PATA and SATA hard drives have been managed by the SCSI subsystem because the old ATA subsystem was many years ago deemed as not maintainable due to the poor quality of its code. As a result, hard drives that would previously have been designated as "hd[a-z]" are now referred to as "sd[a-z]."

You can probably infer the pattern of disk drive minor numbers in the small sample shown above. Minor numbers 0, 16, 32 and so on up through 240 are the whole disk numbers. So major/minor 8/16 represents the whole disk /dev/sdb and 8/17 is the device file for the first partition, /dev/sdb1. Numbers 8/34 would be /dev/sdc2.

The tty device files in the list above are numbered a bit more simply from tty0 through tty63.

Adapted from:
"Managing devices in Linux" by David Both , Opensource.com is licensed under CC BY-SA 4.0

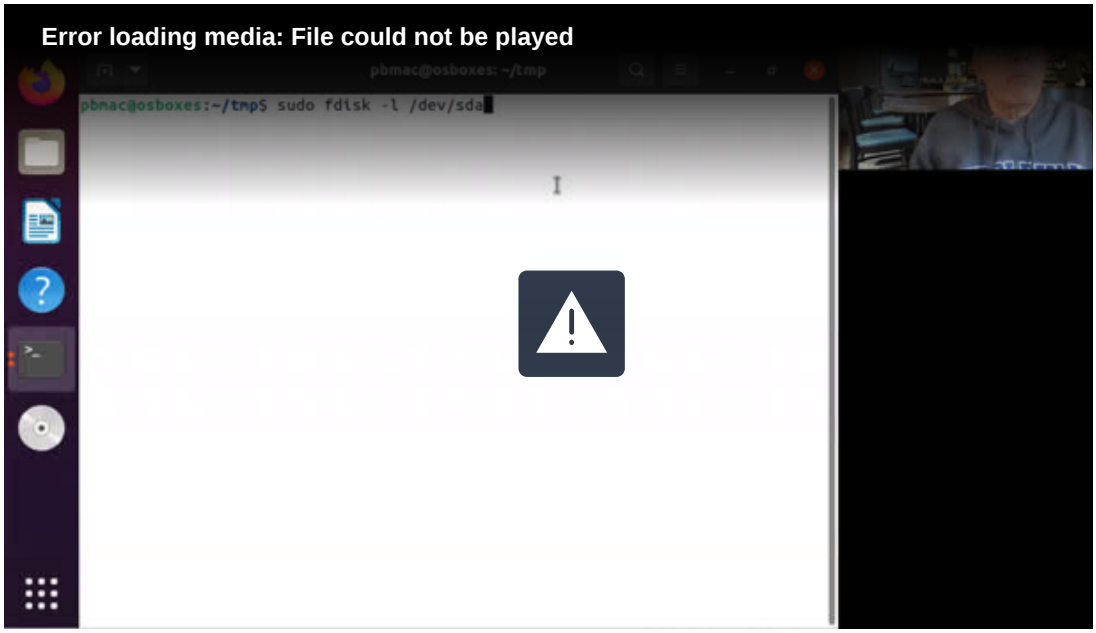
04-A.1: Storage Devices is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.2: Linux File Systems

What is a File System?

In computing, a file system or filesystem (often abbreviated to fs), controls how data is stored and retrieved. Without a file system, data placed in a storage medium would be one large body of data with no way to tell where one piece of data stops and the next begins. By separating the data into pieces and giving each piece a name, the data is easily isolated and identified. Taking its name from the way paper-based data management system is named, each group of data is called a "file." The structure and logic rules used to manage the groups of data and their names is called a "file system."

One of the important parts of file system managment is the creation of disk partitions. This video covers that topic with some examples for you to follow along.



There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications.

Below is a very brief comparison of the most common file systems in use with the Linux world.

File System	Max File Size	Max Partition Size	Journaling	Notes
Fat16	2 GiB	2 GiB	No	Legacy
Fat32	4 GiB	8 TiB	No	Legacy
NTFS	**2 TiB**	**256 TiB**	**Yes**	**(For Windows Compatibility) NTFS-3g can be installed on Linux, allowing read/write support.**
ext2	2 TiB	32 TiB	No	Legacy
ext3	2 TiB	32 TiB	Yes	Standard Linux filesystem for many years. Best choice for super-standard installation.
ext4	16 TiB	1 EiB	Yes	Modern iteration of ext3. Best choice for new installations where super-standard isn't necessary.
reiserFS	8 TiB	16 TiB	Yes	No longer well-maintained.
JFS	4PiB	32PiB	Yes (metadata)	Created by IBM - Not well maintained.
XFS	8 EiB	8 EiB	Yes (metadata)	Created by SGI. Best choice for a mix of stability and advanced journaling. Default for Centos / RHEL7 installations.

While we normally tend to think of the file system to be located on the system we are working on, with the proliferation of networking technology there are also technologies that allow us to use various network protocols that enable sharing data over a network. These capabilities are referred to as network file systems, and there are several that are supported by various Linux distros. The following table depicts the three most common protocols.

Network File System	Description
Server Message Block (SMB)	A network communication protocol for providing shared access to files, printers, and serial ports between nodes on a network. It also provides an authenticated inter-process communication mechanism. Most usage of SMB involves computers running Microsoft Windows, where it was known as "Microsoft Windows Network" before the introduction of Active Directory. Corresponding Windows services are LAN Manager Server for the server component, and LAN Manager Workstation for the client component.
Common Internet File System (CIFS)	A specific version of SMB designed by Microsoft as a successor to SMB version 1, but SMB versions 2 and 3 superseded it. However, The CIFS name still lingers in some of the Linux even though these tools actually support the newer versions of SMB.
Network File System	NFS functionality is very similar to SMB, however, the protocols are not compatible. When using Linux, NFS is preferred when you have Linux clients accessing Linux servers. When you are operating in an environment with a mix of Windows and Linux, the SMB protocol is the better choice.

Adapted from:
"LinuxFilesystemsExplained" by Anthony J Simon, CommunityHelpWiki is licensed under CC BY-SA 4.0

"Server Message Block" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

04-A.2: Linux File Systems is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.3: Inodes

Linux Inodes

A file system relies on data structures *about* the files, as opposed to the contents of that file. The former are called metadata—data that describes data. Each file is associated with an *inode*, which is identified by an integer, often referred to as an *i-number* or *inode number*.

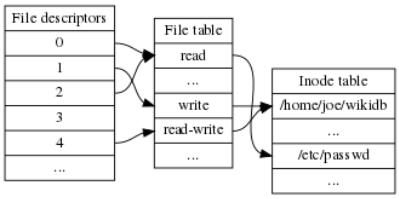


Figure **4.3.1**: File descriptors for a single process, file table and inode table ("File_table_and_inode_table.svg" by Qwertyus, Wikimedia is licensed under CC BY-SA 4.0)

Inodes store information about files and directories (folders). Each and every file under Linux (and UNIX) has the following attributes:

- File type (executable, block special etc)
- Permissions (read, write etc)
- Owner
- Group
- File Size
- File access, change and modification time (Remember UNIX or Linux never stores file creation time - this is a favorite question asked in UNIX/Linux sys admin job interviews.)
- File deletion time
- Number of links (soft/hard)
- Extended attribute such as append only or no one can delete file including root user (immutability)
- Access Control List (ACLs)

On many types of file system implementations, the maximum number of inodes is fixed at file system creation, limiting the maximum number of files the file system can hold. A typical allocation heuristic for inodes in a file system is one inode for every 2K bytes contained in the filesystem.

The inode number indexes a table of inodes in a known location on the device. From the inode number, the kernel's file system driver can access the inode contents, including the location of the file, thereby allowing access to the file. A file's inode number can be found using the **`ls -i`** command. The **`ls -i`** command prints the i-node number in the first column of the report.

File names and directory implications:

- Inodes do not contain its hardlink names, only other file metadata.
- Linux directories are lists of association structures, each of which contains one filename and one inode number.
- The file system driver must search a directory looking for a particular filename and then convert the filename to the correct corresponding inode number.

Adapted from:
"inode" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

04-A.3: Inodes is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.4 Journaling File Systems

Journaling

A journaling file system is more reliable when it comes to data storage. Journaling file systems do not necessarily prevent corruption, but they do prevent inconsistency and are much faster at file system checks than non-journaled file systems. If a power failure happens while you are saving a file, the save will not complete and you end up with corrupted data and an inconsistent file system. Instead of actually writing directly to the part of the disk where the file is stored, a journaling file system first writes it to another part of the hard drive and notes the necessary changes to a log. Then in the background it goes through each entry to the journal and begins to complete the task, and when the task is complete, it checks it off on the list. Thus the file system is always in a consistent state (the file got saved, the journal reports it as not completely saved, or the journal is inconsistent (but can be rebuilt from the file system)). Some journaling file systems can prevent corruption as well by writing data twice.

A journaling file system is a file system that keeps track of changes not yet committed to the file system's main part by recording the intentions of such changes in a data structure known as a "journal," which is usually a circular log. In the event of a system crash or power failure, such file systems can be brought back online more quickly with a lower likelihood of becoming corrupted.

Depending on the actual implementation, a journaling file system may only keep track of stored metadata, resulting in improved performance at the expense of increased possibility for data corruption. Alternatively, a journaling file system may track both stored data and related metadata, while some implementations allow selectable behavior in this regard.

Why Use Journaling

Updating file systems to reflect changes to files and directories usually requires many separate write operations. This makes it possible for an interruption (like a power failure or system crash) between writes to leave data structures in an invalid intermediate state.

For example, deleting a file on a Unix file system involves three steps:

1. Removing its directory entry.
2. Releasing the inode to the pool of free inodes.
3. Returning all disk blocks to the pool of free disk blocks.

If a crash occurs after step 1 and before step 2, there will be an orphaned inode and hence a storage leak; if a crash occurs between steps 2 and 3, then the blocks previously used by the file cannot be used for new files, effectively decreasing the storage capacity of the file system. Re-arranging the steps does not help, either. If step 3 preceded step 1, a crash between them could allow the file's blocks to be reused for a new file, meaning the partially deleted file would contain part of the contents of another file, and modifications to either file would show up in both. On the other hand, if step 2 preceded step 1, a crash between them would cause the file to be inaccessible, despite appearing to exist.

Detecting and recovering from such inconsistencies normally requires a complete walk of its data structures, for example by a tool such as fsck (the file system checker). This must typically be done before the file system is next mounted for read-write access. If the file system is large and if there is relatively little I/O bandwidth, this can take a long time and result in longer downtimes if it blocks the rest of the system from coming back online.

To prevent this, a journaled file system allocates a special area—the journal—in which it records the changes it will make ahead of time. After a crash, recovery simply involves reading the journal from the file system and replaying changes from this journal until the file system is consistent again. The changes are thus said to be atomic (not divisible) in that they either succeed (succeeded originally or are replayed completely during recovery), or are not replayed at all (are skipped because they had not yet been completely written to the journal before the crash occurred).

How is Journaling Accomplished

Some file systems allow the journal to grow, shrink and be re-allocated just as a regular file, while others put the journal in a contiguous area or a hidden file that is guaranteed not to move or change size while the file system is mounted. Some file systems may also allow external journals on a separate device, such as a solid-state drive or battery-backed non-volatile RAM. Changes to

the journal may themselves be journaled for additional redundancy, or the journal may be distributed across multiple physical volumes to protect against device failure.

The internal format of the journal must guard against crashes while the journal itself is being written to. Many journal implementations (such as the JBD2 layer in ext4) bracket every change logged with a checksum, on the understanding that a crash would leave a partially written change with a missing (or mismatched) checksum that can simply be ignored when replaying the journal at next remount.

Physical Journals

A physical journal logs an advance copy of every block that will later be written to the main file system. If there is a crash when the main file system is being written to, the write can simply be replayed to completion when the file system is next mounted. If there is a crash when the write is being logged to the journal, the partial write will have a missing or mismatched checksum and can be ignored at next mount.

Physical journals impose a significant performance penalty because every changed block must be committed twice to storage, but may be acceptable when absolute fault protection is required.

Logical Journals

A logical journal stores only changes to file metadata in the journal, and trades fault tolerance for substantially better write performance. A file system with a logical journal still recovers quickly after a crash, but may allow unjournaled file data and journaled metadata to fall out of sync with each other, causing data corruption.

For example, appending to a file may involve three separate writes to:

1. The file's inode, to note in the file's metadata that its size has increased.
2. The free space map, to mark out an allocation of space for the to-be-appended data.
3. The newly allocated space, to actually write the appended data.

In a metadata-only journal, step 3 would not be logged. If step 3 was not done, but steps 1 and 2 are replayed during recovery, the file will be appended with garbage.

Adapted from:
"Journaling file system" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

04-A.4 Journaling File Systems is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.5: Linux Virtual File System

VFS

A **virtual file system** (**VFS**) is an abstract layer on top of a more concrete file system. The purpose of a VFS is to allow client applications to access different types of concrete file systems in a uniform way. A VFS can, for example, be used to access local and network storage devices transparently without the client application noticing the difference. It can be used to bridge the differences in Windows, classic Mac OS/macOS and Unix filesystems, so that applications can access files on local file systems of those types without having to know what type of file system they are accessing.

A VFS specifies an interface "contract" between the kernel and an actual file system. This contract specifies how the kernel communicates to the VFS, and the kernel does not worry about how to actually communicate to the various file systems that the VFS communicates with. Therefore, it is easy to add support for new file system types to the kernel simply by fulfilling the contract.

If a new release of one of the file systems includes changes, then the VFS might also have to be modified for that specific file system type. It is better to require changes in the VFS than in the kernel. Kernel changes can have a much larger impact, so it is better to change the VFS than it is to change the kernel.

In a similar manner, if a change that is made to the kernel impacts the communications between the kernel and the VFS, then, again, we can change the VFS, and not make code changes to the kernel. This greatly simplifies the maintenance of the kernel, as it doesn't have to be modified every time one of the supported file systems is changed.

Adapted from:
"Virtual file system" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

04-A.5: Linux Virtual File System is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.6: Partitions and Labels

Partitions

A hard disk can be divided into several **partitions**. Each partition functions as if it were a separate hard disk. The idea is that if you have one hard disk, and want to have, say, two operating systems on it, you can divide the disk into two partitions. Each operating system uses its partition as it wishes and doesn't touch the other ones. This way the two operating systems can co-exist peacefully on the same hard disk. Without partitions one would have to buy a hard disk for each operating system.

Floppies are not usually partitioned. There is no technical reason against this, but since they're so small, partitions would be useful only very rarely. CD-ROMs are usually also not partitioned, since it's easier to use them as one big disk, and there is seldom a need to have several operating systems on one.

The MBR, Boot Sectors and Partition Table

The information about how a hard disk has been partitioned is stored in its first sector (that is, the first sector of the first track on the first disk surface). The first sector is the *master boot record* (MBR) of the disk; this is the sector that the BIOS reads in and starts when the machine is first booted. The master boot record contains a small program that reads the partition table, checks which partition is active (that is, marked bootable), and reads the first sector of that partition, the partition's *boot sector* (the MBR is also a boot sector, but it has a special status and therefore a special name). This boot sector contains another small program that reads the first part of the operating system stored on that partition (assuming it is bootable), and then starts it.

The partitioning scheme is not built into the hardware, or even into the BIOS. It is only a convention that many operating systems follow. Not all operating systems do follow it, but they are the exceptions. Some operating systems support partitions, but they occupy one partition on the hard disk, and use their internal partitioning method within that partition. The latter type exists peacefully with other operating systems (including Linux), and does not require any special measures, but an operating system that doesn't support partitions cannot co-exist on the same disk with any other operating system.

Primary Partition Rules and Conventions

You can only create four primary partitions on any single physical hard drive. This partition limit extends to the Linux Swap partition as well as for any Operating System installation or extra special purpose partitions, such as separate /root, /home, /boot, etc., that you might want to create.

If you attempt to create more than four primary partitions, you will see a warning: "It is not possible to create more than 4 primary partitions." You do NOT have to have 4 primary partitions. If you decide to create an extended partition, be aware that it is counted as one of the 4 partitions.

Adapted from:
"5.9. Partitions" by Multiple contributors, The Linux Documentation Project is in the Public Domain, CC0
"How to Partition / Partitioning Basics" by Ben64, Community Help Wiki is licensed under CC BY-SA 4.0

04-A.6: Partitions and Labels is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.6.1: Partitions and Labels continued

Partitions

Extended Partitions

While in most circumstances four partitions are enough, there are circumstances in which you might need to create more than four partitions. This is the reason for an **Extended partition.**

An Extended partition is a special type of partition that contains "Free Space" in which more than the four Primary partitions can be created. Partitions created within the Extended partition are called **Logical partitions,** and any number of Logical partitions can be created within an Extended partition. The following conventions apply to Extended partitions:

1. When you create an Extended partition, it occupies one of the four Primary partition spots. When an Extended partition exists on a hard drive, only three Primary partitions may co-exist with it. If there are four Primary partitions already on a hard drive and you wish to create an Extended partition in which to create more, one of the Primary partitions must be deleted in order to create it.
2. Only one extended partition may be created on a hard drive (though you can have one extended partition on each of the hard drives connected to your system). The partition editor will not allow it, and it would serve no purpose at any rate. If you need the extra Extended partition space, you only need expand the one you have.

3. An Extended partition cannot be formatted with a filesystem, such as ext4, FAT, or NTFS, nor can it directly hold data. That is the function of the Logical Drives which are created within it.

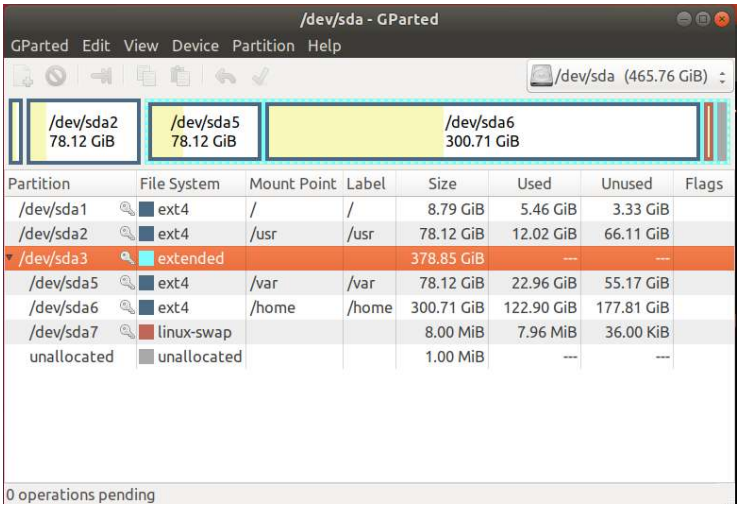


Figure **4.6.1.1**: The partitioning editor, showing primary, extended and logical drives("Gparted_screen_showing_partition.png" by Pbmacsodak, Wikimedia Commons is licensed under CC BY-SA 4.0)

Logical Partitions

A partition created within an Extended partition is called a **Logical partition.** Any number of Logical partitions may be created within an Extended partition, and they may be formatted with any filesystem.

All Operating Systems that I am aware of are able to access any Logical partition that is formatted to a compatible filesystem. For instance, while Windows will not recognize a Linux ext2 partition without a third-party driver, it will be able to access any partition formatted with FAT or NTFS, depending on the version of Windows. Linux, of course, will access all of these.

The image on the left shows the partitioning on my Linux system using the gparted editor. The / (root) and /usr partitions are the primary partitions. Then notice that /dev/sda3 is an extended partition. Within this extended partition then are the other three partitions on this system: /var, /home and the swap partition.

File System Labels

File system labels are assigned to file systems for easy identification. The labels may be up to 16 characters long and can be displayed or changed using the e2label command for ext3 or ext4 file systems. You MUST be logged in as root or use the sudo capability to execute the e2label command. You enter the /dev/designation for the partition and the name you want to use for that partition. For the above image, the command that was used to set the /dev/sda6 to be /home is:

```
# e2label /dev/sda6 home
```

Swap Space

The primary function of swap space is to substitute disk space for RAM memory when real RAM fills up and more space is needed.

For example, assume you have a computer system with 8GB of RAM. If you start up programs that don't fill that RAM, everything is fine and no swapping is required. But suppose the spreadsheet you are working on grows when you add more rows, and that, plus everything else that's running, now fills all of RAM. Without swap space available, you would have to stop working on the spreadsheet until you could free up some of your limited RAM by shutting down some other programs.

The kernel uses a memory management program that detects blocks (pages) of memory in which the contents have not been used recently. The memory management program swaps enough of these relatively infrequently used pages of memory out to a special partition on the hard drive specifically designated for "paging," or swapping. This frees up RAM and makes room for more data to be entered into your spreadsheet. Those pages of memory swapped out to the hard drive are tracked by the kernel's memory management code and can be paged back into RAM if they are needed.

The total amount of memory in a Linux computer is the RAM plus swap space and is referred to as virtual memory. Usually, the swap space equals twice the RAM capacity.

Adapted from:
"How to Partition / Partitioning Basics" by Ben64, Community Help Wiki is licensed under CC BY-SA 4.0
"An introduction to swap space on Linux systems" by David Both, OpenSource.com is licensed under CC BY-SA 4.0

04-A.6.1: Partitions and Labels continued is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.7: Disk Setup Process - fdisk

Setting Up a Disk

In this section, we look at several of the command line tools that can be used to prepare a disk for use. Take note - you only need to run EITHER **`fdisk`**, or **`parted`** - you do NOT need to run them both, as they do the same thing to the disk. They are alternatives to one another.

The `fdisk` Command

The **fdisk** command, also known as "format disk," is a dialog-driven command in Linux used for creating and manipulating disk partition table. It is used for the view, create, delete, change, resize, copy and move partitions on a hard drive using the dialog-driven interface.

fdisk allows you to create a maximum of four Primary partitions, and the number of Logical partitions depends on the size of the hard disk you are using. It allows the user:

- To create space for new partitions.
- To organize space for new drives.
- To Re-organize old drives.
- To copy or move data to new disks(partitions).

Syntax:

```
fdsik [ OPTIONS ] device

or

fdisk -l [device...]
```

Command Options:

Options	Option Meaning
-b sectorsize	Specify the sector size of the disk. Valid values are 512, 1024, 2048 or 4096. (Recent kernels know the sector size. Use this only on old kernels or to override the kernel's ideas.) Since util-linux-ng 2.17 fdisk differentiates between logical and physical sector size, this option changes both sector sizes to sectorsize.
-h	Print help and then exit.
-c	Switch off DOS-compatible mode. (Recommended)
-C cyls	Specify the number of cylinders of the disk. I have no idea why anybody would want to do so.
-H heads	Specify the number of heads of the disk. (Not the physical number, of course, but the number used for partition tables.) Reasonable values are 255 and 16.
-S sects	Specify the number of sectors per track of the disk. (Not the physical number, of course, but the number used for partition tables.) A reasonable value is 63.
-l	List the partition tables for the specified devices and then exit. If no devices are given, those mentioned in /proc/partitions (if that exists) are used.
-u	When listing partition tables, give sizes in sectors instead of cylinders.
-s partition	The size of the partition (in blocks) is printed on the standard output.

Below is an example of running the utility using /dev/sda. We will create a single partition that will occupy the entire disk.

1) Initiate **`fdisk`** with the following command:

```
sudo fdisk /dev/sda
```

2) **`fdisk`** will display the following menu:

```
Command (m for help): m <enter>
Command action
 a   toggle a bootable flag
 b   edit bsd disklabel
 c   toggle the dos compatibility flag
 d   delete a partition
 l   list known partition types
 m   print this menu
 n   add a new partition
```

```
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

  Command (m for help):
```

3) We want to add a new partition. Type "n" and press enter.

```
  Command action
   e   extended
   p   primary partition (1-4)
```

4) We want a primary partition. Enter "p" and enter.

```
  Partition number (1-4):
```

5) Since this will be the only partition on the drive, number 1. Enter "1" and enter.

```
  Command (m for help):
```

If it asks about the first cylinder, just type "1" and enter. (We are making one partition to use the whole disk, so it should start at the beginning.)

6) Now that the partition is entered, choose option "w" to write the partition table to the disk. Type "w" and enter.

```
  The partition table has been altered!
```

7) If all went well, you now have a properly partitioned hard drive that's ready to be formatted. Since this is the first partition, Linux will recognize it as /dev/sda1, while the **disk** that the partition is on is still /dev/sda.

Adapted from:
"Installing A New Hard Drive" by Robert Unverzagt, The Community Help Wiki is licensed under CC BY-SA 4.0
"fdisk command in Linux with examples" by Mandeep_Sheoran, Geeks for Geeks is licensed under CC BY-SA 4.0

04-A.7: Disk Setup Process - fdisk is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.7.1: Disk Setup Process - parted

The parted Command

GNU parted is a text-based partitioning utility (although, there is a GUI based tool as well) that works with most Linux disk labels and other disk types. This utility was designed to minimize the chance of data loss, but as always it is a good idea to back up all your files before running this program.

Syntax:

```
parted [options] [device [command [options...]...]]
```

Command Options:

Options	Option Meaning
-l, --list	Lists partition layout on all block devices.
-m, --machine	Displays machine parseable output.
-s, --script	Never prompts for user intervention.
-a alignment-type, --align alignment-type	Sets alignment for newly created partitions. Valid alignment types are: **none** Use the minimum alignment allowed by the disk type. **cylinder** Align partitions to cylinders. **minimal** Use minimum alignment as given by the disk topology information. This and the opt value will use layout information provided by the disk to align the Logical partition table addresses to actual physical blocks on the disks. The min value is the minimum alignment needed to align the partition properly to physical blocks, which avoids performance degradation. **optimal** Use optimum alignment as given by the disk topology information. This aligns to a multiple of the physical block size in a way that guarantees optimal performance.

Again, we will use /dev/sda, and assume that you want a single partition on the disk, occupying all the free space.

1) Start parted as follows:

```
pbmac@pbmac-server $ sudo parted /dev/sda
```

2) Create a new GPT disklabel (aka partition table):

```
(parted) mklabel gpt
```

3) Set the default unit to TB:

```
(parted) unit TB
```

4) Create one partition occupying all the space on the drive. For a 4TB drive:

```
(parted) mkpart
Partition name?  []? primary
File system type?  [ext2]? ext4
Start? 0
End? 4
```

5) Check that the results are correct:

```
(parted) print
```

There should be one partition occupying the entire drive.

6) Save and quit "parted":

```
(parted) quit
```

The partprobe Command

If you use **`fdisk`** or **`parted`** on a running live system and alter the partition table, the kernel still uses the old table. The new table will be used at the next reboot or after you run **`partprobe`**.

After you alter the partition table you should see a warning:

```
The partition table has been altered.

Calling ioctl() to re-read partition table.

Re-reading the partition table failed.: Device or resource busy
```

You simply need to run the **`partprobe`** command to advise the operating system that the partition table has been altered.

```
pbmac@pbmac-server $ sudo partprobe
```

Adapted from:
"InstallingANewHardDrive" by Robert Unverzagt, The Community Help Wiki is licensed under CC BY-SA 4.0
"How to partition and format a drive on Linux" by Seth Kenlon, OpenSource.com is licensed under CC BY-SA 4.0
"How to partition a disk in Linux" by Daniel Oh, OpenSource.com is licensed under CC BY-SA 4.0

04-A.7.1: Disk Setup Process - parted is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.7.2: Disk Setup Process - mkfs & mount

The `mkfs` Command

The **`mkfs`** command is used to make a file system on a device, usually a partition on a hard disk drive (HDD), or it can also be a USB drive, etc.

Syntax:

```
mkfs [ -V ] [ -t fstype ] [ fs-options ] filesys [ blocks ]
```

Command Options:

Options	Option Meaning
-t fstype	Specifies the type of file system to be built. If not specified, the default file system type (currently ext2) is used.
fs-options	File system-specific options to be passed to the real file system builder. Although not guaranteed, the following options are supported by most file system builders.
-c	Check the device for bad blocks before building the file system.
-l filename	Read the bad blocks list from filename.

Command Line Formatting

The **`mkfs`** utility is used to create filesystem (ext2, ext3, ext4, etc) on your Linux system. You should specify the device name to **`mkfs`** on which the filesystem is to be created. WARNING: Executing these commands will destroy all the data on your filesystem.

To format the new partition as ext4 file system you need to use the file system specific mkfs - where you can use ext3, ext4, etc.

```
pbmac@pbmac-server $ sudo mkfs.ext4 /dev/sda1
```

As always, substitute "/dev/sda1" with your own partition's path.

Create A Mount Point

Now that the drive is partitioned and formatted, you need to choose a mount point. This will be the location from which you will access the drive in the future. I would recommend using a mount point with "/media", as it is the default used by Ubuntu. For this example, we'll use the path "/media/mynewdrive"

```
pbmac@pbmac-server $ sudo mkdir /home/Project2020
```

Now we are ready to mount the drive to the mount point.

Mount the Drive

You can choose to have the drive mounted automatically each time you boot the computer, or manually only when you need to use it.

Automatic Mount at Boot

You'll need to edit /etc/fstab: (gedit is a simple text editor, you can use nano or vim).

```
pbmac@pbmac-server $ sudo gedit /etc/fstab
```

Once in editor you should add the line that begins with "/dev/sda6" to the end of the file. The following example to add a partition we have created and it will reside on the /dev/sda6 and is labeled /home/Project2020. The green text is an explanation of the various fields in this file.

```
# <file system> <mount point>      <type>  <options>            <dump>  <pass>
proc            /proc               proc    nodev,noexec,nosuid   0       0
/dev/sda6       /home/Project2020   ext4    defaults              0       2
[--------]      [----------------]  [--]    [------------------------------]
Linux Device      Mount point       File            Mount Options
                                  Sytem Type
```

Note: some Linux distros now recommend to use UUID in the /etc/fstab file. Check with your distro documentation to find out how to discover the UUID value for your partitions. Here is some help from Ubuntu.

If you are on a distribution of Linux that requires the use of the UUID mentioned above then you have a few more steps. You may have to consult your distros man pages for how to find the UUID for your storage devices. The example below uses commands on an Ubuntu distro.

```
pbmac@pbmac-server $ ls -l /dev/disk/by-uuid/
total 0
lrwxrwxrwx 1 root root 10 Jun 15 11:22 3709f1b7-6417-413e-a4cd-2bec3002ba25 -> ../../s
```

We know the partition will be mounted on /dev/sda6, so we now have the UUID value. Edit the /etc/fstab file and enter:

```
# <device>                                <dir>             <type>   <options>
UUID=fcdc8d9a-73f9-40a2-8a66-3d6f18f16ba3 /home/Project2020    ext4     defaults
```

You can now run **`sudo mount -a`** (which will mount all of the file system specified in the /etc/fstab file) or reboot the computer to have the changes take effect.

If you want to allow a normal user to create files on this drive, you can either give this user ownership of the top directory of the drive filesystem: (replace **USERNAME** with the actual username).

```
pbmac@pbmac-server $ sudo chown -R pbmac:pbmac /home/Project2020
```

Or, in a more flexible way and practical if you have several users, allow for instance the users in the proj2020 group (usually those who are meant to be able to mount removable disks, desktop users) to create files and sub-directories on the disk:

```
pbmac@pbmac-server $ sudo chgrp proj2020 /home/Project2020
pbmac@pbmac-server $ sudo chmod g+w /home/Project2020e
pbmac@pbmac-server $ sudo chmod +t /home/Project2020
```

The last **`chmod +t`** adds the sticky bit, so that people can only delete their own files and sub-directories in a directory, even if they have write permissions to it (see man chmod).

Manually Mount

Alternatively, you may want to manually mount the drive every time you need it.

For manual mounting, use the following command:

```
pbmac@pbmac-server $ sudo mount /dev/sda6 /home/Project2020
```

When you are finished with the drive, you can unmount it using:

```
pbmac@pbmac-server $ sudo umount /home/Project2020
```

Adapted from:
"InstallingANewHardDrive" by Robert Unverzagt, The Community Help Wiki is licensed under CC BY-SA 4.0
"mkfs Command in Linux with Examples" by shivaysabharwal, Geeks for Geeks is licensed under CC BY-SA 4.0

04-A.7.2: Disk Setup Process - mkfs & mount is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-A.8: The dev Directory

The /dev/directory

/dev is the location of special or device files. It is a very interesting directory that highlights one important aspect of the Linux filesystem - everything is a file or a directory. Look through this directory and you should hopefully see sda1, sda2 etc.... which represent the various partitions on the first drive of the system. /dev/cdrom represents your CD-ROM drive. This may seem strange but it will make sense if you compare the characteristics of files to that of your hardware. Both can be read from and written to. Take /dev/dsp, for instance. This file represents your speaker device. Any data written to this file will be re-directed to your speaker. If you try 'cat /boot/vmlinuz > /dev/dsp' (on a properly configured system) you should hear some sound on the speaker. That's the sound of your kernel! A file sent to /dev/lp0 gets printed.

The majority of devices are either block or character devices; however other types of devices exist and can be created. In general, 'block devices' are devices that store or hold data, while 'character devices' can be thought of as devices that transmit or transfer data. For example, diskette drives, hard drives and CD-ROM drives are all block devices while serial ports, mice and parallel printer ports are all character devices. There is a naming scheme of sorts but in the vast majority of cases these are completely illogical.

/dev/disk/by- Files

There are four different schemes for persistent naming: by-label, by-uuid, by-id and by-path.

- **`by-id`** creates a unique name depending on the hardware serial number.
- **`by-label -`** almost every file system type can have a label. All your volumes that have one are listed in the /dev/disk/by-label directory.
- **`by-path`** creates a unique name depending on the shortest physical path to the device
- **`by-uuid`** is a mechanism to give each filesystem a unique identifier. These identifiers are generated by the mkfs utilities.

Adapted from:
"1.5 /dev" by Various contributors, The Linux Documentation Project is in the Public Domain, CC0
"Persistent block device naming" by Various contributors, ArchLinux is in the Public Domain, CC0

04-A.8: The dev Directory is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-B.9: Logical Volume Management

EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.

4.1 Given a scenario, analyze system properties and remediate accordingly.

Device Mapper

The device mapper is a framework provided by the Linux kernel for mapping physical block devices onto higher-level virtual block devices. It forms the foundation of the logical volume manager (LVM), software RAIDs and dm-crypt disk encryption, and offers additional features such as file system snapshots.

In Linux the device mapper creates the virtual device, then passes data from the virtual device to one of the physical devices that make up the virtual device. Data can be also modified in transition, which is performed, for example, in the case of device mapper providing disk encryption or simulation of unreliable hardware behavior.

The /dev/mapper/ Directory

The /dev/mapper/ directory is where you will find all of the logical volumes on the system that are managed by LVM. The devices listed in this directory are typically formatted as: /dev/mapper/<volume group name>-<logical volume name> . In some cases, this directory may just include links to the actual logical volume location.

DM-Multipath

DM-Multipath provides input-output fail-over and load-balancing by using multiple data paths for Linux storage devices. By utilizing device-mapper, multiple paths on a redundant network are used to provide continuous availability and higher-bandwidth connectivity between the host server and the storage device. DM-Multipath handles the rerouting of data to an alternate path in the event of a path failure. DM-Multipath can also balance the data transmission load across all of the available network paths. DM-Multipath is based on the device mapper, which provides the basic framework that maps one block device onto another.

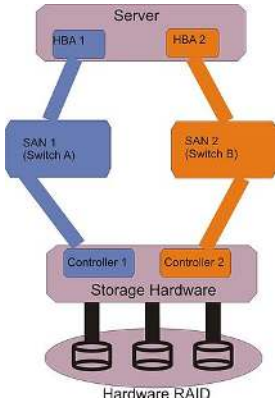


Figure **4.9.1**: Simple example of hardware RAID. ("Multipath_simple_example1.jpg" by 3itrekker, Wikimedia is in the Public Domain, CC0)

Without DM Multipath, each path from a server node to a storage controller is treated by the system as a separate device, even when the I/O path connects the same server node to the same storage controller. DM Multipath provides a way of organizing the I/O paths logically, by creating a single multipath device on top of the underlying devices.

In the image on the right, the server has 2 hosts bus adapters (HBA1, HBA2) that connect the server to a fiber channel network. Each connection goes through a different switch, and a different controller on the other end, where the storage devices are located. This provides redundant connections in case one network connect fails. It also provides for the ability to perform load balancing if necessary to improve performance.

The mdadm Command

`mdadm` is a Linux utility used to manage software RAID devices. The name is derived from the md (multiple device) device nodes it administers.

Redundant Array of Independent Disks (RAID) is a storage technology that combines multiple disk drive components (typically disk drives or partitions thereof) into a logical unit. Depending on the RAID implementation, this logical unit can be a file system or an additional transparent layer that can hold several partitions. Data is distributed across the drives in one of several ways called #RAID levels, depending on the level of redundancy and performance required. The RAID level chosen can thus prevent data loss in the event of a hard disk failure, increase performance, or a combination of both.

Adapted from:
"Linux DM Multipath" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

04-B.9: Logical Volume Management is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-B.9.1: LVM and other stuff

Logical Volume Manager (LVM)

LVM has some terminology that we need to be familiar with to make sense of what is happening.

- There are **physical volumes** - designated as pv. These are the same as the partitions that we have previously talked about, and are the individual physical partition of the actual disk drives. These are physical block devices are used by LVM as the raw building material for higher levels of abstraction. Physical volumes are regular storage devices. LVM writes a header to the device to allocate it for management.
- **Volume groups,** which have the vg extension, are physical volumes that LVM combines into storage pools known as volume groups. Volume groups behave as a single logical device with combined storage capacity of the individual physical volumes.
- Last we have the **logical volumes,** an extension of lv or lvm. The volume groups can be divided up into any number of logical volumes. Logical volumes are basically equivalent to partitions on a physical disk, but they offer much more flexibility. Logical volumes are the primary component that users and applications will interact with. The logical volumes become the mount points that show up in the Linux filesystem.



Figure **4.9.1.1**: An example LVM configuration. ("Howto install Ubuntu 11.10 to LVM/LUKS system" by Jari Laamanen is licensed under CC BY-SA 3.0)

Using the image to the left, starting at the bottom of the image: we have 3 physical volumes. Let's call them /dev/sda, /dev/sdb and /dev/sdc. In the picture they are referred to as PV - physical volumes. These are the physical disks with partitions created per the discussion in the last section. The first disk has 3 partitions - 1) is assigned as the boot disk for this machine 2) is represented by green and 3) represented by the red. Disk #2 has a single partition which is the entire disk. Finally, disk #3 has two partitions.

The colors in this picture are just to help us. So, the green partition from disk #1 and the green partition from disk #3 are grouped together to create a volume group, which is also a logical volume, which then gets mounted as / (root) on the system. The three red partitions are grouped together in a volume group, and it is used as logical volume and mounted on /home.

Be aware that the volume groups can be broken into multiple logical volumes - in this example though each volume group is used for a single logical volume. The DM-CRYPT shown in the image is some encryption piece that has nothing to do with our discussion.

Advantages of LVM

LVM gives you more flexibility than just using normal hard drive partitions. Some advantages of LVM:

- Use any number of disks as one big disk.
- Have logical volumes stretched over several disks.
- Create small logical volumes and resize them "dynamically" as they get filled up.
- Resize logical volumes regardless of their order on disk. It does not depend on the position of the LV within VG, there is no need to ensure surrounding available space.

- Resize/create/delete logical and physical volumes online. File systems on them still need to be resized, but some (such as ext4) support online resizing.
- Online/live migration of LV being used by services to different disks without having to restart services.
- Snapshots allow you to backup a frozen copy of the file system, while keeping service downtime to a minimum.
- Support for various device-mapper targets, including transparent filesystem encryption and caching of frequently used data. This allows creating a system with (one or more) physical disks (encrypted with LUKS) and **LVM on top** to allow for easy resizing and management of separate volumes (e.g. for `/`, `/home`, `/backup`, etc.) without the hassle of entering a key multiple times on boot.

Disadvantages of LVM

- Additional steps in setting up the system, more complicated. Requires (multiple) daemons to constantly run.
- If dual-booting, note that Windows does not support LVM; you will be unable to access any LVM partitions from Windows.
- If your physical volumes are not on a RAID-1, RAID-5 or RAID-6 losing one disk *can* lose one or more logical volumes if you span (or extend) your logical volumes across multiple non-redundant disks.

Adapted from:
"LVM" by Multiple Contributors, Arch Linux is licensed under CC BY-SA 3.0

04-B.9.1: LVM and other stuff is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-B.9.2: LVM Tools

LVM Management Tools

The Logical Volume Manager (LVM) provides tools to create virtual block devices from physical devices. Virtual devices may be easier to manage than physical devices, and can have capabilities beyond what the physical devices provide themselves. A Volume Group (VG) is a collection of one or more physical devices, each called a Physical Volume (PV). A Logical Volume (LV) is a virtual block device that can be used by the system or applications. Each block of data in an LV is stored on one or more PV in the VG, according to algorithms implemented by Device Mapper (DM) in the kernel.

The tools are separated into commands associated with each of the components: the PV tools, the VG tools and the LG tools.

The command tools dealing with the Physical Volume (PV) are as follows:

```
    pvchange   Change attributes of a Physical Volume.
    pvck       Check Physical Volume metadata.
    pvcreate   Initialize a disk or partition for use by LVM.
    pvdisplay  Display attributes of a Physical Volume.
    pvmove     Move Physical Extents.
    pvremove   Remove a Physical Volume.
    pvresize   Resize a disk or partition in use by LVM2.
    pvs        Report information about Physical Volumes.
    pvscan     Scan all disks for Physical Volumes.
```

The command tools dealing with the Volume Group (VG) are as follows:

```
    vgcfgbackup   Backup Volume Group descriptor area.
    vgcfgrestore  Restore Volume Group descriptor area.
    vgchange      Change attributes of a Volume Group.
    vgck          Check Volume Group metadata.
    vgconvert     Convert Volume Group metadata format.
    vgcreate      Create a Volume Group.
    vgdisplay     Display attributes of Volume Groups.
    vgexport      Make volume Groups unknown to the system.
    vgextend      Add Physical Volumes to a Volume Group.
    vgimport      Make exported Volume Groups known to the system.
    vgimportclone Import and rename duplicated Volume Group (e.g. a hardware snapshot)
    vgmerge       Merge two Volume Groups.
    vgmknodes     Recreate Volume Group directory and Logical Volume special files
    vgreduce      Reduce a Volume Group by removing one or more Physical Volumes.
    vgremove      Remove a Volume Group.
    vgrename      Rename a Volume Group.
    vgs           Report information about Volume Groups.
    vgscan        Scan all disks for Volume Groups and rebuild caches.
    vgsplit       Split a Volume Group into two, moving any logical volumes from one V
                  to another by moving entire Physical Volumes.
```

The command tools dealing with the Logical Volume (LV) are as follows:

```
    lvchange    Change attributes of a Logical Volume.
    lvconvert   Convert a Logical Volume from linear to mirror or snapshot.
    lvcreate    Create a Logical Volume in an existing Volume Group.
    lvdisplay   Display attributes of a Logical Volume.
    lvextend    Extend the size of a Logical Volume.
    lvmconfig   Display the configuration information after loading
                lvm.conf(5) and any other configuration files.
    lvmdiskscan Scan for all devices visible to LVM2.
    lvmdump     Create lvm2 information dumps for diagnostic purposes.
    lvreduce    Reduce the size of a Logical Volume.
    lvremove    Remove a Logical Volume.
    lvrename    Rename a Logical Volume.
    lvresize    Resize a Logical Volume.
    lvs         Report information about Logical Volumes.
    lvscan      Scan (all disks) for Logical Volumes.
```

Adapted from:
"LVM2 tools (man page)" by Multiple contributors is in the Public Domain, CC0

04-B.9.2: LVM Tools is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-C.10: Mounting Linux File Systems

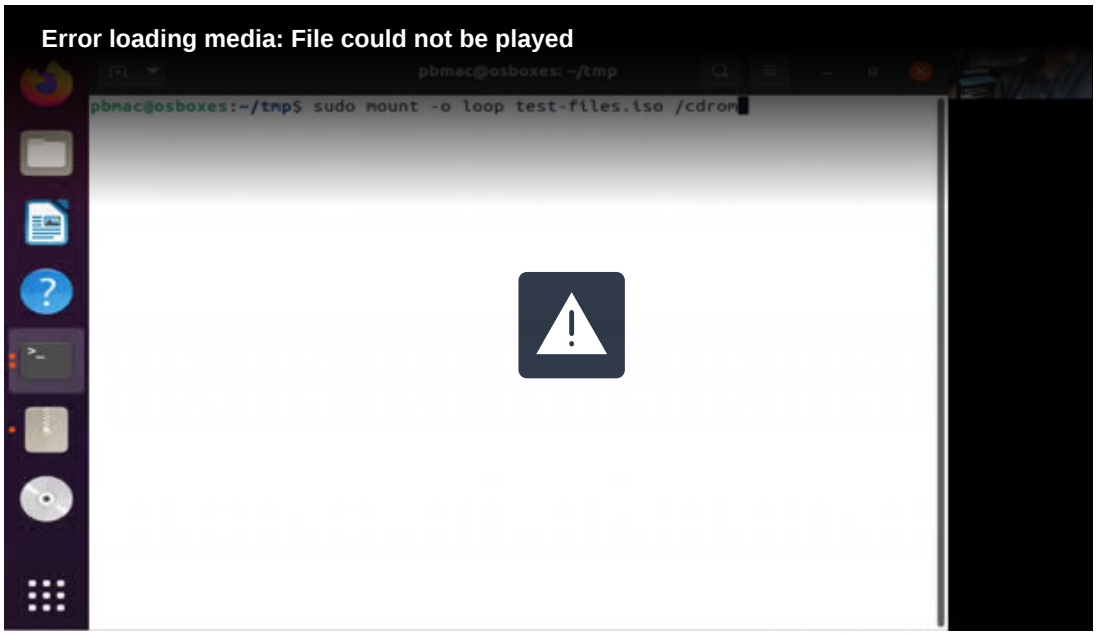
EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.

Mount Points

A mount point is simply a directory, like any other, that is created as part of the root filesystem. So, for example, the home filesystem is mounted on the directory /home. Filesystems can be mounted at mount points on other non-root filesystems but this is less common.

The ability to mount and unmount filesystems is a critical skill for Linux administrators. In this video you will see how to mount and unmount a simple file system. This technique applies to all filesystems.



The `mount` Command

The Linux root filesystem is mounted on the root directory (/) very early in the boot sequence. Mounting of filesystems during the startup process is managed by the /etc/fstab configuration file. An easy way to remember this is that fstab stands for "file system table," and it is a list of filesystems that are to be mounted, their designated mount points, and any options that might be needed for specific filesystems.

Filesystems are mounted on an existing directory/mount point using the **`mount`** command. In general, any directory that is used as a mount point should be empty and not have any other files contained in it. Linux will not prevent users from mounting one filesystem over one that is already there or on a directory that contains files. If you mount a filesystem on an existing directory or filesystem, the original contents will be hidden and only the content of the newly mounted filesystem will be visible.

Syntax:

```
mount -t type device dir
```

- "device" is the device special file, such as /dev/sda1.
- dir is the mount point where the partition is to be mounted.
- If you leave the **dir** argument out of command it looks for a **mount point** in **/etc/fstab**.
- You can use **–source** or **–target** to avoid ambivalent interpretation.

```
mount --target /mountpoint
```

- **/etc/fstab** usually contains information about which device is need to be mounted where.

Command Options:

Options	Option Meaning
auto / noauto	With the auto option, the device will be mounted automatically at bootup or when the mount -a command is issued. auto is the default option. For the device not to be mounted automatically, the noauto option is used in /etc/fstab. With noauto, the device can be only mounted explicitly.
exec / noexec	exec lets binaries that are on the partition be executed, whereas noexec is the opposite. noexec might be useful for a partition that contains no binaries, like /var, or contains binaries the user may not want to execute on the system, or that cannot even be executed on the system, as might be the case of a Windows partition.
rw / ro	Mount the filesystem in either read write or read only mode. Explicitly defining a file system as rw can alleviate some problems in file systems that default to read only.
sync / async	How the input and output to the filesystem should be done. sync means it is done synchronously; async would mean it is to be done asynchronously.
user / users / nouser user	Permits any user to mount the filesystem. This automatically implies noexec, nosuid, nodev unless overridden. If nouser is specified, only root can mount the filesystem. If users is specified, every user in group users will be able to unmount the volume.
defaults	Use default settings. Default settings are defined per file system at the file system level. For ext3 file systems these can be set with the tune2fs command. The normal default for Ext3 file systems is equivalent to rw, suid, dev, exec, auto, nouser, async (no acl support). Modern Red Hat based systems set acl support as default on the root file system but not on user created Ext3 file systems. Some file systems such as XFS enable acls by default. Default file system mount attributes can be overridden in /etc/fstab.

The `umount` Command

The **`umount`** command detaches the file system(s) mentioned from the file hierarchy. A file system is specified by giving the directory where it has been mounted. Giving the special device on which the file system lives may also work, but is obsolete, mainly because it will fail in case this device was mounted on more than one directory.

Syntax:

```
umount -a [-dflnrv] [-t vfstype] [-O options]
umount [-dflnrv] {dir|device}...
```

Command Options:

https://eng.libretexts.org/@go/page/26846

Options	Option Meaning
-h	Print a help message and exit.
-f	Force unmount (in case of an unreachable NFS system). (Requires kernel 2.1.116 or later.)
-l	“Lazy” unmount. Detach the filesystem from the filesystem hierarchy now, and cleanup all references to the filesystem as soon as it is not busy anymore.
-t vfstype	Indicate that the actions should only be taken on file systems of the specified type. More than one type may be specified in a comma separated list. The list of file system types can be prefixed with no to specify the file system types on which no action should be taken.
-O options	Indicate that the actions should only be taken on file systems with the specified options in /etc/fstab. More than one option type may be specified in a comma separated list. Each option can be prefixed with no to specify options for which no action should be taken.

Adapted from:
"mount command in Linux with Examples" by Vivek Agrawal, Geeks for Geeks is licensed under CC BY-SA 4.0
"fstab" by Multiple Contributors, Wijipedia is licensed under CC BY-SA 3.0
"unmount filesystems" by Multiple contributors, man7.org is in the Public Domain, CC0

04-C.10: Mounting Linux File Systems is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-D.11: File System Management

EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.
2.7 Explain the use and operation of Linux devices.
4.1 Given a scenario, analyze system properties and remediate accordingly.

Linux File Systems

The /proc/mounts File

To display all mounted file systems Linux provides the information in /proc/mounts. It is actually not a real file, but part of the virtual file system that represents the status of mounted objects as reported by the Linux kernel. It is in a format similar to the /etc/fstab file: the mounted device, the mount point, the file system type, mount flags, dump frequency, and file system check order. Even though it is not truly a file, you can use commands like **`cat`** or **`less`** on this "file."

The /etc/mtab File

This file lists all currently mounted file systems along with their initialization options. mtab is very similar to /proc/mounts in that they both report the status of currently mounted file systems. However, since the kernel updates /proc/mounts is typically more accurate and includes more up-to-date information about the file systems. The **`mount`** command maintains the contents of the mtab file.

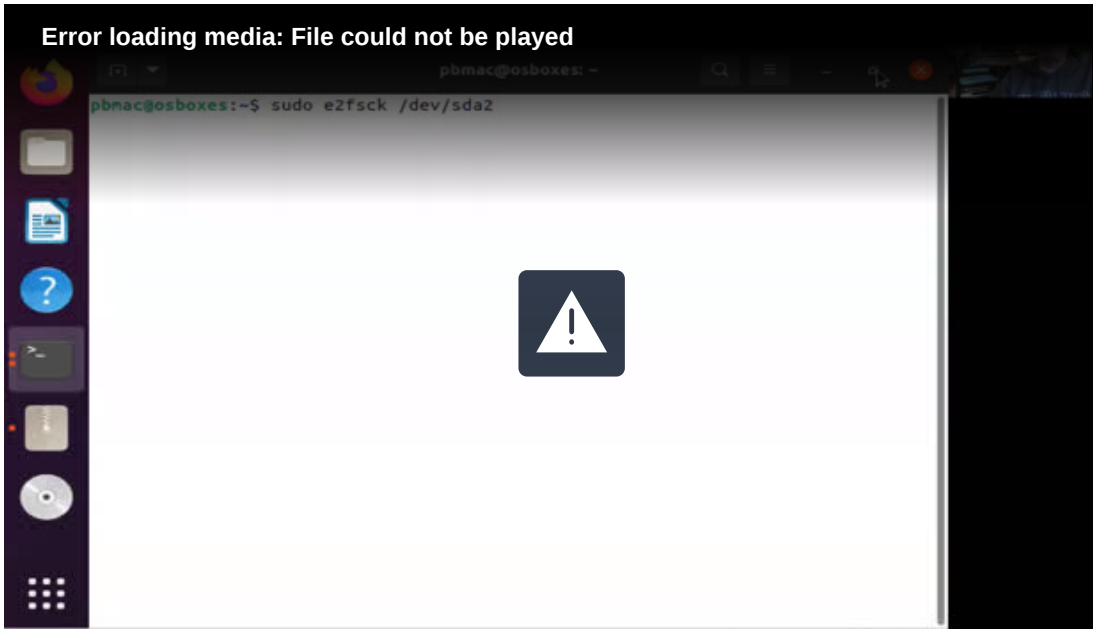
The /proc/partitions File

The /proc/partitions file contains partition block allocation information. Like the other files in the /proc file system, it is not a real file but part of the virtual file system. The format of /proc/partitions contains 4 columns:

• **major**—This value represents the class of device so that it can be mapped to an appropriate driver.
• **minor**—This value separates partitions into physical devices. This corresponds to the number at the end of the partition's name (such as sda1, sda2, etc).
• **#blocks**—How many physical blocks the partition takes up.
• **name**—The name of the partition.

```
major    minor  #blocks  name
   8        0  488386584 sda
   8        1    9216000 sda1
   8        2   81920000 sda2
   8        5   81920000 sda5
   8        6  315317248 sda6
   8        7       8192 sda7
  11        0    1048575 sr0
```

Learn how to use the e2fsck command to troubleshoot file system in this video.



The lsblk Command

The lsblk command lists information about all available or the specified block devices. The lsblk command reads the sysfs file system and udev db to gather information. If the udev db is not available or lsblk is compiled without udev support than it tries to read LABELs, UUIDs and file system types from the block device. In this case root permissions are necessary.

Syntax:

```
lsblk [ OPTIONS ]
lsblk [ OPTIONS ] device
```

Command Options:

Options	Option Meaning
-a, --all	Also list empty devices. (By default they are skipped.)
-e, --exclude list	Exclude the devices specified by the comma-separated list of major device numbers. Note that RAM disks (major=1) are excluded by default. The filter is applied to the top-level devices only.
-f, --fs	Output info about file systems. This option is equivalent to -o NAME,FSTYPE,LABEL,UUID,MOUNTPOINT. The authoritative information about file systems and raids is provided by the blkid(8) command.
-l, --list	Produce output in the form of a list.
-m, --perms	Output info about device owner, group and mode. This option is equivalent to -o NAME,SIZE,OWNER,GROUP,MODE.
-n, --noheadings	Do not print a header line.
-o, --output list	Specify which output columns to print. Use --help to get a list of all supported columns. The default list of columns may be extended if list is specified in the format +list (e.g. lsblk -o +UUID)./pre>

A simple example:

```
pbmac@pbmac-server $ lsblk -f
NAME   FSTYPE  LABEL UUID                                 MOUNTPOINT
loop0  squashf                                            /snap/core18/1705
loop1  squashf                                            /snap/core18/1754
loop2  squashf                                            /snap/nmap/1356
sda
├─sda1 ext4    /     6e73a278-7609-4612-b1dc-5e7451bb3f0d /
├─sda2 ext4    /usr  815e345f-4e06-4a42-b19d-35cf6ef7fd32 /usr
├─sda5 ext4    /var  ad74f5a8-188a-4282-93a4-2107dfb08fc5 /var
├─sda6 ext4    /home 3709f1b7-6417-413e-a4cd-2bec3002ba25 /home
└─sda7 swap          68e33827-8998-49f9-98a1-94415abad3bb [SWAP]
```

The **blkid** Command

The **blkid** command reads information directly from devices and for non-root users it returns cached unverified information. **blkid** is mostly designed for system services and to test libblkid functionality. Options and syntax are very similar to the **lsblk** command.

Syntax:

```
blkid [options] [device name]
```

When device is specified, tokens from only this device are displayed. It is possible to specify multiple device arguments on the command line. If none is given, all devices which appear in */proc/partitions* are shown, if they are recognized. See the **blkid** man page for additional options.

```
pbmac@pbmac-server $ blkid
/dev/sda1: LABEL="/" UUID="6e73a278-7609-4612-b1dc-5e7451bb3f0d" TYPE="ext4" PARTUUID="a3f88df0-0
/dev/sda2: LABEL="/usr" UUID="815e345f-4e06-4a42-b19d-35cf6ef7fd32" TYPE="ext4" PARTUUID="a3f88df
/dev/sda5: LABEL="/var" UUID="ad74f5a8-188a-4282-93a4-2107dfb08fc5" TYPE="ext4" PARTUUID="a3f88df
```

```
/dev/sda6: LABEL="/home" UUID="3709f1b7-6417-413e-a4cd-2bec3002ba25" TYPE="ext4" PARTUUID="a3f88d
/dev/sda7: UUID="68e33827-8998-49f9-98a1-94415abad3bb" TYPE="swap" PARTUUID="a3f88df0-07"
pbmac@pbmac-server $ blkid -i sda1
/dev/sda1:
   MINIMUM_IO_SIZE="512"
   PHYSICAL_SECTOR_SIZE="512"
   LOGICAL_SECTOR_SIZE="512"
```

04-D.11: File System Management is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-D.11.1: File System Management - ext Filesystem Tools

Linux File System Management Tools

ext Utilities

e2fsprogs (sometimes called the e2fs programs) is a set of utilities for maintaining the ext2, ext3 and ext4 file systems. Since those file systems are often the default for Linux distributions, it is commonly considered to be essential software.

Despite what its name might suggest, e2fsprogs works not only with ext2, but also with ext3 and ext4. Although ext3's journaling capability can reduce the need to use e2fsck, it is sometimes still necessary to help protect against kernel bugs or bad hardware.

This is a list of the more commonly used Linux commands.
dumpe2fs - which prints superblock and block group information.
e2fsck - an fsck program that checks for and corrects inconsistencies.
resize2fs - which can expand and shrink ext2, ext3 and ext4 file systems. On-line support was added in 2006.
tune2fs - used to modify file system parameters.

The `dumpe2fs` Command

The **`dumpe2fs`** command is used to print the super block and blocks group information for the filesystem present on device.

- Can be used with **ext2**/**ext3**/**ext4** filesystem for information.
- The printed information may be old or inconsistent when it is used with a mounted filesystem.
- Don’t forget to unmount your partition before using this command. When used with a mounted filesystem, the printed information may be old or inconsistent.

Syntax:

```
dumpe2fs [ OPTIONS] [ -o superblock=superblock ] [ -o blocksize=block-size ] device
```

Command Options:

Options	Option Meaning
-b	Print the blocks which are reserved as bad in the filesystem.
-f	Force dumpe2fs to display a filesystem even though it may have some filesystem feature flags which dumpe2fs may not understand (and which can cause some of dumpe2fs's display to be suspect).
-i	Display the filesystem data from an image file created by e2image, using device as the pathname to the image file.
-x	Print the detailed group information block numbers in hexadecimal format.

The command outputs a LOT of information; here is just the first 15 lines of a typical output.

```
pbmac@pbmac-server $ dumpe2fs /dev/sda1
Filesystem volume name:   /
Last mounted on:          /
Filesystem UUID:          6e73a278-7609-4612-b1dc-5e7451bb3f0d
Filesystem magic number:  0xEF53
```

```
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_r
                          large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              577088
Block count:              2304000
```

The `e2fsck` Command

The **`e2fsck`** command is used to check the integrity of the ext2/ext3/ext4 family of file systems. The command checks for correctness and validity of the filesystem. For ext3 and ext4 filesystems that use a journal, if the system has been shut down uncleanly without any errors, normally after replaying the committed transactions in the journal, the file system should be marked as clean. Hence, for filesystems that use journaling, **`e2fsck`** will normally replay the journal and exit, unless its superblock indicates that further checking is required.

Syntax:

```
e2fsck [ OPTIONS ] [ -b superblock ] [ -B blocksize ] [ -l|-L bad_blocks_file ]
[ -C fd ] [ -j external-journal ] [ -E extended_options ] device
```

Command Options:

Options - only a partial listing	**Option Meaning**
-b superblock	Instead of using the normal superblock, use an alternative superblock specified by superblock.
-B blocksize	Normally, e2fsck will search for the superblock at various different block sizes in an attempt to find the appropriate block size.
-l filename	Add the block numbers listed in the file specified by filename to the list of bad blocks.
-L filename	Set the bad blocks list to be the list of blocks specified by filename.
-c	This option causes e2fsck to use badblocks(8) program to do a read-only scan of the device in order to find any bad blocks.
-C fd	This option causes e2fsck to write completion information to the specified file descriptor so that the progress of the filesystem check can be monitored.
-E extended_options	Set e2fsck extended options. Extended options are comma separated, and may take an argument using the equals ('=') sign.
-j external-journal	Set the pathname where the external-journal for this filesystem can be found.

Note that in general it is not safe to run **`e2fsck`** on mounted filesystems. The only exception is if the -n option is specified, and -c, -l, or -L options are *not* specified. However, even if it is safe to do so, the results printed by **`e2fsck`** are not valid if the filesystem is mounted. If **`e2fsck`** asks whether or not you should check a filesystem which is mounted, the only correct answer is "no'.' Only experts who really know what they are doing should consider answering this question in any other way.

The **`fsck`** command is a front end to the **`e2fsck`**. So, when you run the **`fsck`** command it calls the appropriate command for the type of filesystem that is present on the system.

Note that in general it is not safe to run **`e2fsck`** on mounted filesystems as you do not want to make changes to an active filesystem.

```
pbmac@pbmac-server $ fsck -n /dev/sda5
fsck from util-linux 2.31.1
e2fsck 1.44.1 (24-Mar-2018)
Warning!  /dev/sda5 is mounted.
Warning: skipping journal recovery because doing a read-only filesystem check.
/var: clean, 422698/5120000 files, 5994387/20480000 blocks
```

Since this is a live system and the filesystem is mounted we use the -n option to assume an answer of 'no' to all questions (which might attempt to alter the filesystem). Notice the output shows the version of **`fsck`** AND **`e2fsck`**. On the last line, we see that this filesystem is 'clean.'

The `resize2fs` Command

The **`resize2fs`** program will resize ext2, ext3, or ext4 file systems. It can be used to enlarge or shrink an unmounted file system located on *device*. If the filesystem is mounted, it can be used to expand the size of the mounted filesystem, assuming the kernel and the file system supports on-line resizing. (Modern Linux 2.6 kernels will support on-line resize for file systems mounted using ext3 and ext4; ext3 file systems will require the use of file systems with the resize_inode feature enabled.)

Syntax:

```
resize2fs [ OPTIONS ] [ -d debug-flags ] [ -S RAID-stride ] device [ size ]
```

Command Options:

Options	Option Meaning
-f	Forces resize2fs to proceed with the filesystem resize operation, overriding some safety checks which resize2fs normally enforces.
-F	Flush the filesystem device's buffer caches before beginning. Only really useful for doing resize2fs time trials.
-M	Shrink the filesystem to the minimum size.
-p	Prints out a percentage completion bar for each resize2fs operation, so that the user can keep track of what the program is doing.
-P	Print the minimum size of the filesystem and exit.

The **`resize2fs`** program does not manipulate the size of partitions. If you wish to enlarge a filesystem, you must make sure you can expand the size of the underlying partition first.

```
pbmac@pbmac-server $ resize2fs /dev/sdb1
resize2fs 1.44.1 (24-Mar-2018)
Resizing the filesystem on /dev/sdb1 to 1048576 (4k) blocks.
The filesystem on /dev/sdb1 is now 1048576 (4k) blocks long.
```

The `tune2fs` Command

`tune2fs` allows the system administrator to adjust various tunable filesystem parameters on Linux ext2, ext3, or ext4 filesystems. The current values of these options can be displayed by using the -l option to **`tune2fs`** program, or by using the **`dumpe2fs`** program.

Syntax:

```
tune2fs [OPTIONS] device
```

Command Options:

Options	Option Meaning
-c max-mount-counts	Adjust the number of mounts after which the filesystem will be checked by e2fsck.
-C mount-count	Set the number of times the filesystem has been mounted.
-i interval-between-checks[d|m|w]	Adjust the maximal time between two filesystem checks.
-r reserved-blocks-count	Set the number of reserved filesystem blocks.

Superblocks

In many of the descriptions in this section we have used the word 'superblock'. The first block in the cylinder group is a superblock, which contains the metadata that defines the other filesystem structures and locates them on the physical disk. Some of the additional groups in the partition will have backup superblocks, but not all. A damaged superblock can be replaced by using a disk utility such as **dd** to copy the contents of a backup superblock to the primary superblock. It does not happen often.

The image to the right shows the structure of a cylinder group in the EXT filesystems.

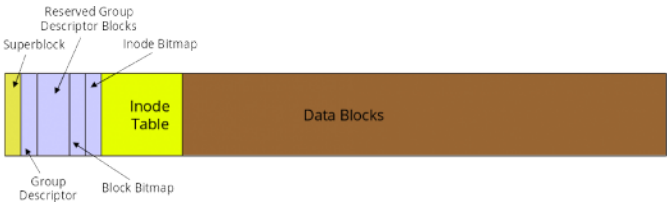


Figure **4.11.1.1**: The structure of a cylinder group in the EXT filesystems. ("The structure of a cylinder group in the EXT filesystems" by David Both, OpenSource.com is licensed under CC BY-SA 4.0)

The `fsck` Command

Linux runs **`fsck`** automatically at startup on partitions that are marked for this in /etc/fstab, which is the file containing information about the filesystems that are permanently mounted on the system. The normal behavior is to perform just a quick cursory examination of a partition if it's been unmounted cleanly. Despite this operation, the Linux boot process isn't delayed because of a filesystem check unless the system wasn't shut down properly. If a filesystem is marked unclean for some reason, that means a lengthy check of the filesystem is probably required.

NOTE: fsck is basically a front end to a variety of file system checking tools. It will determine the appropriate tools to use in order to properly run a file system check on a given files system.

Syntax:

```
fsck [OPTIONS...] [ARGUMENTS...]
```

When errors are detected on the filesystems, most of the time the **`fsck`** command needs your authorization in order to perform any modification on the filesystem. So it will ask you questions every time it finds an error and will try to modify it.

There are numerous options for the **`fsck`** command. Since this command has a lot of power and can possibly alter the disk configuration, make sure you consult the man page and that you are confident in what you are doing.

Adapted from:
"An introduction to Linux's EXT4 filesystem" by David Roth, OpenSource.com is licensed under CC BY-SA 4.0
"How to Use fsck Command in Linux" by Alain Francois, LinoXide is licensed under CC BY 3.0

04-D.11.1: File System Management - ext Filesystem Tools is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-D.11.2: File System Management - xfs Tools

Linux File System Management Tools (continued)

XFS Tools

Just as there are command line tools to manage the ext filesystem, there are tools that allow you to work with the XFS file system. Here are a few of them:

`xfs_growfs` - this allows you to grow an XFS file system. You need to specify a new size of mount point along with -D switch. -D takes argument number as file system **blocks**. Without the -D switch, **`xfs_growfs`** will grow the filesystem to the maximum available on the specified device.

`xfs_repair` - performs a file system consistency check and repair of the filesystem. Running the command with -n switch will only scan and report which modifications need to be done. If you run the command without -n switch, it will modify the file system as necessary to make it clean.

`xfs_db` - normally used for examining the XFS file system. The version command used to enable features in the file system. Without any argument, the current version and feature bits are printed.

```
pbmac@pbmac-server # umount /home
pbmac@pbmac-server # xfz_db /dev/xvdf5
xfz_db> version
versionnum [0xb4a5+0x18a] = V5,NLINK,DIRV2,ALIGN,LOGV2,EXTFLG,MOREBITS,ATTR2,LAZYSBCOU
xfz_db> quit
```

`xfs_info` - displays details of the XFS file system like block size and number of blocks which helps you in calculating new block numbers for growing XFS file system.

`xfs_admin` - used to change the parameters of an XFS file system, such as the label and/or UUID.

`xfs_metadump` - will copy the superblock metadata of the XFS file system to a disk file.

`xfs_copy` - used to copy the contents of the XFS file system to another location.

04-D.11.2: File System Management - xfs Tools is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-E.12: Linux Directory Structure

EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.

Files

In Linux, everything is a file. There are some special types of files, which we will talk about, but they are all files. We have looked at file permissions before. The very first position in a file's permission tells us what type of file it is.

```
The first position is:
A dash, this is an ordinary file or a hard link
-rwxr-xr-x

The letter d, this file is a directory
drwxr-xr-x

The letter c, a special character file
crw-------

The letter b, a block special file
brw-rw----

The letter l, this is a symbolic link
lrwxrwxrwx

The named pipe and the socket are more infrequent.
```

The following list outlines: 1) the character that will show up in the first column for the various file types 2) what it is called and 3) a brief description of that file type. We will cover this in much greater detail in a later section.

```
-   Ordinary or Regular File - contain text, are executable programs or are data files
-   Hard Link - an additional name for an existing file
d   Directory - special files that store both other ordinary and special files in an o
c   Character special file - device files that communicates by sending and receiving s
b   Block special file - device files that communicates by sending entire blocks of da
l   Symbolic link - a reference that points to another file on the system
p   Named pipe - a file that is used by two process to communicate with each other
s   Socket - a file that connects the output of one process to the input of another pr
```

Linux happens to provide a file command that basically helps you determine what type of file you are looking at. Here are some simple examples of the **`file`** command - see the man page for additional options.

```
pbmac@pbmac-server $ file /home/pbmac
/home/pbmac: directory

pbmac@pbmac-server $ file /etc/passwd
/etc/passwd: ASCII text
```

```
pbmac@pbmac-server $ file vmlinuz
vmlinuz: symbolic link to boot/vmlinuz-4.15.0-91-generic

pbmac@pbmac-server $ file my-pipe
my-pipe: fifo (named pipe)
```

File Naming Rules

Each file name must be unique within the directory where it exists. File naming guidelines are as follows:

- A file name can be up to 255 characters long and can contain letters, numbers, and underscores.
- The operating system is case-sensitive, which means it distinguishes between uppercase and lowercase letters in file names. Therefore, FILEA, FiLea, and filea are three distinct file names, even if they reside in the same directory.
- File names should be as descriptive and meaningful as possible.
- Directories follow the same naming conventions as files.
- Certain characters have special meaning to the operating system. Avoid using these characters when you are naming files. These characters include the following:
  \ " ' * ; - ? [ ] ( ) ~ ! $ { } < > # @ & | space tab newline
- File names may NOT contain: a NULL character, or a / (forward slash).
- A file may not be named . or .. as these are reserved names.
- A file name is hidden from a normal directory listing if it begins with a dot (.). When the **ls** command is entered with the **-a** flag, the hidden files are listed along with regular files and directories.

Adapted from:
"A beginner's guide to Linux permissions " by Bryant Son, OpenSource.com is licensed under CC BY-SA 4.0

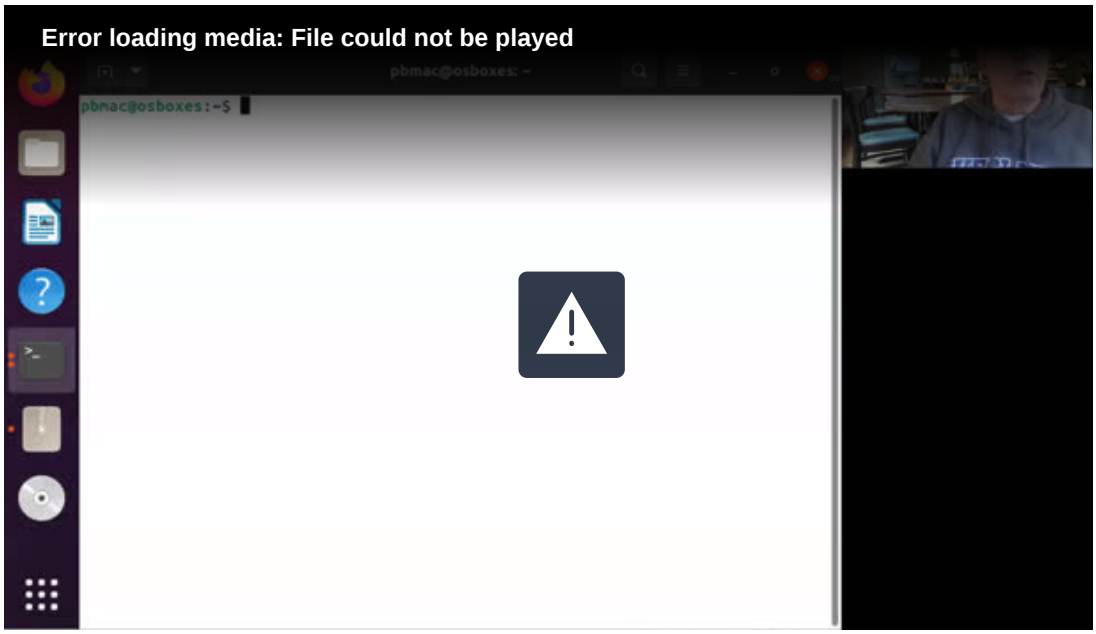
04-E.12: Linux Directory Structure is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-E.12.1: Linux Directory Structure - Hierarchy

Filesystem Hierarchy Standard

The **Filesystem Hierarchy Standard** (**FHS**) defines the directory structure and directory contents in Linux distributions. It is maintained by the Linux Foundation. The latest version is 3.0, and was released on 3 June 2015.

The video discusses the Filesystem Hierarchy Standard and explains what it means.



Directory Structure

In the FHS, all files and directories appear under the root directory / , even if they are stored on different physical or virtual devices. Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.

Most of these directories exist in all Unix-like operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS, and are not considered authoritative for platforms other than Linux.

Directory	Description
/	*Primary hierarchy* root and root directory of the entire file system hierarchy.
/bin	Essential command binaries that need to be available in single user mode; for all users, *e.g.*, cat, ls, cp.
/boot	Boot loader files, *e.g.*, kernels, initrd.
/dev	Device files, *e.g.*, /dev/null , /dev/disk0 , /dev/sda1 , /dev/tty , /dev/random.
/etc	Host-specific system-wide configuration files. There has been controversy over the meaning of the name itself. In early versions of the UNIX Implementation Document from Bell labs, /etc is referred to as the *etcetera directory*, as this directory historically held everything that did not belong elsewhere (however, the FHS restricts /etc to static configuration files and may not contain binaries). Since the publication of early documentation, the directory name has been re-explained in various ways. Recent interpretations include acronyms such as "Editable Text Configuration" or "Extended Tool Chest."
/etc/opt	Configuration files for add-on packages that are stored in /opt .
/etc/sgml	Configuration files, such as catalogs, for software that processes SGML.
/etc/X11	Configuration files for the X Window System, version 11.
/etc/xml	Configuration files, such as catalogs, for software that processes XML.
/home	Users' home directories, containing saved files, personal settings, etc.
/lib	Libraries essential for the binaries in /bin and /sbin .
/lib<qual>	Alternative format essential libraries. Such directories are optional, but if they exist, they have some requirements.
/media	Mount points for removable media such as CD-ROMs (appeared in FHS-2.3 in 2004).
/mnt	Temporarily mounted filesystems.
/opt	Optional application software packages.
/proc	Virtual filesystem providing process and kernel information as files. In Linux, corresponds to a procfs mount. Generally automatically generated and populated by the system, on the fly.
/root	Home directory for the root user.
/run	Run-time variable data: Information about the running system since last boot, *e.g.*, currently logged-in users and running daemons. Files under this directory must be either removed or truncated at the beginning of the boot process, but this is not necessary on systems that provide this directory as a temporary filesystem (tmpfs).
/sbin	Essential system binaries, *e.g.*, fsck, init, route.
/srv	Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems (appeared in FHS-2.3 in 2004).
/sys	Contains information about devices, drivers, and some kernel features.
/tmp	Temporary files (see also /var/tmp). Often not preserved between system reboots, and may be severely size restricted.
/usr	*Secondary hierarchy* for read-only user data; contains the majority of (multi-)user utilities and applications.
/usr/bin	Non-essential command binaries (not needed in single user mode); for all

	users.
/usr/include	Standard include files.
/usr/lib	Libraries for the binaries in /usr/bin and /usr/sbin.
/usr/lib<qual>	Alternative format libraries, *e.g.* /usr/lib32 for 32-bit libraries on a 64-bit machine (optional).
/usr/local	*Tertiary hierarchy* for local data, specific to this host. Typically has further subdirectories, *e.g.*, bin , lib , share .
/usr/sbin	Non-essential system binaries, *e.g.*, daemons for various network-services.
/usr/share	Architecture-independent (shared) data.
/usr/src	Source code, *e.g.*, the kernel source code with its header files.
/usr/X11R6	X Window System, Version 11, Release 6 (up to FHS-2.3, optional).
/var	Variable files—files whose content is expected to continually change during normal operation of the system—such as logs, spool files, and temporary e-mail files.
/var/cache	Application cache data. Such data are locally generated as a result of time-consuming I/O or calculation. The application must be able to regenerate or restore the data. The cached files can be deleted without loss of data.
/var/lib	State information. Persistent data modified by programs as they run, *e.g.*, databases, packaging system metadata, etc.
/var/lock	Lock files. Files keeping track of resources currently in use.
/var/log	Log files. Various logs.
/var/mail	Mailbox files. In some distributions, these files may be located in the deprecated /var/spool/mail .
/var/opt	Variable data from add-on packages that are stored in /opt .
/var/run	Run-time variable data. This directory contains system information data describing the system since it was booted. In FHS 3.0, /var/run is replaced by /run ; a system should either continue to provide a /var/run directory, or provide a symbolic link from /var/run to /run , for backwards compatibility.
/var/spool	Spool for tasks waiting to be processed, *e.g.*, print queues and outgoing mail queue.
/var/spool/mail	Deprecated location for users' mailboxes.
/var/tmp	Temporary files to be preserved between reboots.

In a pictorial manner, here is what the filesystem looks like. Some of these directories have standard files as well - /bin, /etc and /sbin specifically.

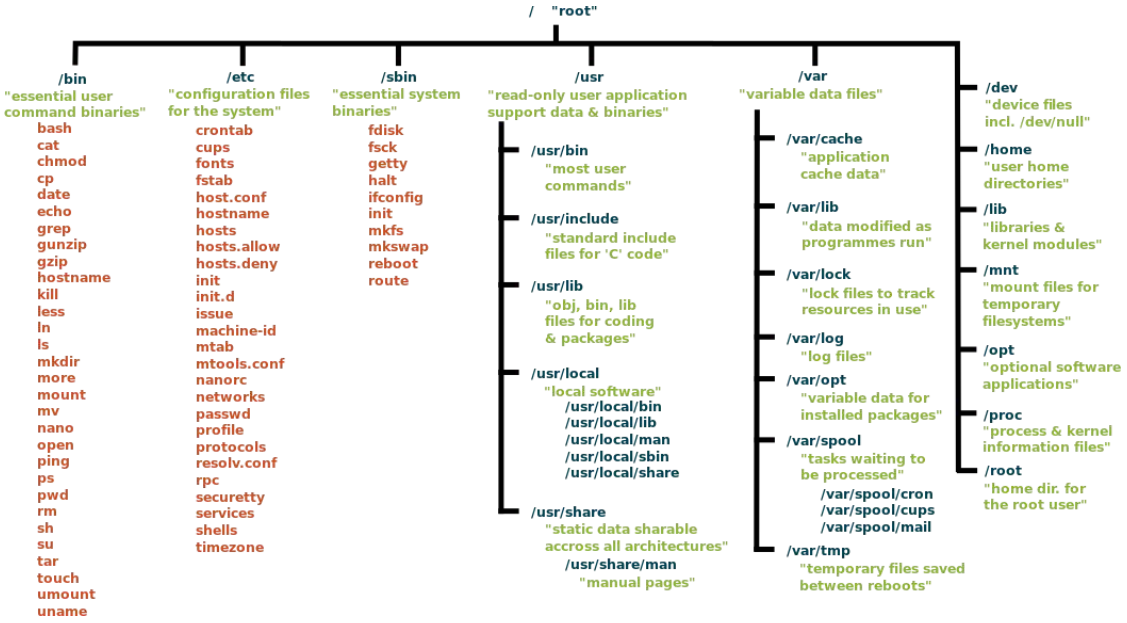


Figure **4.12.1.1**: Standard-unix-filesystem-hierarchy. ("Standard-unix-filesystem-hierarchy.svg" by Ppgardne, Wikimedia Commons is licensed under CC BY-SA 4.0)

Adapted from:

"Filesystem Hierarchy Standard" by Multiple contributors, Wikipedia is licensed under CC BY-SA 4.0

04-E.12.1: Linux Directory Structure - Hierarchy is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-E.12.2: Linux Directory Structure - Home Directory

Special Linux Directory Concepts

Home Directory

A user's home directory is intended to contain that user's files, including text documents, music, pictures or videos, etc. It may also include their configuration files of preferred settings for any software they have used there and might have tailored to their liking: web browser bookmarks, favorite desktop wallpaper and themes, passwords to any external services accessed via a given software, etc. The user can install executable software in this directory, but it will only be available to users with permission to this directory. The home directory can be organized further with the use of sub-directories.

Be aware - the home directory for the root user is /root. Do not confuse /root with the system root which is /, the top of the filesystem.

Most shells refer to a user's home directory with the ~ (the tilde).

A partial listing of a user's home directory:

```
santiago@pbmac-server $ ls -al
total 136
drwxr-xr-x 11 santiago santiago  4096 Jun  4 15:10 .
drwxr-xr-x  5 root     root      4096 Jun  4 14:21 ..
-rw-------  1 santiago santiago  2027 Jun  5 16:42 .bash_history
-rw-r--r--  1 santiago santiago   220 Apr 18 16:30 .bash_logout
-rw-r--r--  1 santiago santiago  3802 Apr 23 16:22 .bashrc
drwxrwxr-x  2 santiago santiago  4096 Jun  4 14:18 Desktop
drwxrwxr-x  2 santiago santiago  4096 Jun  4 14:18 Documents
drwxrwxr-x  2 santiago santiago  4096 Jun  4 14:18 Downloads
```

We will talk more about what all of this means in great detail in the next Module.

Next we will go over some concepts that we will need to deal with regarding the filesystem.

Current Working Directory

The concept of current directory refers to the directory you are located in on the system. When a user logs into the system their current working directory (cwd) is their home directory. As you move around on the system your cwd changes.

The cwd is represented by a single period. Remember when we talked about naming files we stated that . and .. have special meaning? So, . is always the cwd. You can always find your cwd using the Linux **pwd** command. There is also an environmental variable, PWD, that maintains the state of your cwd.

```
pbmac@pbmac-server $ echo $PWD
/home/santiago
pbmac@pbmac-server $ pwd
/home/santiago
pbmac@pbmac-server $ ls -l xyz.txt
---------- 1 santiago santiago 99 Jun  4 14:19 xyz.txt
pbmac@pbmac-server $ ls -l ./xyz.txt
---------- 1 santiago santiago 99 Jun  4 14:19 ./xyz.txt
pbmac@pbmac-server $ cd ~
pbmac@pbmac-server $ pwd
```

```
/home/pbmac
pbmac@pbmac-server $ echo $PWD
/home/pbmac
```

In the output: we echo the environment variable (Yes - the $ is required); using the **pwd** command we print the working directory; we list the file xyz.txt and then print out "here"/xyz.txt - because . is "this directory"; then using tilde we change directories with the **cd** command to my home directory; again use both the **pwd** command and echo the PWD environment variable.

Parent Directory

Every directory, except the / directory, has a parent directory. For my home directory, /home/pbmac, the parent directory is /home. It is simply the directory that is one level above the current directory. The double dot notation .. refers to the parent directory.

```
pbmac@pbmac-server $ pwd
/home/pbmac
pbmac@pbmac-server $ cd ..
pbmac@pbmac-server $ pwd
/home
```

Adapted from:
"Home directory`" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

04-E.12.2: Linux Directory Structure - Home Directory is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-E.12.3: Linux Directory Structure - Paths

Linux Paths

Linux organizes files in a hierarchical tree, where relationships are thought of in teams of children and parent. Directories can contain other directories as well as regular files, which are the "leaves" of the tree. Any element of the tree can be referenced by a path name; an **absolute path name starts with the character** / (identifying the root directory, which contains all other directories and files), then every child directory that must be traversed to reach the element is listed, each separated by a / sign.

A **relative path name is one that doesn't start with** / ; in that case, the directory tree is traversed starting from a given point (which changes depending on context) called the current directory. In every directory, there are two special directories called . and .. , which refer respectively to the directory itself, and to its parent directory.

The fact that all files and directories have a common root means that, even if several different storage devices are present on the system, they are all seen as directories somewhere in the tree, once they are mounted to the desired place.

An absolute path name, pointing to what is normally an executable file on a Linux system:

```
/usr/bin/test
```

This represents the file in the /usr/bin directory.

An absolute path name, but pointing to a directory instead of a regular file:

```
/usr/bin/
```

Notice it ENDS with a /, thereby specifying the specified file is indeed a directory.

A relative path name, which will point to /usr/bin/test only if the current directory is /usr/ :

```
pbmac@pbmac-server $ pwd
/usr
pbmac@pbmac-server $ ls bin/test
bin/test
```

We are already sitting in the /usr directory, so an **`ls`** command can use a relative path to list the file **`test`** in the subdirectory bin.

A relative path name, which will point to /usr/bin/test if the current directory is any directory in /usr/ , for instance /usr/share/ :

```
pbmac@pbmac-server $ pwd
/usr/local
pbmac@pbmac-server $ ls ../bin/test
../bin/test
```

We start with our current location in the /usr/local directory. We use the .. notation to tell our ls command to look up one directory, to /usr, and from there look in the bin subdirectory at the **`test`** command file.

A path name using the special shortcut ~ , which refers to the current user's home directory:

```
pbmac@pbmac-server $ cd ~/Desktop/
```

No matter what your current directory is, this takes you to the Desktop directory in your home directory (because we used the tilde).

Path names can contain almost any character, but some characters, such as **space** must be escaped in most software, usually by enclosing the name in quotation marks:

```
"~/Examples/Experience ubuntu.ogg"
```

or by employing the escape character \ in front of the space:

```
~/Examples/Experience\ ubuntu.ogg
```

Command Review

We have talked about several Linux commands:

`cd`: The **`cd`** command will allow you to change directories. When you open a terminal you will be in your home directory. To move around the file system you will use **`cd`** in conjunction with absolute or relative path name.

`pwd`: The **`pwd`** command will allow you to know in which directory you're located (**`pwd`** stands for "print working directory").

`ls`: The **`ls`** command will show you ('list') the files in your current directory, or the directory specified with absolute/relative path names. With certain options, you can see sizes of files, when files were made, and permissions of files.

Adapted from:
"Linux Filesystem Tree Overview" by Edwin Khoo, Ubuntu Community is licensed under CC BY-SA 4.0
"Using The Terminal" by Chris Perry, Ubuntu Community is licensed under CC BY-SA 4.0

04-E.12.3: Linux Directory Structure - Paths is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-F.13: Storage Issues - Find and Fix

Troubleshooting Storage Device Issues

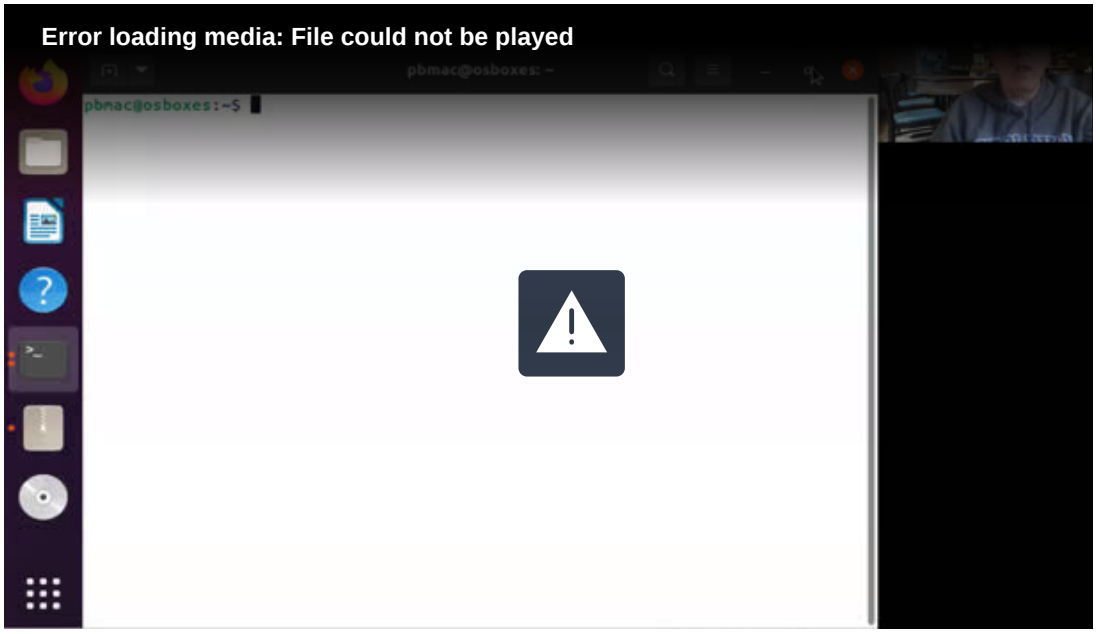
We have covered a lot of details in regards to setting up storage systems. With something this complex, it is not unheard of to have problems. There are numerous situations that may arise as we add and configure storage capacity to a system. It might be that the device isn't showing up on the system, or perhaps volumes that were created are not showing up. Some of the configuration files may have been improperly edited, and contain errors - this has to be found and resolved.

To troubleshoot issues it is a good thing to go back and verify each step, to ensure that all the steps actually accomplished the necessary task.

When troubleshooting the various storage issues that present themselves there is a process to follow:

- Be sure the devices are physically connected to the system. Check that cables, both communication cables and power cables, are securely attached on both ends. Perhaps test your cables to ensure they are working.
- Ensure the devices are powered on. Is the power cable good? Is it securely connected on both ends? Check for breaks in the wiring - use a tester if possible.
- Ensure the devices are turned on, if applicable. Some devices have a power switch that must be turned on prior to use.
- Verify the system recognizes the device by checking the /proc/ directory. Use either the fdisk command or the parted to verify that the partitions are present on the disk you think they are on.
- Confirm that the configuration files do not contain any typographical errors. Files such as the /etc/fstab are very particular about entries that are made or edited. Be sure that any edits you made were correct.
- Reload any configuration files that have had changes made to them.
- Ensure that there is enough storage capacity for what you are attempting to do.
- Confirm that the I/O workload is not overwhelming the device. Other problems such as a failed RAID device might also be causing issues with I/O performance.
- Use the partprobe command to cause the system to scan for new storage devices and partitions.
- Check system settings to make sure that you are not reaching limits imposed by system configuration.

The video covers some of the storage system managment commands that you need to properly manage your Linux system.



Storage System Configuration and Monitoring

Linux provides a set of tools that can be used to assist us in finding issues and resolving them, allowing the system to function at the most proficient level.

The `ulimit` Command

This command is actually built into the bash shell, so you won't find a **`ulimit`** man page on most distributions. It is often addressed in the bash man page; you will have to search for **`ulimit`** and the options are listed there.

Using the **ulimit** command with the -a option shows you all of the current user's limits:

```
pbmac@pbmac-server $ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 47525
max locked memory       (kbytes, -l) 16384
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 47525
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Before you start changing your limits make sure you know what the limit refers to and how the value is interpreted.

The **du** Command

du summarizes disk usage of the set of FILEs, recursively for directories.

Syntax:

```
du [ OPTION ] [ FILE ]
```

Specifying a FILE on the command line is optional with the du command.

Command Options:

Optoins	Option Meaning
-0, –null	End each output line with NULL.
-a, –all	Write count of all files, not just directories.
–apparent-size	Print apparent sizes, rather than disk usage.
-B, –block-size=SIZE	Scale sizes to SIZE before printing on console.
-c, –total	Produce grand total.
-d, –max-depth=N	Print total for directory only if it is N or fewer levels below command line argument.
-h, –human-readable	Print sizes in human readable format.
-S, -separate-dirs	For directories, don’t include size of subdirectories.
-s, –summarize	Display only total for each directory.
–time	Show time of last modification of any file or directory.
–exclude=PATTERN	Exclude files that match PATTERN.

A simple example, specifying a file/directory, and the output shows the size of that directory, including all sub-directories:

```
pbmac@pbmac-server $ du -hs ~/Documents/
311M    /home/pbmac/Documents/
```

Without the -s option it would list each file recursively from this point of the file system, specifying the file name and the size of the file. Other options can be found on the man page.

The `df` Command

`df` displays the amount of disk space available on the file system containing each file name argument. If no file name is given, the space available on all currently mounted file systems is shown. Disk space is shown in 1K blocks by default, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used.

Syntax:

```
df [ OPTION ] [FILE]
```

The file name is optional when using the df command. It is also permissible to enter multiple file names.

Command Options:

Options	Option Meaning
-a, –all	Includes pseudo, duplicate and inaccessible file systems.
-B, –block-size=SIZE	Scales sizes by SIZE before printing them.
-h, –human-readable	Print sizes in power of 1024.
-H, –si	Print sizes in power of 1000.
-i, –inodes	List inode information instead of block usage.
-l, –local	Limit listing to local file systems.
-P, –portability	Use POSIX output format.
–sync	Invoke sync before getting usage info.
–total	Elide all entries insignificant to available space, and produce grand total.
-t, –type=TYPE	Limit listing to file systems of type TYPE.
-T, –print-type	Print file system type

```
pbmac@pbmac-server $ df /home
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/sda6      310238080 125360048 169095788  43% /home

No options...defaults to all devices
pbmac@pbmac-server $ df
/dev/sda1        8940016   5730240   2732592  68% /
/dev/sda2       80503616  11943000  64448232  16% /usr
/dev/sda6      310238080 133552864 160902972  46% /home
/dev/sda5       80503616  24386528  52004704  32% /var
```

The output shows which file system this mount point is on, how many 1K blocks are in this partition, how many are used, how many are yet available and the percent of use, as well as the mount point of this file system.

Adapted from:
"du command in Linux with examples" by msdeep14, Geeks for Geeks is licensed under CC BY-SA 4.0
"df command in Linux with Examples" by msdeep14, Geeks for Geeks is licensed under CC BY-SA 4.0

04-F.13: Storage Issues - Find and Fix is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-F.13.1: Storage Issues - IO Scheduling

Linux I/O schedulers

I/O schedulers attempt to improve throughput by reordering request access into a linear order based on the logical addresses of the data and trying to group these together. While this may increase overall throughput it may lead to some I/O requests waiting for too long, causing latency issues. I/O schedulers attempt to balance the need for high throughput while trying to fairly share I/O requests amongst processes.

Different approaches have been taken for various I/O schedulers and each has their own set of strengths and weaknesses; the general rule is that there is no perfect default I/O scheduler for all the range of I/O demands a system may experience.

Multiqueue I/O schedulers

Note: These are the only I/O schedulers available in certain distributions of Linux as they are supported in the Linux 5.3 kernel.

The following I/O schedulers are designed for multiqueue devices. These map I/O requests to multiple queues, and are handled by kernel threads that are distributed across multiple CPUs.

bfq (Budget Fair Queuing) (Multiqueue)

Designed to provide good interactive response, especially for slower I/O devices. This is a complex I/O scheduler and has a relatively high per-operation overhead so it is not ideal for devices with slow CPUs or high throughput I/O devices. Fair sharing is based on the number of sectors requested and heuristics rather than a time slice. Desktop users may like to experiment with this I/O scheduler as it can be advantageous when loading large applications.

kyber (Multiqueue)

Designed for fast multi-queue devices and is relatively simple. Has two request queues:

- Synchronous requests (e.g. blocked reads)
- Asynchronous requests (e.g. writes)

There are strict limits on the number of request operations sent to the queues. In theory this limits the time waiting for requests to be dispatched, and hence should provide quick completion time for requests that are high priority.

none (Multiqueue)

The multi-queue no-op I/O scheduler. Does no reordering of requests, minimal overhead. Ideal for fast random I/O devices such as NVME.

mq-deadline (Multiqueue)

This is an adaption of the deadline I/O scheduler but designed for Multiqueue devices. A good all-rounder with fairly low CPU overhead.

Non-multiqueue I/O schedulers

NOTE: Non-multiqueue have been deprecated in some distributions of Linux onwards as they are no longer supported in the Linux 5.3 kernel.

deadline

This fixes starvation issues seen in other schedulers. It uses 3 queues for I/O requests:

- Sorted
- Read FIFO - read requests stored chronologically
- Write FIFO - write requests stored chronologically

Requests are issued from the sorted queue unless a read from the head of a read or write FIFO expires. Read requests are preferred over write requests. Read requests have a 500ms expiration time, write requests have a 5s expiration time.

cfq (Completely Fair Queuing)

- Per-process sorted queues for synchronous I/O requests.
- Fewer queues for asynchronous I/O requests.
- Priorities from ionice are taken into account.

Each queue is allocated a time slice for fair queuing. There may be wasteful idle time if a time slice quantum has not expired.

noop (No-operation)

Performs merging of I/O requests but no sorting. Good for random access devices (flash, ramdisk, etc) and for devices that sort I/O requests such as advanced storage controllers.

Adapted from:
"IOSchedulers" by Colin Ian King, Ubuntu Wiki is licensed under CC BY-SA 3.0

04-F.13.1: Storage Issues - IO Scheduling is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-F.13.2: Storage Issues - More IO Scheduling

Selecting I/O Schedulers

Prior to Ubuntu 19.04 with Linux 5.0 or Ubuntu 18.04.3 with Linux 4.15, the multiqueue I/O scheduling was not enabled by default, and just the deadline, cfq and noop I/O schedulers were available by default.

For Ubuntu 19.10 with Linux 5.0 or Ubuntu 18.04.3 with Linux 5.0 onwards, multiqueue is enabled by default providing the bfq, kyber, mq-deadline and none I/O schedulers. For Ubuntu 19.10 with Linux 5.3 the deadline, cfq and noop I/O schedulers are deprecated.

With the Linux 5.0 kernels, one can disable these and fall back to the non-multiqueue I/O schedulers using a kernel parameter. For example for SCSI devices one can use:

```
scsi_mod.use_blk_mq=0
```

Add this to the `GRUB_CMDLINE_LINUX_DEFAULT` string in `/etc/default/grub` and run `sudo update-grub` to enable this option.

Changing an I/O scheduler is performed on a per block device basis. For example, for non-multi queue device /dev/sda one can see the current I/O schedulers available using the following:

```
pbmac@pbmac-server $ cat /sys/block/sda/queue/scheduler
noop deadline [cfq]
```

To change this to deadline use:

```
pbmac@pbmac-server $ echo "deadline" | sudo tee /sys/block/sda/queue/scheduler
```

For multiqueue devices the default will show:

```
pbmac@pbmac-server $ cat /sys/block/sda/queue/scheduler
[mq-deadline] none
```

To use kyber, install the module:

```
pbmac@pbmac-server $ sudo modprobe kyber-iosched
pbmac@pbmac-server $ cat /sys/block/sda/queue/scheduler
[mq-deadline] kyber none
```

and enable it:

```
pbmac@pbmac-server $ echo "kyber" | sudo tee /sys/block/sda/queue/scheduler
```

To use bfq, install the module:

```
pbmac@pbmac-server $ sudo modprobe bfq
pbmac@pbmac-server $ cat /sys/block/sda/queue/scheduler
[mq-deadline] kyber none
```

and enable it:

```
pbmac@pbmac-server $ echo "bfq" | sudo tee /sys/block/sda/queue/scheduler
```

Tuning I/O Schedulers

Each I/O scheduler has a default set of tunable options that may be adjusted to help improve performance or fair sharing for your particular use case. The following kernel documentation covers these per-I/O scheduler tunable options:

- deadline (and mq-deadline) deadline-iosched.txt
- cfq cfq-iosched.txt
- bfq bfq-iosched.txt
- kyber kyber-iosched.txt

Best I/O scheduler to use

Different I/O requirements may benefit from changing from the Ubuntu distro default. A quick start guide to select a suitable I/O scheduler is below. The results are based on running 25 different synthetic I/O patterns generated using fio on ext4, xfs and btrfs with the various I/O schedulers using the 5.3 kernel.

SSD or NVME drives

It is worth noting that there is little difference in throughput between the mq-deadline/none/bfq I/O schedulers when using fast multi-queue SSD configurations or fast NVME devices. In these cases it may be preferable to use the 'none' I/O scheduler to reduce CPU overhead.

HDD

Avoid using the none/noop I/O schedulers for a HDD as sorting requests on block addresses reduce the seek time latencies and neither of these I/O schedulers support this feature. mq-deadline has been shown to be advantageous for the more demanding server related I/O, however, desktop users may like to experiment with bfq as it has been shown to load some applications faster.

Of course your use-case may differ - the above are just suggestions to start with based on some synthetic tests. You may find other choices with adjustments to the I/O scheduler tunables produce better results.

Adapted from:
"IOSchedulers" by Colin Ian King, Ubuntu Wiki is licensed under CC BY-SA 4.0

04-F.13.2: Storage Issues - More IO Scheduling is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-F.13.3: Storage Issues - iostat, ioping

Tools for Storage Issues

The `iostat` Command

The **`iostat`** command is used for monitoring system input/output device loading by observing the time the devices are active in relation to their average transfer rates. The **`iostat`** command generates reports that can be used to change system configuration to better balance the input/output load between physical disks. This tool is not installed on most Linux distros, so you will need to determine how to install this package for your particular distribution.

Syntax:

```
iostate [ OPTION ] [ device ]
```

Command Options:

Options	Option Meaning
-c	Show only the cpu statistic.
-d	Display only the device report.
-j ID mmcbkl0 sda6 -x -m 2 2	Display persistent device name statistics.
-k	Capture the statistics in kilobytes or megabytes.
-k 2 3	Display cpu and device statistics with delay.
-N	Display lvm2 statistic information.
-p	Display statistics for block devices.
-x	Show more detailed statistics information.
-xd	Show extended I/O statistic for device only.

```
pbmac@pbmac-server $ iostat
Linux 4.15.0-91-generic (pbmac-server)     06/20/2020     _x86_64_    (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           9.47    0.02    1.66    1.89    0.00   86.95

Device             tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda              10.46       182.70       120.75  275705960  182212856
```

The first section contains CPU report:

- **%user :** Shows the percentage of CPU utilization that occurred while executing at the user level.
- **%nice :** Shows the percentage of CPU utilization that occurred while executing at the user level with a nice priority.
- **%system :** Shows the percentage of CPU utilization that occurred while executing at the system (kernel) level.
- **%iowait :** Shows the percentage of the time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.
- **%steal :** Shows the percentage of time being spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing by another virtual processor.

- **%idle :** Shows the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.

The second section of the output contains device utilization report:

- **Device :** The device/partition name is listed in /dev directory.
- **tps :** The number of transfers per second that were issued to the device. Higher tps means the processor is busier.
- **kB_read/s :** Shows the amount of data read from the device expressed in a number of blocks (kilobytes, megabytes) per second.
- **kB_wrtn/s :** The amount of data written to the device expressed in a number of blocks (kilobytes, megabytes) per second.
- **kB_read :** Shows the total number of blocks read.
- **kB_wrtn :** Shows the total number of blocks written.

The `ioping` Command

The **`ioping`** command is a tool to report disk latency — how long it takes a disk to respond to requests. It is of great use in diagnosing disk problems.

```
pbmac@pbmac-server $ sudo ioping -c 5 /dev/sda5
4 KiB <<< /dev/sda5 (block device 78.1 GiB): request=1 time=16.4 ms (warmup)
4 KiB <<< /dev/sda5 (block device 78.1 GiB): request=2 time=11.3 ms
4 KiB <<< /dev/sda5 (block device 78.1 GiB): request=3 time=17.9 ms
4 KiB <<< /dev/sda5 (block device 78.1 GiB): request=4 time=10.9 ms
4 KiB <<< /dev/sda5 (block device 78.1 GiB): request=5 time=11.6 ms

--- /dev/sda5 (block device 78.1 GiB) ioping statistics ---
4 requests completed in 51.7 ms, 16 KiB read, 77 iops, 309.4 KiB/s
generated 5 requests in 4.01 s, 20 KiB, 1 iops, 4.99 KiB/s
min/avg/max/mdev = 10.9 ms / 12.9 ms / 17.9 ms / 2.87 ms
```

Check out the man page for other options for this command. This tool is not installed on most Linux distros, so you will need to determine how to install this package for your particular distribution.

Adapted from:
"iostat command in Linux with examples" by rahulkumarmandal, Geeks for Geeks is licensed under CC BY-SA 4.0

04-F.13.3: Storage Issues - iostat, ioping is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

04-F.13.4: Storage Issues - Quota Management

Disk Quota

This feature of Linux allows the system administrator to allocate a maximum amount of disk space a user or group may use. It can be flexible in its adherence to the rules assigned and is applied per filesystem.

Set Up the filesystem

Edit **`fstab`** to enable the quota mount option(s) on selected file systems, e.g.:

```
pbmac@pbmac-server $ /etc/fstab
/dev/sda3 /home ext4 defaults,usrquota 0 2
```

To additionally enable the group quota mount option:

```
pbmac@pbmac-server $ /etc/fstab
/dev/sda3 /home ext4 defaults,usrquota,grpquota 0 2
```

If supported by the kernel and file system it is recommended to use journaled quota instead:

```
pbmac@pbmac-server $ /etc/fstab
/dev/sda3 /home ext4 defaults,usrjquota=aquota.user,jqfmt=vfsv1 0 2
```

Append **`grpjquota=aquota.group`** to enable group quota.

Remount the partition to apply the change:

```
 pbmac@pbmac-server # mount -vo remount /home
```

Create Quota Index

To create the quota index for **`/home`**:

```
 pbmac@pbmac-server # quotacheck -cum /home
```

Append the **`-g`** parameter to also create a group index.

To enable disk quotas for the desired file system:

```
pbmac@pbmac-server # quotaon -v /home
```

To disable disk quotas for the file system:

```
pbmac@pbmac-server # quotaoff -v /home
```

Enable Quota for user/group

Tip:

- To find out how many 1 kilobyte blocks there are for a partition use **`df`** command.
- You may use an online bytes converter to calculate the correct amount of blocks.
- The command **`setquota`** may be used as an alternative of **`edquota`** .

Note: Block size is statically set to 1k regardless of filesystem block size.

Quotas are configured using **`edquota`** (as the root user) that will be opened in the default configured text editor:

```
pbmac@pbmac-server # edquota user
Disk quotas for user  user  (uid 1000):
  Filesystem                   blocks       soft       hard     inodes     soft     ha
  /dev/sda3                        24          0          0          6        0
```

```
blocks -   Indicates number of 1k blocks currently used by the user/group.
soft -   Indicates max number of blocks for the user/group before a warning is issued
hard -   Indicates max number of blocks for the user/group can use. If maximum amount
inodes -   Indicates the current inodes amount used by the user/group.
soft -   Indicates the soft inode limit for the user/group.
hard -   Indicates the hard inode limit for the user/group.
```

```
pbmac@pbmac-server # edquota ftpuser1
Disk quotas for user ftpuser1 (uid  1000 ):
  Filesystem                   blocks       soft       hard     inodes     soft     ha
  /dev/sda3                        24    1000000    1048576          6        0
```

In this case if ftpuser1 uses over 976MB of space a warning will be issued. If the hard limit of 1GB has been reached the user will be unable to write any more data.

Warning: The **`hard`** limit applies to all files written by and for the respective user/group, including temporary files by started applications, which may crash at this point.

Specify a Grace Period

To give current users some time to reduce their file usage, a grace period can be configured. This specifies the allowed time a user/group can exceed their soft limit and while under their hard limit:

```
pbmac@pbmac-server # edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem             Block grace period     Inode grace period
  /dev/sda3              7days                  7days
```

The grace period can be set in seconds, minutes, hours, days, weeks or months.

Reports

Shows all configured quotas:

```
pbmac@pbmac-server # repquota -a
```

Shows quotas on a specific partition:

```
pbmac@pbmac-server # repquota  /home
```

Shows quotas that apply to a **user**/**user group**:

```
pbmac@pbmac-server # quota -u  user
```

```
pbmac@pbmac-server # quota -g  group
```

Copy Quota Settings

To one or several users. To copy quota settings from **`user1`** to **`user2`**:

```
pbmac@pbmac-server # edquota -p  user1   user2
```

To copy quota settings to several other users, append **`user3`** **`user4`** ...

To groups

To copy quota settings from **`group1`** to **`group2`**:

```
pbmac@pbmac-server # edquota -g -p  group1   group2
```

To All Users

The idea is to modify the quota settings for one user and copy the setting to all other users. Set the quota for **`user1`** and apply the quota to users with a UID greater than 999:

```
pbmac@pbmac-server # edquota -p  user1  $(awk -F: '$3 > 999 {print $1}' /etc/passwd)
```

Quota Warnings

The command **`warnquota`** can be used to warn the users about their quota. Configuration is available in **`/etc/warnquota.conf`**.

Stats

The command **`quotastats`** can be used to give more information about the current quota usage:

```
pbmac@pbmac-server $ quotastats<
Number of dquot lookups: 101289
Number of dquot drops: 101271
Number of still active inodes with quota : 18
Number of dquot reads: 93
Number of dquot writes: 2077
Number of quotafile syncs: 134518740
Number of dquot cache hits: 7391
Number of allocated dquots: 90
Number of free dquots: 2036
Number of in use dquot entries (user/group): -1946
```

Adapted from:
"Disk quota" by Multiple contributors, ArchLinux is licensed under CC BY-SA 4.0

04-F.13.4: Storage Issues - Quota Management is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 05: File and Directory Management

- 05-A.1: Creating and Editing Text files
- 05-A.2: Creating and Editing Text files - Vim Editor
  - 05-A.2.1: Vim - Insert Mode
  - 05-A.2.2: Vim - Visual Mode
  - 05-A.2.3: Vim - Command-Line Mode
- 05-A.3: Nano Editor
- 05-A.4: Gedit and More
- 05-B.5: Searching for Files on Linux
  - 05-B.5.1: Searching for Files on Linux - find Command
  - 05-B.5.2: Searching for Files on Linux - which Command
  - 05-B.5.3: Searching for Files on Linux - whereis Command
- 05-C.6: Working with Files and Directories - cat Command
  - 05-C.6.1: Working with Files and Directories - head/tail Commands
  - 05-C.6.2: Working with Files and Directories - less/more Command
  - 05-C.6.3: Working with Files and Directories - cp/mv Command
  - 05-C.6.4: Working with Files and Directories - rm/touch Command
  - 05-C.6.5: Working with Files and Directories - ls/rmidir/mkdir Command
- 05-D.7: Handling Text Files - echo/printf
  - 05-D.7.1: Handling Text Files - tr/wc Command
  - 05-D.7.2: Handling Text Files - sort/diff Commands
  - 05-D.7.3: Handling Text Files - cut/paste
  - 05-D.7.4: Handling Text Files - grep Command
  - 05-D.7.5: Handling Text Files - awk Command
  - 05-D.7.6: Handling Text Files - sed Command
  - 05-D.7.7: Handling Text Files - ln Command
- 05-E.8: File Output Manipulation
  - 05-E.8.1: File Output Manipulation - pipe
  - 05-E.8.2: File Output Manipulation - xargs
  - 05-E.8.3: File Output Manipulation - tee & /dev/null

Module 05: File and Directory Management is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-A.1: Creating and Editing Text files

EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

Module Introduction

We have covered quite a bit of ground already in this course. Now it is time to begin the managing of data itself. The management of files and directories allows us to quickly create and retrieve process data so that it provides the information we will need to benefit our company, our users and ourselves as we serve as system administrator for our Linux systems.

Module Objectives

In this lesson, we will:

- Learn to create and edit text files.
- Understand how to search for files on a Linux system.
- Execute various operations on files and directories. We will view, copy, and delete files.
- Manage text files to make it easier to analyze and use.
- Manipulate file output through redirection, piping, etc.

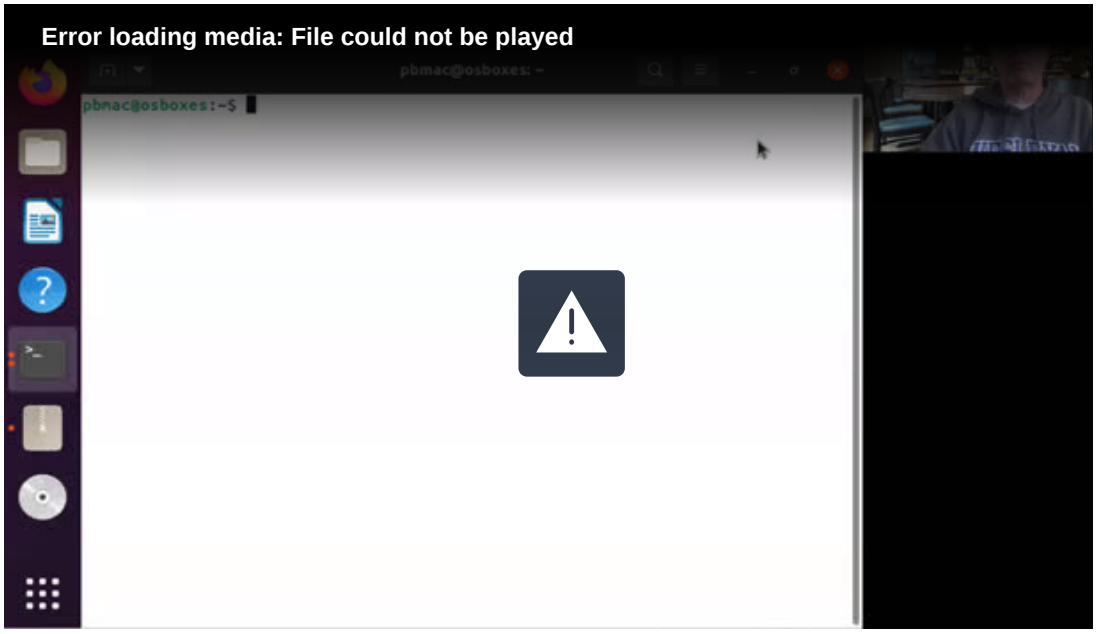
Text Editors

Many of us are familiar with some form of a text editor. Windows Notepad is probably the simplest text editor available. A text editor provides the simple ability to create, edit and view text documents. In the world of Linux administration we find text editors useful because most of the files that we have to modify or view are text files. The reason we don't use a word processor is that word processors introduce various formats for their resulting files; they are not text files.

Some of the more traditional Linux text editors:

- vi / Vim / gVim - these are all the same family of editors. vi was the original tool, then later editions became Vim (vi iMproved). Vim was the default editor of Linux, while gVim is a graphical version of Vim.
- GNU nano - a very simple text editor found in Linux. There is not a graphical version of nano.
- gedit - this editor runs in a graphical interface in the Linux environment.
- Emacs - a LONG time part of Unix and Linux. It is a powerful and very flexible tool that can be configured to handle a variety of tasks.

We will take a brief look at a couple of the editors over the next several pages.



This page titled 05-A.1: Creating and Editing Text files is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

05-A.2: Creating and Editing Text files - Vim Editor

Vim Editor

`Vim` offers more modes than the original vi (which offers only the "normal", "insert" and "command–line" modes). These additional modes make **`vim`** more powerful and easier to use; because of this, **`vim`** users should at least be aware that they exist. (NOTE: *If you ever enter a mode you are unfamiliar with, you can usually press **ESC** to get back to* normal *mode.*)

Here is a short overview of each mode available in **`vim`**:

Mode Name	Description	help page
normal	For navigation and manipulation of text. This is the mode that vim will usually start in, which you can usually get back to with **ESC**.	`:help Normal-mode`
insert	For inserting new text. The main difference from vi is that many important "normal" commands are also available in insert mode - provided you have a keyboard with enough meta keys (such as Ctrl, Alt, Windows-key, etc.).	`:help Insert-mode`
visual	For navigation and manipulation of text selections, this mode allows you to perform most normal commands, and a few extra commands, on selected text.	`:help Visual-mode`
command-line	For entering editor commands - like the help commands in the third column.	`:help Command-line-mode`

Each mode is described below.

Normal Mode

Open a console terminal from your Linux operating system, and enter **vi .**

```
pbmac@pbmas-server $ vi
```

You are now in Normal mode - also referred to as Command mode, because we enter commands from this mode. This is where **`vim`** normally starts. If you have text, you can move around with your arrow keys or other navigation keystrokes (which you will see later). To make sure you are in Normal mode, simply hit the **Esc** (Escape) key.

Tip: Esc switches to Normal mode. Even though you are already in Normal mode, hit **Esc** just for practice's sake. This is what a new file would look like; if you opened an existing file you would see the content of the file.

```
~
~
~
~
:q!
```

Now, this will be interesting. Press **:** (the colon key) followed by **q!** (i.e., **:q!**). Your screen will look like this:

Pressing the colon in Normal mode switches **`vim`** to Command Line mode, and the **:q!** command quits (q) without saving (!). In other words, you are abandoning all changes. You can also use **ZQ** - choose whichever option is more convenient.

A brief look at the other modes continues in the following pages.

Adapted from:
"Learning the vi Editor/Vim/Modes" by Multiple contributors, WikiBooks is licensed under CC BY-SA 3.0

05-A.2: Creating and Editing Text files - Vim Editor is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-A.2.1: Vim - Insert Mode

Insert Mode

In insert mode you can type new text. In classic vi the insert mode was just that: insert text and nothing else. **`Vim`** makes use of many meta keys on modern keyboards; with a correctly configured **`vim`**, cursor keys should work in insert mode.

Insert mode can be reached in several ways, but some of the most common ones are as simple as entering a keystroke:

- i – New text will appear **before** the cursor.
- a – New text will appear **after** the cursor.
- I – New text will appear at the **beginning of the current line** - cursor is moved to the beginning of the current line.
- A – Next text will appear at the **end of the current line** - cursor is moved to the end of the current line.
- o – A new line is created **after the current line** - cursor is placed at the beginning of the new line
- O – A new line is created **before the current line** - cursor is placed at the beginning of the new line

You can enter any text you desire. Be aware that, by default, there is no autowrap - **`vim`** will let you enter as long as you keep typing. There is a configuration setting that will help you with this, but we do not cover that in this course.

Your arrow keys are usable while in insert mode.

To exit insert mode, simply hit the ESC key.

As always you can exit the editor by making sure you are in command mode, hitting the ESC key, then entering :wq (which **w**rites, then **q**uits), or :q! which will quit without saving your changes.

Adapted from:
""Learning the vi Editor/Vim/Modes" by Multiple contributors, WikiBooks is licensed under CC BY-SA 3.0

05-A.2.1: Vim - Insert Mode is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-A.2.2: Vim - Visual Mode

Visual Mode

When editing text with **vim**, visual mode can be extremely useful for identifying chunks of text to be manipulated.

Vim's visual mode has three versions: character, line, and block. The keystrokes to enter each mode are:

- Character mode: **v** (lowercase)
- Line mode: **V** (uppercase)
- Block mode: **Ctrl+v**

Here are some ways to use each mode to simplify your work.

Character Mode

Character mode can highlight a sentence in a paragraph or a phrase in a sentence. Then the visually identified text can be deleted, copied, changed, or modified with any other **vim** editing command.

```
Not like the brazen giant of Greek fame,
With conquering limbs astride from land to land;
Here at our sea-washed, sunset gates shall stand
A mighty woman with a torch, whose flame
Is the imprisoned lightning, and her name
Mother of Exiles. From her beacon-hand
Glows world-wide welcome; her mild eyes command
The air-bridged harbor that twin cities frame.
“Keep, ancient lands, your storied pomp!” cries she
With silent lips. “Give me your tired, your poor,
Your huddled masses yearning to breathe free,
The wretched refuse of your teeming shore.
Send these, the homeless, tempest-tost to me,
I lift my lamp beside the golden door!
~
~
~
-- VISUAL --
```

Once you have a file opened (or have typed in some text), make sure you are in command mode by hitting ESC. Then hit the v key and you will see the word "-- VISUAL --" show at the bottom left corner of the screen. Now, using the arrow keys, you can move one character at a time, or up and down a line at a time. Once you have selected the desired text, you can edit it with any **vim** command: delete, edit, yank it (to be pasted elsewhere).

Also, notice at the lower right hand corner of the terminal there are 2 numbers - 4,35. The first number is the row in the file, and the second number is the column the cursor is currently on.

There is also the word 'All.' This means that All of the lines of the file are shown in the terminal. If the file extended past the end of the screen it would show a percentage of where the cursor is within the entire file, OR, it would say 'Top' if the cursor was at the top of the file, OR, it would say 'Bot' if the cursor was at the bottom of the file.

Line Mode

Line mode is similar to character mode, although it highlights a single sentence or sentences. Then the visually identified text can be deleted, copied, changed, or modified with any of **vim**'s editing commands.

```
Not like the brazen giant of Greek fame,
With conquering limbs astride from land to land;
Here at our sea-washed, sunset gates shall stand
A mighty woman with a torch, whose flame
Is the imprisoned lightning, and her name
Mother of Exiles. From her beacon-hand
Glows world-wide welcome; her mild eyes command
The air-bridged harbor that twin cities frame.
“Keep, ancient lands, your storied pomp!” cries she
With silent lips. “Give me your tired, your poor,
Your huddled masses yearning to breathe free,
The wretched refuse of your teeming shore.
Send these, the homeless, tempest-tost to me,
I lift my lamp beside the golden door!
~
~
~
-- VISUAL LINE--
```

Once you have a file opened (or have typed in some text), make sure you are in command mode by hitting ESC. Then hit the SHIFT-V keys and you will see the word "-- VISUAL LINE --" show at the bottom left corner of the screen. Now, using the arrow keys, you can move one character at a time, or up and down a line at a time. Once you have selected the desired text, you can edit it with any **`vim`** command: delete, edit, yank it (to be pasted elsewhere).

As always you can exit the editor by making sure you are in command mode, hitting the ESC key, then entering :wq (which **w**rites, then **q**uits), or :q! which will quit without saving your changes.

Adapted from:
"Getting started with Vim visual mode" by Susan Lauber, opensource.com is licensed under CC BY-SA 4.0

05-A.2.2: Vim - Visual Mode is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-A.2.3: Vim - Command-Line Mode

Command-Line Mode

You may be asking, "Why does **vim** have so many modes, why is that necessary?" Well, you must remember that the original vi, the forerunner to **vim**, was created a LONG time ago, when keyboards did not have arrow keys, there were no graphical user interfaces, and everything was done on a character-based terminal.

Vim uses one mode to insert text into the file; one mode to navigate around the file - although **vim** allows use of the arrow keys to move while in insert mode; one mode that allows us to select chunks of code to modify; and now we have the Command-Line mode where we can do a majority of the editing.

In Command-Line we use certain keys or key sequences to edit the file and to move around within the file. Depending on where you look for information about **vim**, it may tell you that the Command-Line mode is only for the commands that start with ':' - but we are including other editing and search functionality as well.

Remember - you need to make sure you are in Normal mode (which is actually the same as Command-Line mode) - if you are not sure what mode you are in you can simply hit the ESC key which is harmless if you are in Normal mode already.

ex Commands

There is an editor called **ex**, which is even older than **vi**, and it has some interesting syntax to its command. These commands are all entered by using a ":' and then the actual command. There are lots of **vim** resources that discuss the use of these very powerful commands, but the syntax is very different, especially if you are not used to operating from a command line.

Some examples of ex commands (some of these we have already seen):

Mode Name	Purpose
:wq	Write file to disk and quit the editor
:q!	Quit - does NOT save file before quit
:q	Quit (a warning is printed if a modified file has not been saved)
:w	Writes the file back to the original file
:w myFile	Writes the current file to the file named myFile; it is created if it doesn't exist (if it does exist you will need to issue :w! myFile)
:/[some pattern]	Search forward in the file for the pattern - places cursor at the beginning of the line
:?[some pattern]	Search backward in the file for the pattern - places cursor at the beginning of the line
:10,25 w newFile	Writes lines 10 through 25 to a file named newFile (if that file exists you will need to issue 10,25 w! newFile)

The example below is only showing the bottom of the terminal window, which is where the ex commands show up as soon as you enter the colon character. This image shows writing the current file to a file named myTest.



Figure **5.2.3.1**: Run ex commands in vi. ("vi using ex" by Patrick McClanahan is licensed under CC BY-SA 4.0)

Cursor Movement

To move the cursor you can use the arrow keys. There is also a set of commands to move the cursor to specific places within the file. These commands allow you to move quickly throughout the file. These keystrokes DO NOT show up anywhere on the

terminal window.

Key	Purpose
nG	Cursor goes to the specified n (where n is some integer value) line
^F	(CTRl F) Forward screenful
^B	(CTRL B) Backward screenful
^f	One page forward
^b	One page backward
^U	Up half screenful
^D	Down half screenful
$	Move cursor to the end of current line
0	(zero) Move cursor to the beginning of current line
w	Forward one word
b	Backward one word

Edit Commands

There are keys/key sequences that edit the file. These keystrokes DO NOT show up anywhere on the terminal window.

Key	Purpose
x	Delete character
dw	Delete word from cursor on
db	Delete word backward
dd	Delete line
d$	Delete to end of line
d^	(d caret, not CTRL d) Delete to beginning of line
yy	Yank current line
y$	Yank to end of current line from cursor
yw	Yank from cursor to end of current word
5yy	Yank, for example, 5 lines
p	Paste below/after cursor
P	Paste above/before cursor

The two paste commands have two slightly different uses. If you yank an entire line from the file, then p (lower case) will paste that line below the current line and P (upper case) will paste it above the current line. If, however, you have yanked a word or even a part of the line, then p (lower case) will paste the text after the cursor position on the same line and P (upper case) will paste it before the cursor position on the current line.

Adapted from:
"Learning the vi Editor/Vim/Modes" by Multiple Contributors, WikiBooks is licensed under CC BY-SA 3.0

05-A.2.3: Vim - Command-Line Mode is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-A.3: Nano Editor

Nano Editor

Anther editor that is often used is **nano**. It is much less powerful than vim, but can be simpler to use for those not used to all the nuances of vim.

To get to **nano**, simply type the command on the command line, and if you have a file you want to edit, include the file name. **Nano** opens up the file and places the cursor at the top of the file. To move around you can use the arrow keys or the PageUp/PageDown keys, as well as the Home/End keys. These will behave as you expect them to.

```
 GNU nano 4.8                        New Buffer

Not like the brazen giant of Greek fame,
With conquering limbs astride from land to land;
Here at our sea-washed, sunset gates shall stand
A mighty woman with a torch, whose flame









                              [ Read 5 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

For editing, the Delete key and backspace key will remove individual characters. Using the key sequence CTRL+^ you can "mark" text that you want to copy, then move the cursor to the end of the section of text, hit Alt+^ which copies the marked text, then move the cursor to where you desire to place the text, and Ctrl+U will paste it before the cursor position.

All of the commands shown at the bottom of the window are a CTRL key in conjunction with the specified key. You do NOT have to use the shift key - so ^G is simply the Ctrl key pressed along with the g key.

05-A.3: Nano Editor is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-A.4: Gedit and More

Other Editors

There are actually other editors, both for the command line as well as simple GUI based text editors. **`Gedit`** is a well known and simple editor. If you are working on a system that has graphics capabilities enabled, **`gedit`** might be the editor of choice.

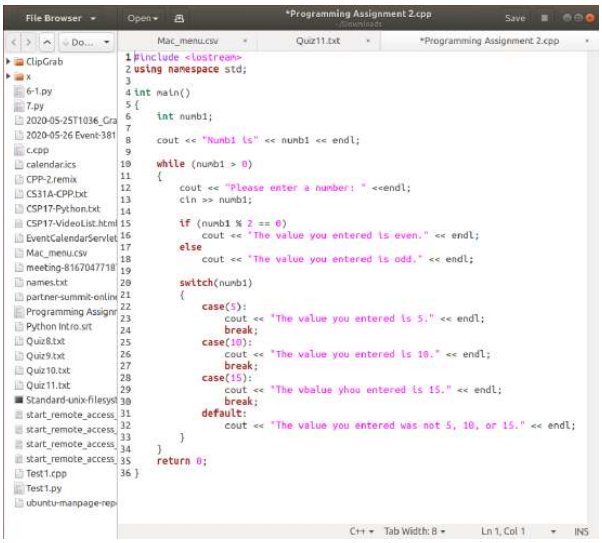


Figure **5.4.1**: Using gedit. ("gedit in Linux" by Patrick McClanahan is licensed under CC BY-SA 4.0)

Some of the features of **`gedit`** are:

- supports multiple tabs, allowing editing of multiple files
- syntax highlighting for various languages (C, C++, Java, Python…) /li>
- undo/redo
- print preview support
- clipboard support
- search and replace
- multi-language, spell-checking
- configurable fonts and colors
- a flexible plugin system

The example shows **`gedit`** with the file browser enabled on the left side, and three tabs across the top. Notice the colored text that is available for computer program creation/editing.

Other Linux Editors

There are numerous other editors that run on Linux. They can be found by searching the web. Simply search "Linux editors" - responses include command line editors as well as GUI based tools. Some are great for writing code, others are better for a variety of other uses.

05-A.4: Gedit and More is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-B.5: Searching for Files on Linux

EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

It is easy to forget where certain files are located, or perhaps the system, or another user/administrator created the file you need. No worries - Linux provides multiple tools to locate files on the system.

locate Command in Linux with Examples

The **`locate`** command in Linux is used to find the files by name. The **`locate`** utility works by searching through a database to find the file in question. This database contains bits and parts of files and their corresponding paths on your system. By default, the **`locate`** command does not check whether the files found in the database still exist and it will not report files created after the most recent update of the database.

Syntax:

```
locate [OPTION]... PATTERN...
```

Command Options:

Option	Optoin Meaning
-b, –basename	Match only the base name against the specified patterns, which is the opposite of –wholename.
-c, –count	Instead of writing file names on standard output, write the number of matching entries only.
-d, –database DBPATH	Replace the default database with DBPATH. DBPATH is a (colon) separated list of database file names. If more than one –database option is specified, the resulting path is a concatenation of the separate paths. An empty database file name is replaced by the default database. A database file name – refers to the standard input. Note that a database can be read from the standard input only once.
-e, –existing	Print only entries that refer to files existing at the time locate is run.
-i, –ignore-case	Ignore case distinctions when matching patterns.
-n some number	Write matches up to the specified number.
-L, –follow	When checking whether files exist (if the –existing option is specified), follow trailing symbolic links. This causes broken symbolic links to be omitted from the output. This option is the default behavior. The opposite can be specified using –nofollow.
-h, –help	Write a summary of the available options to standard output and exit successfully.
-l, –limit, -n LIMIT	Exit successfully after finding LIMIT entries. If the –count option is specified, the resulting count is also limited to LIMIT.
-m, –mmap	Ignored, but included for compatibility with BSD and GNU locate.
-P, –nofollow, -H	When checking whether files exist (if the –existing option is specified), do not follow trailing symbolic links. This causes broken symbolic links to be reported like other files. This option is the opposite of –follow.
-0, –null	Separate the entries on output using the ASCII NUL character instead of writing each entry on a separate line. This option is designed for interoperability with the –null option of GNU xargs.
-S, –statistics	Write statistics about each read database to standard output instead of searching for files and exit successfully.
-q, –quiet	Write no messages about errors encountered while reading and processing databases.
-r, –regexp REGEXP	Search for a basic regexp REGEXP. No PATTERNs are allowed if this option is used, but this option can be specified multiple times.
–regex	Interpret all PATTERNs as extended regexps.
-s, –stdio	Ignored, for compatibility with BSD and GNU locate.

Option	Optoin Meaning
-V, –version	Write information about the version and license of locate on standard output and exit successfully.
-w, –wholename	Match only the whole path name against the specified patterns. This option is the default behavior. The opposite can be specified using –basename.

The `updatedb` Command

The updatedb command creates or updates the database used by locate, and is usually run daily by cron to update the default database. If the database already exists, its data is reused to avoid rereading directories that have not changed. This command reads the /etc/updatedb.conf file, which contains directories that **`updatedb`** excludes from the database. The default location of the database is /etc/mlocate/mlocate.db.

Syntax:

```
updatedb [ OPTION ]
```

Command Options:

Options	Option Meaning
-f, --add-prunefsFS	Add entries in white-space-separated list FS to PRUNEFS.
-n, --add-prunenamesNAMES	Add entries in white-space-separated list NAMES to PRUNENAMES.
-e, --add-prunepathsPATHS	Add entries in white-space-separated list PATHS to PRUNEPATHS.
-U, --database-root PATH	Store only results of scanning the file system subtree rooted at PATH to the generated database. The whole file system is scanned by default.
-o, --output FILE	Write the database to FILE instead of using the default database.
--prune-bind-mounts FLAG	Set PRUNE_BIND_MOUNTS to FLAG, overriding the configuration file.
--prunefs FS	Set PRUNEFS to FS, overriding the configuration file.
--prunenames NAMES	Set PRUNENAMES to NAMES, overriding the configuration file.
--prunepaths PATHS	Set PRUNEPATHS to PATHS, overriding the configuration file.
-l, --require-visibility FLAG	Set PRUNEPATHS to PATHS, overriding the configuration file.

Adapted from
"locate command in Linux with Examples" by Pragya_Chaurasia, Geeks for Geeks is licensed under CC BY-SA 4.0

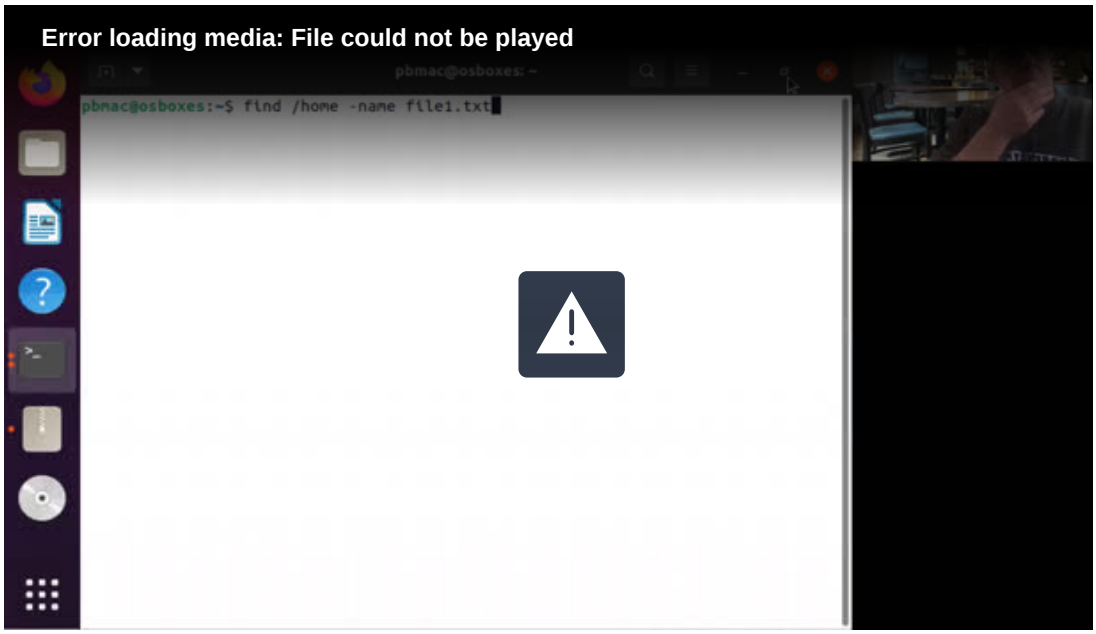
05-B.5: Searching for Files on Linux is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-B.5.1: Searching for Files on Linux - find Command

The **find** Command

The **find** command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the '-exec' other UNIX commands can be executed on files or folders found.

Find is a powerful tool, yet it can be a bit confusing at first. This video discusses some of the more common uses of find, and will show some examples.



Syntax :

```
$ find [where to start searching from]
 [expression determines what to find] [-options] [what to find]
```

find Command Options :

Option	Option Meaning
-exec CMD	The file being searched which meets the above criteria and returns 0 as its exit status for successful command execution.
-ok CMD	It works the same as -exec except the user is prompted first.
-print	Display the path name of the files found by using the rest of the criteria.
-inum N	Search for files with inode number ‘N.’
-links N	Search for files with ‘N’ links.
-name demo	Search for files that are specified by ‘demo.’
-newer file	Search for files that were modified/created after ‘file.’
-perm octal	Search for the file if permission is ‘octal.’
-empty	Search for empty files and directories.
-size +N/-N	Search for files of ‘N’ blocks; ‘N’ followed by ‘c’ can be used to measure size in characters; ‘+N’ means size > ‘N’ blocks and ‘-N’ means size < 'N' blocks.
-user name	Search for files owned by user name or ID ‘name.’
expr	True if ‘expr’ is true; used for grouping criteria combined with OR or AND.
! expr	True if ‘expr’ is false.

find versus locate

The **`find`** command searches in the real system. Even though it is slower it always returns up-to-date output. The **`find`** command also has many more options (size, modification time...)

Whereas the **`locate`** command uses a previously built database (the **`updatedb`** command usually builds the database once a day). It is much faster, but uses an 'older' database and searches only for file names or parts of the name.

Adapted from:
"find command in Linux with examples" by rishabh1322, Geeks for Geeks is licensed under CC BY-SA 4.0

05-B.5.1: Searching for Files on Linux - find Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-B.5.2: Searching for Files on Linux - which Command

The which Command

The **which** command returns the pathnames of the files (or links) which would be executed in the current environment, had its arguments been given as commands in a strictly POSIX-conformant shell. It does this by searching the PATH variable for executable files matching the names of the arguments.

```
pbmac@ubuntu $ echo $PATH
/home/pbmac/.rbenv/plugins/ruby-build/bin:/home/pbmac/.rbenv/shims:/home/pbmac/.rbenv/
/home/pbmac/scripts:/home/pbmac/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:
/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/bin:
/var/local/Android-SDK/Android-sdk/platform-tools:/var/local/Android-SDK/tools:
pbmac@ubuntu $ which ping
/usr/bin/ping
pbmac@ubuntu $ which netcat uptime
/usr/bin/netcat
/usr/bin/uptime
pbmac@ubuntu $ which -a touch
/usr/bin/touch
/bin/touch
/bin/touch
pbmac@ubuntu $
```

The PATH is the directories that Linux will look in to find an executable file when you attempt to issue a command, either from the command line, or selecting an icon on a GUI based system. in this example, Linux will sequentially search through each directory looking for the command, and will execute the first file that matches the user's input.

which can accept a single argument, such as ping, or multiple arguments as shown with netcat and uptime; and if you use the -a option, it will show ALL of the files with the given name, not just the first one it finds. In the example it found two files named touch and lists the path to both files.

The use of the -a option is a great tool to make sure you are executing the file you think you are executing. If there is another file named the same, and it is in your PATH before the file you want to execute, the first one gets executed. This can cause you to see output that is not what you expected, or different from what you expected.

05-B.5.2: Searching for Files on Linux - which Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-B.5.3: Searching for Files on Linux - whereis Command

The whereis Command

The **`whereis`** command locates the binary, source and manual files for the specified command names. The supplied names are first stripped of leading pathname components and any (single) trailing extension of the form ***.ext*** (for example: **.c**) Prefixes of **s.** resulting from use of source code control are also dealt with. **`whereis`** then attempts to locate the desired program in the standard Linux places, and in the places specified by **$PATH** and **$MANPATH**.

Syntax :

```
$ whereis [options] [directory name] {file name}
```

Command Options :

Option	Option Meaning
-b	Search for binaries.
-m	Search for manuals.
-s	Search for sources.
-u	Only show the command names that have unusual entries. A command is said to be unusual if it does not have just one entry of each explicitly requested type. Thus 'whereis -m -u *' asks for those files in the current directory which have no documentation file, or more than one.
-B list	Limit the places where whereis searches for binaries by a whitespace-separated list of directories.
-M list	Limit the places where whereis searches for manuals and documentation in Info format, by a whitespace-separated list of directories.
-S list	Limit the places where whereis searches for sources by a whitespace-separated list of directories.
-f	Terminates the directory list and signals the start of filenames. It must be used when any of the -B, -M, or -S options are used.
-l	Output the list of effective lookup paths that whereis is using. When none of -B, -M, or -S is specified, the option will output the hard-coded paths that the command was able to find on the system.
-h, --help	Display help text and exit.
-V, --version	Display version information and exit.

```
pbmac@ubuntu $ whereis ping
ping: /usr/bin/ping /usr/share/man/man8/ping.8.gz
pbmac@ubuntu $ whereis netcat uptime
netcat: /usr/bin/netcat /usr/share/man/man1/netcat.1.gz
uptime: /usr/bin/uptime /usr/share/man/man1/uptime.1.gz
pbmac@ubuntu $ whereis touch
```

```
touch: /usr/bin/touch /usr/share/man/man1/touch.1.gz
pbmac@ubuntu $
```

The **`whereis`** command accepts a single argument - ping, or multiple arguments - netcat and uptime. The output shows the path to the execuatable, and to the man page file. Where we entered multiple arguments, the output shows the path for the executable and the man page for all the arguments. Where an argument exists in multiple locates, such as touch, it outputs the path to all known executables and manuals. Notice in this example the touch command is located in two places on the system BUT the man pages have only a single copy.

Adapted from:
"whereis command in Linux with Examples" by Suraj1994, Geeks for Geeks is licensed under CC BY-SA 4.0

05-B.5.3: Searching for Files on Linux - whereis Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-C.6: Working with Files and Directories - cat Command

EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Linux File and Directory Commands

It is easy to get overwhelmed by the sheer number of Linux commands. Depending on which list you look at you will find anywhere from about 125 to over 600 (the 600 number includes a lot of commands from various distributions).

This portion will cover some of the commands that will be useful for managing files and directories.

The `cat` Command

The **`cat`** (short for concatenate) command is a fairly simple tool designed to concatenate and write file(s) to your screen, which is known as standard output (stdout). The simplest use of **`cat`** is to show the contents of a file. One of the most common ways to use the **`cat`** command is for viewing configuration files, such as those in the /etc directory. The **`cat`** command will display a file without risking damage to it. Opening a critical configuration file using an editor such as Vi or Nano could inadvertently make unwanted changes to the file. The **`cat`** command is not an editor and therefore poses no risk of making changes to a file's content.

Syntax:

```
cat [filename]
```

Command Options:

Option	Option Meaning
`-A`	Equivalent to -vET.
`-b`	Number nonempty output lines.
`-e`	Equivalent to -vE.
`-E`	Display $ at end of each line.
`-n`	Number all output lines.
`-s`	Suppress repeated empty output lines.
`-t`	Equivalent to -vT.
`-T`	Display TAB characters as ^I.

```
pbmac@ubuntu $ cat temp
Not like the brazen giant of Greek fame,
With conquering limbs astride from land to land;
Here at our sea-washed, sunset gates shall stand
A mighty woman with a torch, whose flame
pbmac@ubuntu $
```

In the example the **`cat`** command is used to display the contents of the file **temp**.

05-C.6: Working with Files and Directories - cat Command is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-C.6.1: Working with Files and Directories - head/tail Commands

The **head** Command

The **head** command, as the name implies, prints lines at the top of the file; by default, it prints the first 10 lines of the specified files. There is an option that allows you to print a number of lines other than 10. If more than one file name is provided then data from each file is preceded by its file name.

Syntax:

```
head [OPTION]... [FILE]...
```

Command Options

Options	Option Meaning
-c, --bytes=,[-]NUM/	Print the first NUM bytes of each file; if the leading '-' is specified then print all but the last NUM bytes of each file.
-n, --lines=,[-]NUM/	Print the first NUM lines instead of the first 10; if the leading '-' is specified then print all but the last NUM lines of each file.
-q, --quiet, --silent	Never print headers giving file names.
-v, --verbose	Always print headers giving file names.
-z, --zero-terminated	Line delimiter is NUL, not newline.

There are a variety of options...but **head** is very useful for any of the options. The example show the top 10 lines since there is NO option specified. The second command specifies only the first five lines of the file are to be output.

```
pbmac@ubuntu $ head temp
Not like the brazen giant of Greek fame,
With conquering limbs astride from land to land;
Here at our sea-washed, sunset gates shall stand
A mighty woman with a torch, whose flame
Is the imprisoned lightning, and her name
Mother of Exiles. From her beacon-hand
Glows world-wide welcome; her mild eyes command
The air-bridged harbor that twin cities frame.
“Keep, ancient lands, your storied pomp!” cries she
With silent lips. “Give me your tired, your poor,
pbmac@ubuntu $
pbmac@ubuntu $
pbmac@ubuntu $ head -n 5 temp
Not like the brazen giant of Greek fame,
With conquering limbs astride from land to land;
Here at our sea-washed, sunset gates shall stand
A mighty woman with a torch, whose flame
Is the imprisoned lightning, and her name
pbmac@ubuntu $
```

The tail Command

The **`tail`** command is the complementary of **`head`** command.The **`tail`** command, as the name implies, prints the last N number of data of the given input. By default it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

Syntax:

```
tail [OPTION]... [FILE]...
```

Command Options

Options	Meaning
-c, --bytes=,[-]NUM/	Output the last NUM bytes; or use **-c** +NUM to output starting with byte NUM of each fileprint the first NUM bytes of each file; if the leading '-' is specified then print all but the last NUM bytes of each file.
-n, --lines=,[-]NUM/	Output the last NUM lines, instead of the last 10; or use **-n** +NUM to output starting with line NUM.
-f, --follow	Output appended data as the file grows.
-q, --quiet, --silent	Never print headers giving file names.
-v, --verbose	Always print headers giving file names.
-z, --zero-terminated	Line delimiter is NUL, not newline.

The **`tail`** command operates just like the head command, except it operates from the bottom of the file, not the top.

Adapted from:
"Head command in Linux with examples" by Akash Gupta, Geeks for Geeks is licensed under CC BY-SA 4.0
"Tail command in Linux with examples" by Akash Gupta, Geeks for Geeks is licensed under CC BY-SA 4.0

05-C.6.1: Working with Files and Directories - head/tail Commands is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

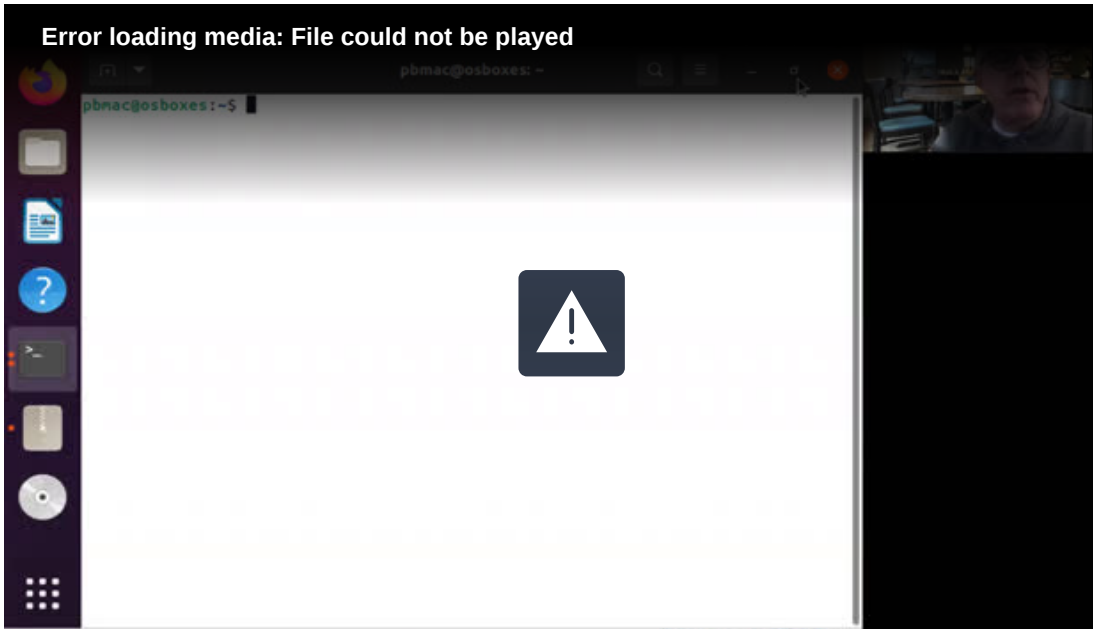
05-C.6.2: Working with Files and Directories - less/more Command

The less Command

The **`less`** command is Linux utility which can be used to read contents of text file one page (one screen) per time. It has faster access because if the file is large, it doesn't access the complete file, but accesses it page by page.

For example, if it's a large file and you are reading it using any text editor, then the complete file will be loaded to main memory, but less command doesn't load the entire file, instead it loads it part by part, which makes it faster.

This video covers the basics of several commands, a lot of good information.



Syntax:

```
less [options] [input file]
```

Command Options:

Options	Option Meaning
-E	Causes less to automatically exit the first time it reaches end of file.
-f	Forces non-regular file to open.
-F	Causes less to exit if entire file can be displayed on first screen.
-g	Highlights the string which was found by last search command.
-G	Suppresses all highlighting of strings found by search commands.
-i	Cause searches to ignore case.
-n	Suppresses line numbers.
-p pattern	Tells less to start at the first occurrence of pattern in the file.
-s	Causes consecutive blank lines to be squeezed into a single blank line.

The **`less`** command can display either directly from a file, or from the output of another command.

```
pbmac@ubuntu $ less dmesg
...the contents of the file are displayed...
```

```
pbmac@ubuntu $ cat dmesg | less
```

In this example, we see the **less** command with a single file name as the input for the less command. In the second line we see a very common use of the output of one command being "piped" to the input of the second command.

When the **less** command is used to display data, it will show a screenful, then pause with the cursor at the bottom of the screen next to a colon prompt. At this point the user has several options: 1) move on to the next screen by simply hitting the space bar, the PageDown or the **f** key (forward) 2) move down one line by hitting the Enter key or the down arrow 3) move to the previous screen by hitting the PageUp button, or the **b** key (back) 4) move up one line by using the up arrow key 5) enter a / followed by a text string, then use the **n** key to search forward, or the **N** key to search backwards in the file or 6) enter **q** to quit.

There are powerful search capabilities built into **less**, but that is beyond what we would like to cover here. Numerous resources are available on the web to understand how this capability works.

The more Command

The **more** command is actually the original command, with **less** being developed at a later time. They both are known as "pagers," in that they display a page at a time. Both commands have similar capabilities and operate in very much the same manner. The biggest difference is that **less** is a bit faster because it does not load the entire file into memory before it displays the output.

Syntax:

```
more [ OPTIONS ] [-num] [+/pattern] [+linenum] [file_name]
```

Command Options:

Options	Option Meaning
-f	Causes more to count logical, rather than screen lines (i.e., long lines are not folded).
-p	Do not scroll. Instead, clear the whole screen and then display the text.
-c	Do not scroll. Instead, paint each screen from the top, clearing the remainder of each line as it is displayed.
-l	**more** usually treats ^L (form feed) as a special character, and will pause after any line that contains a form feed. The -l option will prevent this behavior.
-d	**more** will prompt the user with the message "[Press space to continue, 'q' to quit.]" and will display "[Press 'h' for instructions.]" instead of ringing the bell when an illegal key is pressed.
+num	Start at line number num .
-u	Suppress underlining.
+/	The +/ option specifies a string that will be searched for before each file is displayed.
-s	Squeeze multiple blank lines into one.

To help remember the difference between **less** and **more**, a common joke is to say, "**less** > **more**," implying that **less** has greater functionality than **more**. A similar saying is that "**less** is **more**, more or less."

Adapted from:
"less command in Linux with Examples" by Mandeep Singh, Geeks for Geeks is licensed under CC BY-SA 4.0

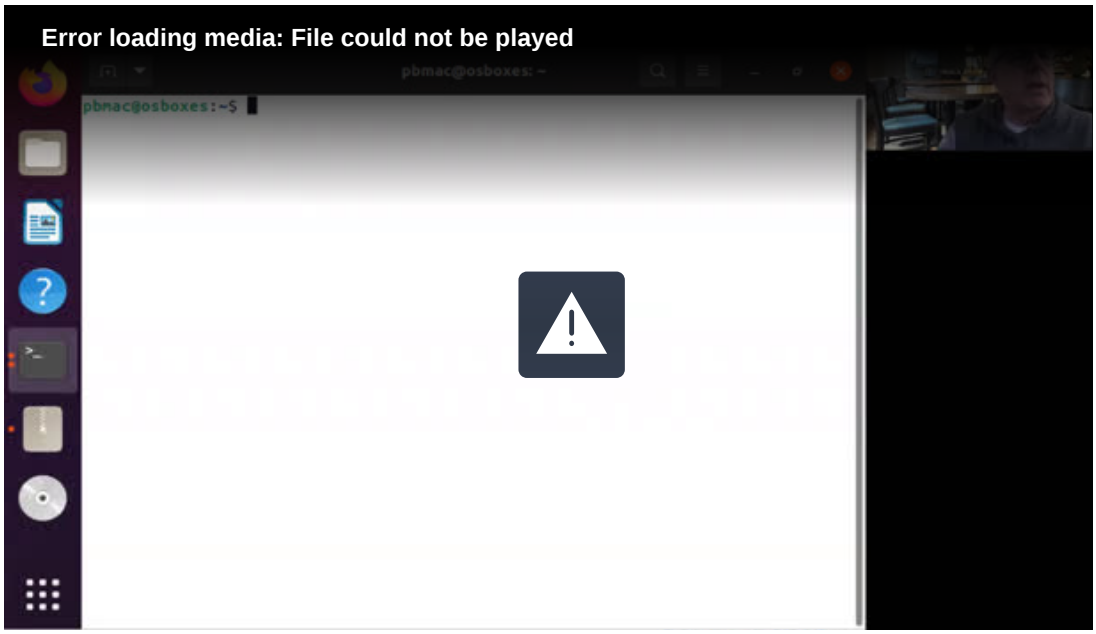
05-C.6.2: Working with Files and Directories - less/more Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-C.6.3: Working with Files and Directories - cp/mv Command

The cp Command

cp stands for copy. This command is used to copy files or group of files or directories. It creates an exact copy of a file on a disk with different file name. **cp** command requires at least two filenames in its arguments.

This video covers the use of copy and move commands.



Syntax:

```
cp [OPTION] Source Destination
cp [OPTION] Source Directory
cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory
```

Looking at the above options a bit of clarification is necessary.

In the first option, the Destination can be a file or a directory. If the Destination does NOT exist, **cp** will assume the Destination is to be a file and will create a copy of the Source file. If the Destination file exists cp will overwrite the destination file unless the -i option (see below) is specified.

In the second option things can get a bit confusing. If the command contains two directory names, **cp** copies all files of the source directory to the destination directory, creating any files or directories needed. This mode of operation requires an additional option, typically **-R**, to indicate the recursive copying of directories as in:

```
cp -R Src_directory Dest_directory
```

When copying one directory to another as in the above command, the behavior of **cp** depends on whether Dest_directory exists or not. If the Dest_directory does NOT exist, **cp** creates it and recursively copies the contents of Src_directory into Dest_directory. But if Dest_directory DOES exist the copy of Src_directory becomes sub-directory within Dest_directory.

In the last option the **cp** command allows multiple source files to be copied to a single destination directory. The destination directory MUST exist as **cp** will not create it.

Command Option:

Options for cp **and** mv:

Options	Option Meaning
-i, --interactive	Prompt before overwrite (overrides a previous -n option).
-n	Do not overwrite an existing file (overrides a previous -ioption).
-R, -r, --recursive	Copy directories recursively.

The mv Command

mv stands for move, and it is used to move one or more files or directories from one place to another in Linux. It has two distinct functions:
1) It renames a file or directory and 2) it moves a group of files to different directory.

mv options are pretty much the same as the cp command, although there is not a recursive capability with the mv command

No additional space is consumed on a disk during renaming. This command normally works silently, which means no prompt for confirmation. The source file or directory no longer exists.

Syntax:

```
mv [ OPTION ] source destination
```

mv has a few nuances just like **cp** (If the source is a file and the destination does not exist, **mv** assumes the destination is a file, and will create the new file).

- If the source is a directory and the destination does NOT exist, **mv** will create the destination directory.
- If the source is a directory and the destination is a directory, then the source directory is moved as a subdirectory of destination.

The options listed above for **cp** are also valid for **mv**. If you desire to see the full range of options for any command take a look at the man page for the command.

Adapted from:
"cp command in Linux with examples" by AKASH GUPTA 6, Geeks for Geeks is licensed under CC BY-SA 4.0
"mv command in Linux with examples" by AKASH GUPTA 6, Geeks for Geeks is licensed under CC BY-SA 4.0

05-C.6.3: Working with Files and Directories - cp/mv Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

- **Current page** is licensed CC BY-SA 4.0.
- **05-C.6: Working with Files and Directories - cat Command** is licensed CC BY-NC 4.0.

05-C.6.4: Working with Files and Directories - rm/touch Command

The rm Command

`rm` stands for remove, and it is used to remove files, directories, and links. **By default, it does not remove directories.**

This command normally works silently and it should be used carefully, because once you delete a file in Linux the content cannot be recovered.

Syntax:

```
rm [OPTION]... FILE...
```

There are a couple of common options, which you might notice are similar to the options for both **`cp`** and **`mv`**.

Options	Option Meaning
-i	Prompt before every removal.
-r, -R, --recursive	Remove directories and their contents recursively.

The **`unlink`** command is very similar to the **`rm`** command, but only removes a single file at a time, and does not remove directories.

The touch Command

The **`touch`** command updates the access and modification times of each file to the current time. The **`touch`** command is a standard command in Linux used to create, change, and modify the timestamp of a file. If the file specified does not exist **`touch`** will create the file, but the file does not have any content.

Syntax:

```
touch file_name
```

Command Options:

Options	Option Meaning
-a	Change only the access time.
-c, --no-create	Do not create any files.
-d, --date=STRING	Parse STRING and use it instead of current time.
-h, --no-dereference	Affect each symbolic link instead of any referenced file (useful only on systems that can change the timestamps of a symlink).
-m	Change only the modification time.
-r, --reference=FILE	Use this file's times instead of current time.
-t STAMP	Use [[CC]YY]MMDDhhmm[.ss] instead of current time.
--time=WORD	Change the specified time: WORD is access, atime, or use: equivalent to -a WORD is modify or mtime.

Adapted from:
"rm command in Linux with examples" by AKASH GUPTA 6, Geeks for Geeks is licensed under CC BY-SA 4.0
"touch command in Linux with Examples" by Bhumika_Rani, Geeks for Geeks is licensed under CC BY-SA 4.0

05-C.6.4: Working with Files and Directories - rm/touch Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-C.6.5: Working with Files and Directories - ls/rmidir/mkdir Command

The `ls` Command

The **`ls`** command lists files on a Linux system. It's a simple command. The **`ls`** command is a basic command in Linux but is used extensively to look at file and directory information. The command has a plethora of arguments and features and most users barely scratch the surface of the depth of this command. By this point in the course you should be comfortable using the **`ls`** command in the lab environment.

Syntax

```
ls [ PTION ] [FILE]
```

Command Options

Options	Option Meaning
-a, --all	Do not ignore entries starting with . (a period).
-f	Do not sort, enable -aU, disable -ls --color.
-i, --inode	Print the index number of each file.
-l	Use a long listing format.
-r, --reverse	Reverse order while sorting.
-R, --recursive	List subdirectories recursively.

The `mkdir` Command

The **`mkdir`** command creates directories. This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have enough permissions to create a directory in the parent directory.

Syntax:

```
mkdir [ OPTIONS ] [directories ...]
```

Command Options

Option	Option Meaning
-m, --mode=MODE	Set file mode (as in chmod), not a=rwx - umask .

The **`mkdir`** has only a couple of options, and only one that is used with any regularity - that is the -m option.

```
pbmac@ubuntu $ mkdir one
pbmac@ubuntu $ ls -ld one
drwxrwxr-x 2 pbmac pbmac 4096 Jul 15 16:39 one
pbmac@ubuntu $ mkdir -m a=rwx two
pbmac@ubuntu $ ls -ld two
drwxrwxrwx 2 pbmac pbmac 4096 Jul 15 16:39 two
pbmac@ubuntu $ mkdir -m a=r three
pbmac@ubuntu $ ls -ld three
```

```
dr--r--r-- 2 pbmac pbmac 4096 Jul 15 16:39 three
pbmac@ubuntu $ mkdir -m u=r,g=w,o=x four
pbmac@ubuntu $ ls -l
total 16
dr---w---x 2 pbmac pbmac 4096 Jul 15 16:41 four
drwxrwxr-x 2 pbmac pbmac 4096 Jul 15 16:39 one
dr--r--r-- 2 pbmac pbmac 4096 Jul 15 16:39 three
drwxrwxrwx 2 pbmac pbmac 4096 Jul 15 16:39 two
pbmac@ubuntu $
```

In the example to the above:

1. Using **`mkdir`** with no permission settings, using system defaults.
2. Second **`mkdir`**, specify a=rwx - the a means ALL get rwx permissions.
3. Third **`mkdir`** specify a=r - ALL get read permission.
4. Fourth **`mkdir`** has u=r,g=w,o=x - this is USER gets read, GROUP gets write, and OTHER gets execute.

`mkdir` can set permissions other than the default permissions, which can be useful at times.

The rmdir Command

The **`rmdir`** command is used remove empty directories from the filesystem. The **`rmdir`** command removes each directory specified in the command line, but only if the directory is empty. Directories that have sub-directories or files within them will not be removed.

Syntax:

```
rmdir [options...] [directories ...]
```

`rmdir` has options that are the same as **`mkdir`** shown above.

Adapted from:
"mkdir command in Linux with Examples" by rossoskull, Geeks for Geeks is licensed under CC BY-SA 4.0
"rmdir command in Linux With Examples" by rahulkumarmandal, Geeks for Geeks is licensed under CC BY-SA 4.0

05-C.6.5: Working with Files and Directories - ls/rmidir/mkdir Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7: Handling Text Files - echo/printf

EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

Now that we have learned how to use various commands to manage our files, let's look at how we can process text files in various ways. An introduction to many of the command-line tools for text manipulation is important to those interested in growing in their knowledge and use of Linux.

The `echo` Command

The **`echo`** command is used to display a line of text that is passed as an argument. This is a built-in command that is mostly used in shell scripts and batch files to output status text to the screen or a file.

Syntax :

```
echo [ OPTIONS ] [string]
```

Echo has a few options, but usually it is simply used by itself.

```
pbmac@ubuntu $ echo "This is Linux"
This is Linux
pbmac@ubuntu $ echo -e "The \nIs \nLinux"
The
Is
Linux
pbmac@ubuntu $
```



Figure **5.7.1**: Linux echo Command. ("echo command" by Patrick McClanahan is licensed under CC BY-SA 4.0)

The first line simply repeats what is given on the command line - it "echoes" the text.

The -e option allows the use of certain special linefeed characters. In this example we see the \n, which stand for a linefeed - this puts a newline where the \n is.

The `printf` Command

The **`printf`** command is very similar to **`echo`**. This command is exactly the same as the **`printf`** command in the C programming language. It is used at times in Linux shell scripts, but is a bit more complex. However **`printf`** does provide some great options for outputting information that needs formatting.

```
pbmac@ubuntu $ name='Pat'; printf "Hi %s\n" $name
Hi Pat
pbmac@ubuntu $
```

The example above might be a bit confusing since we have not talked about shell scripts. The variable "name" is assigned a value, then the **`printf`** interprets the %s as saying "look at the next variable ($name) and interpret it as a string" (the %s says string - a %d says a decimal integer, %f is a floating point value). There are some good references on the web if you want to learn more about this command.

05-D.7: Handling Text Files - echo/printf is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7.1: Handling Text Files - tr/wc Command

The `tr` Command

The **`tr`** command is a command line utility for translating or deleting characters. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace. **`tr`** stands for translate.

Syntax:

```
tr [ OPTION ] SET1 [SET2]
```

Command Options:

Options	Option Meaning
-c	Complements the set of characters in string.i.e., operations apply to characters not in the given set.
-d	Delete characters in the first set from the output.
-s	Replaces repeated characters listed in the set1 with single occurrence.
-t	Truncates set1.

The **`tr`** command is pretty powerful, but also quite simple. If you get good at working with regular expressions you can do some complex things with **`tr`**, but mostly it is used with simple types of translations.

The example below shows the content of the file classes.txt, just two lines of lowercase text:

```
pbmac@ubuntu $ cat classes.txt
Linux
Operating Systems
Introduction to Cybersecurity
pbmac@ubuntu $ cat classes.txt | tr "[a-z]" "[A-Z]"
LINUX
OPERATING SYSTEMS
INTRODUCTION TO CYBERSECURITY
pbmac@ubuntu $ cat classes.txt | tr "[::lower:]" "[:upper:]"
tr: invalid character class ‘:lower’
pbmac@ubuntu $ cat classes.txt | tr "[:lower:]" "[:upper:]"
LINUX
OPERATING SYSTEMS
INTRODUCTION TO CYBERSECURITY
pbmac@ubuntu $
```

The first **`tr`** command gives the range [a-z], which says to take any character between a and z. The second range is [A-Z]. So the **`tr`** command says take the characters that match the first argument and translate them to the appropriate character in the second argument.

The second **`tr`** invocation uses the :lower: and :upper: arguments to accomplish the same results.

The wc Command

The **wc** command is mainly used for counting purpose. **wc** stands for word count. It is used to find out the number of lines, word count, byte and characters count in the files specified in the file arguments. By default it displays four-columnar output. The first column shows the number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in a file and fourth column itself is the file name which is given as argument.

Syntax:

```
wc [OPTION]... [FILE]...
```

Command Options:

Options	Option Meaning
-c, --bytes	Print the byte counts.
-m, --chars	Print the character counts.
-l, --lines	Print the newline counts.
-w, --words	Print the word counts.

The **wc** command is simple - it provides a variety of counts: lines, words and characters. The default is that **wc** display all three of the outputs, and using the options allows the user to choose which output is desired.

```
pbmac@ubuntu $ wc classes.txt nextsemester.txt
 3  6 54 classes.txt
 3  3 27 nextsemester.txt
 6  9 81 total
pbmac@ubuntu $ wc -l classes.txt nextsemester.txt
 3 classes.txt
 3 nextsemester.txt
 6 total
pbmac@ubuntu $ wc -c classes.txt nextsemester.txt
54 classes.txt
27 nextsemester.txt
81 total
pbmac@ubuntu $
```

Adapted from:
"tr command in Unix/Linux with examples" by Shivani Ghughtyal, Geeks for Geeks is licensed under CC BY-SA 4.0
"wc command in Linux with examples" by Akash Gupta, Geeks for Geeks is licensed under CC BY-SA 4.0

05-D.7.1: Handling Text Files - tr/wc Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7.2: Handling Text Files - sort/diff Commands

The `sort` Command

The **`sort`** command is used to sort a file, arranging the records in a particular order. By default, the **`sort`** command sorts a file assuming the contents are ASCII. Using options in **`sort`** command, it can also be used to sort numerically.

Some features of the command are as follows:

- **`sort`** command sorts the contents of a text file, line by line.
- **`sort`** is a standard command line program that prints the lines of its input or concatenation of all files listed in its argument list in sorted order.
- The **`sort`** command is a command line utility for sorting lines of text files. It supports sorting alphabetically, in reverse order, by number, by month and can also remove duplicates.
- The **`sort`** command can also sort by items not at the beginning of the line, ignore case sensitivity and return whether a file is sorted or not. Sorting is done based on one or more sort keys extracted from each line of input.
- By default, the entire input is taken as sort key. Blank space is the default field separator.

Syntax:

```
sort [ OPTION ] filename
```

Command Options:

Options	Option Meaning
-b, --ignore-leading-blanks	ignore leading blanks
-d, --dictionary-order	consider only blanks and alphanumeric characters
-f, --ignore-case	fold lower case to upper case characters
-g, --general-numeric-sort	compare according to general numerical value
-i, --ignore-nonprinting	consider only printable characters
-M, --month-sort	compare (unknown) < 'JAN' < ... < 'DEC'
-h, --human-numeric-sort	compare human readable numbers (e.g., 2K 1G)
-n, --numeric-sort	compare according to string numerical value
-R, --random-sort	shuffle, but group identical keys. See shuf(1)
--random-source=,FILE/	get random bytes from FILE
-r, --reverse	reverse the result of comparisons
--sort=,WORD/	sort according to WORD: general-numeric -g, human-numeric -h, month -M, numeric -n, random -R, version -V
-V, --version-sort	natural sort of (version) numbers within text

The **`sort`** command is another command that has an abundance of options, and only a few are shown in the table above. The example below is a very straightforward sort. The cat command shows the random names of states. Then the sort command produces an output list of the states sorted in alphabetic order. NOTE: the original file, states, is not altered at all. The new list is simply output to the terminal.

```
pbmac@pbmac-server $ cat states
California
New York
Florida
Texas
North Carolina
Alabama
South Dakota
Washington
Georgia
Ohio
pbmac@pbmac-server $ sort states
Alabama
```

```
California
Florida
Georgia
New York
North Carolina
Ohio
South Dakota
Texas
Washington
```

With the plethora of options **`sort`** can sort according to alpha or numeric values, or reverse sort. For columnar data it can sort by any one of the columns, and specifying any character as the column delimiter.

This command is very useful and very powerful.

The `diff` Command

The **`diff`** command is used to display the differences in the files by comparing the files line by line. Unlike its fellow members, cmp and comm, it tells us which lines in one file have to be changed to make the two files identical.

The important thing to remember is that **`diff`** uses certain special symbols and instructions that are required to make two files identical. It tells you the instructions on how to change the first file to make it match the second file.

Special symbols are:

```
a : add
c : change
d : delete
```

Syntax :

```
diff [ OPTIONS ] File1 File2
```

Command Options

Options	Option Meaning
-b	Ignore spacing differences.
-c	Display a list of differences with three lines of context.
-i	Ignore case differences.
-t	Expand tab characters in output lines.
-u	Output results in unified mode, which presents a more streamlined format.
-w	Ignore spacing differences and tabs.

Let's say we have two files with names a.txt and b.txt containing 5 American states.

```
pbmac@pbmac-server $ cat states.1
New York
Florida
```

```
Texas
Alabama
South Dakota
Washington
pbmac@pbmac-server $ cat states.2
California
New York
Florida
Texas
North Carolina
Alabama
Washington
Ohio
```

Now, applying **`diff`** command without any option we get the following output:

```
pbmac@pbmac-server $ diff states.1 states.2
0a1
> California
3a5
> North Carolina
5d6
< South Dakota
6a8
> Ohio
```

NOTE: neither file is altered, only output of the differences is sent to the terminal.

Let's take a look at what this output means. The first line of the **`diff`** output will contain:

- Line numbers corresponding to the first file
- A special symbol
- Line numbers corresponding to the second file.

Like in our case, 0a1 which means after lines 0(at the very beginning of file) you have to add California to match the second file line number 1. It then tells us what those lines are in each file proceeded by the symbol:

- Lines preceded by a < are lines from the first file.
- Lines preceded by > are lines from the second file.

Next line contains 3a5 which means at line 3 of the first file we need to add line 5 from the second file. The we have to delete from line 5 to line 6 (BUT not deleting line 6) from the first file. Finally, after line 6 of the first file we add line 8 from the second file.

Adapted from:
"SORT command in Linux/Unix with examples" by Mohak Agrawal, Geeks for Geeks is licensed under CC BY-SA 4.0
"diff command in Linux with examples" by AKASH GUPTA 6, Geeks for Geeks is licensed under CC BY-SA 4.0

05-D.7.2: Handling Text Files - sort/diff Commands is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7.3: Handling Text Files - cut/paste

The `cut` Command

The **`cut`** command cuts out the sections from each line of files and writes the result to standard output. It can be used to cut parts of a line by byte position, character and field. Basically the cut command slices a line and extracts the text. Specify an option with the command otherwise it throws an error. If more than one file name is provided then data from each file is not preceded by its file name.

Syntax:

```
cut [ OPTION ] [FILE]...
```

Command Options:

Options	Option Meaning
-b, --bytes=,LIST/	select only these bytes
-c, --characters=,LIST/	select only these characters
-d, --delimiter=,DELIM/	use DELIM instead of TAB for field delimiter - VALID ONLY FOR FIELDS
-f, --fields=,LIST/	select only these fields; also print any line that contains no delimiter character, unless the -s option is specified
-s, --only-delimited	do not print lines not containing delimiters

So **`cut`** can be used look at a file by the byte, the character or fields. In the example below we have a file statecap.txt - it is three fields that are separated by a colon.

```
pbmac@pbmac-server $ cat statecap.txt
Montana:Helena:MT
Arizona:Phoenix:AZ
South Dakota:Pierre:SD
Missouri:Jefferson City:MO
Alaska:Juneau:AK
Wyoming:Cheyenne:WY
Nevada:Carson City:NV
```

We can **`cut`** a range of bytes/characters/fields. The first example we **`cut`** only the first 3 bytes of the file. In the second we **`cut`** the first eight characters. (The bytes and characters are often the same - but bytes will pick up non-characters). In the third example we specify the colon delimiter and the third field - so we get the state abbreviation.

```
pbmac@pbmac-server $ cut -b 1-3 statecap.txt
Mon
Ari
Sou
Mis
Ala
Wyo
```

```
Nev
pbmac@pbmac-server $ cut -c 1-8 statecap.txt
Montana:
Arizona:
South Da
Missouri
Alaska:J
Wyoming:
Nevada:Cpbmac@pbmac-server $ cut -d':' -f3 statecap.txt
MT
AZ
SD
MO
AK
WY
NV
```

As with most commands we can combine **`cut`** with other command input/output to form more powerful command string. We **`cat`** the file, **`cut`** the first field, then **`sort`** it in alphabetic order.

```
pbmac@pbmac-server $ cat statecap.txt | cut -d':' -f 1 | sort -
Alaska
Arizona
Missouri
Montana
Nevada
South Dakota
Wyoming
```

The `paste` Command

The **`paste`** command is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by **tab** as delimiter, to the standard output. When no file is specified, or a dash ("-") is inserted instead of file name, **`paste`** reads from standard input and gives output as it is until a interrupt command **[Ctrl-c]** is given.

Syntax:

```
paste [ OPTIONS ] [FILES]...
```

Command Options:

Options	Option Meaning
-d, --delimiters=, LIST/	reuse characters from LIST instead of TAB
-s, --serial	paste one file at a time instead of in parallel
-z, --zero-terminated	line delimiter is NUL, not newline

The **paste** command will take each specified element from each file (in this case the first line of each file) and concatenate them together on a single line. Obviously the files must be in sync with one another otherwise the output would be gibberish.

```
pbmac@pbmac-server $ paste state.txt capital.txt abbrev.txt
Montana    Helena    MT
Arizona    Phoenix    AZ
South Dakota    Pierre    SD
Missouri    Jefferson City    MO
Alaska    Juneau    AK
Wyoming    Cheyenne    WY
Nevada    Carson City    N
```

Adapted from:
"cut command in Linux with examples" by Saloni Gupta, Geeks for Geeks is licensed under CC BY-SA 4.0
"Paste command in Linux with examples" by Akash Gupta, Geeks for Geeks is licensed under CC BY-SA 4.0

05-D.7.3: Handling Text Files - cut/paste is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7.4: Handling Text Files - grep Command

The `grep` Command

`grep` is a command-line utility for searching plain-text data sets for lines that match a regular expression.

Regular expression provides an ability to match a “string of text” in a very flexible and concise manner. A “string of text” can be further defined as a single character, word, sentence or particular pattern of characters.

Like the shell’s wild–cards which match similar filenames with a single expression, **`grep`** uses an expression of a different sort to match a group of similar patterns.

- [ ]: Matches any one of a set characters
- [ ] with hyphen: Matches any one of a range characters
- ^: The pattern following it must occur at the beginning of each line
- ^ with [ ] : The pattern must not contain any character in the set specified
- $: The pattern preceding it must occur at the end of each line
- . (dot): Matches any one character
- \ (backslash): Ignores the special meaning of the character following it
- *: Zero or more occurrences of the previous character
- (dot).*: Nothing or any numbers of characters

Syntax:

```
grep [OPTION...] PATTERNS [FILE...]
grep [OPTION...] -e PATTERNS ... [FILE...]
grep [OPTION...] -f PATTERN_FILE ... [FILE...]
```

Command Options:

Options	Option Meaning
-E, --extended-regexp	Interpret PATTERNS as extended regular expressions (EREs, see below).
-F, --fixed-strings	Interpret PATTERNS as fixed strings, not regular expressions.
-c, --count	Suppress normal output; instead print a count of matching lines for each input file. With the -v, --invert-match option (see below), count non-matching lines.
-l, --files-with-matches	Suppress normal output; instead print the name of each input file from which output would normally have been printed. The scanning will stop on the first match.
-o, --only-matching	Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.
-v, --invert-match	Invert the sense of matching, to select non-matching lines.
-f FILE, --file=FILE	Obtain patterns from FILE, one per line. If this option is used multiple times or is combined with the -e (--regexp) option, search for all patterns given. The empty file contains zero patterns, and therefore matches nothing.

The **`grep`** command has numerous options which makes it a powerful tool for processing text files.

```
pbmac@pbmac-server $ grep  “New[abc]”  filename
```

The command states look for the text "New," followed by any ONE of the characters in the brackets. So this command specifies the search pattern as any one of the following:

```
Newa , Newb or Newc
```

Another example is:

```
pbmac@pbmac-server $ grep  “New[a-e]” filename
```

Specifying a dash means "any character in that sequence." So, the above command will match on any of these:

```
Newa , Newb, Newc , Newd or Newe
```

It is okay to use NO options and just search for the word "failed" (or any other word) in a particular file.

```
pbmac@pbmac-server $ grep failed /var/log/syslog | head
Aug  4 00:06:34 pbmac-server colord[1061]: failed to get session [pid 23278]: No data
Aug  4 00:06:34 pbmac-server colord[1061]: message repeated 5 times: [ failed to get s
Aug  4 06:05:14 pbmac-server google-chrome.desktop[2996]: [3035:3040:0804/060514.43282
handshake failed; returned -1, SSL error code 1, net_error -113
Aug  4 06:05:14 pbmac-server google-chrome.desktop[2996]: [3035:3040:0804/060514.47484
handshake failed; returned -1, SSL error code 1, net_error -113
```

As you get more comfortable with regular expressions many commands will become more useful and more powerful.

Adapted from:
"Regular Expression in grep" by Akshay Rajput., Geeks for Geeks is licensed under BY-SA 4.0

05-D.7.4: Handling Text Files - grep Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7.5: Handling Text Files - awk Command

The awk Command

The **awk** command programming language is a scripting language used for manipulating data and generating reports. It requires no compiling, and allows the user to use variables, numeric functions, string functions, and logical operators.

awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in **each line** of a file, and the action that is to be taken when a match is found within a line. **awk** is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that match with the specified patterns and then performs the associated actions.

Syntax:

```
awk options 'selection _criteria {action }' input-file > output-file
```

Options	Meaning
`awk '/regular expression/ {print}' filename`	**awk** will print any line that contains the specified regular expression
`awk 'relational expression {print}' filename`	**awk** allows users to look at individual fields with $1, $2 etc. Users can specify all the typical relational operators: ==, !=, >=, <=, <, >
`awk 'pattern1 && pattern2 {print}' filename`	the && is an AND - so BOTH patterns have to be true for **awk** to output the line
`awk 'pattern1 || pattern2 {print}' filename`	the || is an OR - so ONE pattern has to be true for **awk** to output the line
`awk '{print pattern1 ? option1 : option2 }' filename`	if pattern1 is TRUE, then **awk** prints option1, else it prints option2
`awk 'pattern1, pattern2 {print}' filename`	if pattern1 is true, the **awk** prints the line until pattern2 is true

```
# if the line contains a 'y' or a 'z' - then print that line
pbmac@pbmac-server $ awk '/[yz]/ {print}' state.txt
Arizona
Wyoming

# if $2 - that is the second field (space is the separator)- is Dakota, then print tha
pbmac@pbmac-server $ awk '$2 == "Dakota" {print}' state.txt
South Dakota

# if the firsts field is South, and the second field is Dakota, then print that line
pbmac@pbmac-server $ awk '$1 == "South" && $2 == "Dakota" {print}' state.txt
South Dakota

# if the fisrt field is South OR Wyoming - then print that line
pbmac@pbmac-server $ awk '$1 == "South" || $1 == "Wyoming" {print}' state.txt
South Dakota
Wyoming
```

```
# if the first field is South, then print "Abbreviation is $4" - where $4 is the fourt
# "Capital is $2" - where $2 is the second field on that line.
pbmac@pbmac-server $ awk '{print ($1 == "South") ? "Abbrevition is " $4 : "Capital is
Capital is Helena
Capital is Phoenix
Abbrevition is SD
Capital is Jefferson
Capital is Juneau
Capital is Cheyenne
Capital is Carson

# if the first field is Montana - then print all lines until the third field of the ir
pbmac@pbmac-server $ awk '$1 == "Montana", $3 == "Pierre" {print}' stcap.txt
Montana Helena MT
Arizona Phoenix AZ
South Dakota Pierre SD
```

Adapted from:
"gawk command in Linux with Examples" by Mandeep Sheoran, Geeks for Geeks is licensed under BY-SA 4.0

05-D.7.5: Handling Text Files - awk Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7.6: Handling Text Files - sed Command

The **sed** Command

The **sed** command is a stream editor which is used to perform basic text transformations on an input stream, usually either a file or input from a Linux command pipeline. However the most common use of **sed** command is for substitution or for find and replace. By using **sed** you can edit files even without opening them, which is a much quicker way to find and replace something in a file than first opening that file in VI Editor and then changing it.

- **sed** is a powerful text stream editor. It can do insertion, deletion, and search and replace (substitution).
- **sed** command in Linux supports regular expression which allows it to perform complex pattern matching.

Syntax:

```
sed [ OPTIONS ] [SCRIPT] [INPUTFILE...]
```

Command Options:

Options	Option Meaning
-n, --quiet, --silent	Suppress automatic printing of pattern space.
-e script, --expression=script	Add the script to the commands to be executed.
-f script-file, --file=script-file	Add the contents of script-file to the commands to be executed.
--follow-symlinks	Follow symlinks when processing in place.
-i[SUFFIX], --in-place[=SUFFIX]	Edit files in place (this makes a backup with file extension SUFFIX, if SUFFIX is supplied).
-l N, --line-length=N	Specify the desired line-wrap length, N, for the "l" command.
--POSIX	Disable all GNU extensions.
-r, --regexp-extended	Use extended regular expressions in the script.
-s, --separate	Consider files as separate rather than as a single continuous long stream.
-u, --unbuffered	Load minimal amounts of data from the input files and flush the output buffers more often.
--help	Display a help message, and exit.
--version	Output version information, and exit.

A brief example: notice we have the work 'geek' in the file colossus.txt. This word should be Greek. We can easily fix that with a single command.

```
pbmac@pbmac-server $ grep geek colossus.txt
Not like the brazen giant of geek fame,
pbmac@pbmac-server $ sed 's/geek/Greek/' colossus.txt
Not like the brazen giant of Greek fame,
With conquering limbs astride from land to land;
```

```
Here at our sea-washed, sunset gates shall stand
A mighty woman with a torch, whose flame
Is the imprisoned lightning, and her name
Mother of Exiles. From her beacon-hand
Glows world-wide welcome; her mild eyes command
The air-bridged harbor that twin cities frame.
```

The **sed** command can be broken down this way: s says to search; the word geek between the slash characters - /geek/ - is the search term; the second term - /Greek/ - is the replacement term; and then we provide the file name. So - we search for "geek" and replace it with "Greek." The file remains unchanged though, and you would have to save the changes.

Adapted from:
"Sed Command in Linux/Unix with examples" by Akshay Rajput and Mohak Agrawal., Geeks for Geeks is licensed under BY-SA 4.0

05-D.7.6: Handling Text Files - sed Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-D.7.7: Handling Text Files - ln Command

The `ln` Command

Hard Links

Each hard linked file is assigned the same Inode value as the original, therefore they reference the same physical file location. Hard links are more flexible and remain linked even if the original or linked files are moved throughout the file system, although hard links are unable to cross different file systems. **`ls -l`** command shows all the links while the link column shows number of links. Links have actual file contents, and removing any link just reduces the link count, but doesn't affect other links. We cannot create a hard link for a directory to avoid recursive loops. If the original file is removed, then the link will still show the content of the file.

Command to create a hard link is:

Syntax:

```
ln [,OPTION/]... [,-T/] ,TARGET LINK_NAME/     (1st form)
ln [,OPTION/]... ,TARGET/                      (2nd form)
ln [,OPTION/]... ,TARGET/... ,DIRECTORY/       (3rd form)
ln [,OPTION/]... ,-t DIRECTORY TARGET/...      (4th form)
```

In the first form, the command creates a link to TARGET with the name LINK_NAME.

In the second form, the command creates a link to TARGET in the current directory.

In the third and fourth forms, the command creates links to each TARGET in DIRECTORY. The ln command creates hard links by default, to create symbolic links you must yse the **--symbolic option**. By default, each destination (name of new link) should not already exist. When creating hard links, each TARGET must exist. Symbolic links can hold arbitrary text; if later resolved, a relative link is interpreted in relation to its parent directory.

```
# A simple link of one file to another name - notice the output of thje ls -li commanc
# showing the file's node number is the same for both files, bacause they are links
pbmac@pbmac-server $ ln program1.cpp linkprogram1.cpp
pbmac@pbmac-server $ ls -li
total 0
12987377 -rw-r--r-- 2 pbmac pbmac 0 Aug  4 17:42 linkprogram1.cpp
12987377 -rw-r--r-- 2 pbmac pbmac 0 Aug  4 17:42 program1.cpp

# link the file to a file with the same name in the directory subFolder
pbmac@pbmac-server $ ln program1.cpp subFolder/
pbmac@pbmac-server $ ls subFolder/
program1.cpp

# link the file to a different name in the directory subFolder
pbmac@pbmac-server $ ln program1.cpp subFolder/anotherLink.cpp
pbmac@pbmac-server $ ls subFolder/
anotherLink.cpp  program1.cpp

# link a file in a subfolder to a file with the same name in the current folder.
pbmac@pbmac-server $ ln subFolder/testFile.txt
pbmac@pbmac-server $ ls
```

```
linkprogram1.cpp  myStuff.txt  program1.cpp  subFolder  testFile.txt
pbmac@pbmac-server $
```

The image shows the concept - two different filenames pointing to the same space on the physical disk.

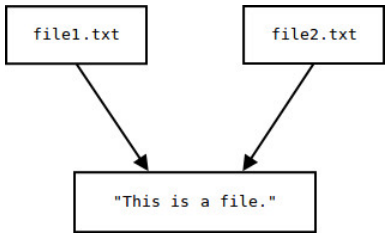


Figure 5.7.7.1: Hard link files. ("hard links" by Patrick McClanahan is licensed under CC BY-SA 4.0)

Symbolic/Soft Links

- A symbolic link is similar to the file shortcut feature which is used in Windows Operating systems. Each symbolic linked file contains a separate Inode value that points to the original file. Similar to hard links, any changes to the data in either file is reflected in the other. Symbolic links can be linked across different file systems, although if the original file is deleted or moved, the symbolically linked file will not work correctly (called hanging link).
- ls -li command shows the node numbers, all linked files will show the first permission column value of l, and the filename field points to original file.
- Symbolic link contains the path for original file and not the contents.
- Removing symbolic link doesn't affect anything, but if removing the original file, the link becomes a "dangling" link which points to nonexistent file.
- A symbolic link can link to a directory.
- Link across filesystems: If you want to link files across the filesystems, you can only use symlinks links.

Symbolic/Soft Link Syntax

```
$ ln  -s [original filename] [link name]
```

As mentioned above, the symbolic, or soft link, has a different filename, but the link points back to the original filename, not to the disk drive. Notice in the following example, the node numbers are different signifying that these are two different names; also the filename field shows the link back to program1.cpp.

```
# notice the node numbers are different, the first colum of permissions is a 'l', and
pbmac@pbmac-server $ ls -li program1.cpp newprogram.cpp
12987272 lrwxrwxrwx 1 pbmac pbmac 12 Aug  5 07:55 newprogram.cpp -> program1.cpp
12987377 -rw-r--r-- 4 pbmac pbmac  0 Aug  4 17:42 program1.cpp
```

This can be viewed as the shown in the image to the right. The linked filename, file2.txt, points back to the original filename. The original filename points to the physical file on disk.

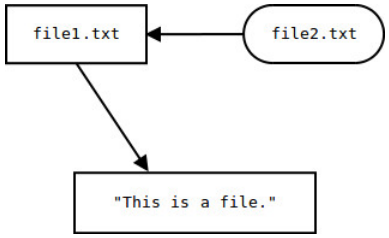


Figure 5.7.7.1: Soft links files. ("soft links" by Patrick McClanahan is licensed under CC BY-SA 4.0)

Adapted from:
"ln command in Linux with Examples" by Rangeesh_A_R, Geeks for Geeks is licensed under CC BY-SA 4.0

"Soft and Hard links in Unix/Linux" by Akash Gajjar, Geeks for Geeks is licensed under CC BY-SA 4.0

05-D.7.7: Handling Text Files - ln Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-E.8: File Output Manipulation

EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

Text Streams—a Universal Interface

Everything in Linux revolves around streams of data—particularly text streams. Data streams are the raw materials upon which the Linux core utilities, and many other command-line tools, perform their work. The use of Standard Input/Output (STDIO) for program input and output is a key foundation of the Linux way of doing things. STDIO was first developed for Unix and has found its way into most other operating systems since then, including DOS, Windows, and Linux.

STDIO

STDIO, standard input/output, was developed by Ken Thompson, one of the original authors of Unix, in the early versions of Unix. Programs that implement STDIO use standardized file handles for input and output rather than files that are stored on a disk or other recording media. STDIO is best described as a buffered data stream, and its primary function is to stream data from the output of one program, file, or device to the input of another program, file, or device.

There are three STDIO data streams, each of which is automatically opened as a file at the startup of a program—well, those programs that use STDIO. Each STDIO data stream is associated with a file handle, which is just a set of metadata that describes the attributes of the file. File handles 0, 1, and 2 are explicitly defined by convention and long practice as STDIN, STDOUT, and STDERR, respectively.

STDIN, File handle 0, is standard input which is usually input from the keyboard. STDIN can be redirected from any file, including device files, instead of the keyboard. It is not common to need to redirect STDIN, but it can be done.

STDOUT, File handle 1, is standard output which sends the data stream to the display by default. It is common to redirect STDOUT to a file or to pipe it to another program for further processing.

STDERR, File handle 2. The data stream for STDERR is also usually sent to the display.

Redirection

The work of any command is either taking input or giving an output or both. So, Linux has some commands or special characters to redirect these input and output functionalities. For example: suppose we want to run a command called “date” - if we run it will print the output to the current terminal screen. But our requirement is different - we don’t want the output to be displayed on the terminal, we want the output to be saved in a file. This could be done very easily with output redirection. Redirection here simply means diverting the output or input.

Output Redirection

Output redirection takes the output from command, system utility or error messages from Linux and redirects that output to a different location, usually a file. Below are most of the redirection operators and comments about how they work.

```
pbmac@pbmac-server $ command > somefile
      # Redirect stdout to a file.
      # Creates the file if not present, otherwise overwrites it.

pbmac@pbmac-server $ ls -lR > dir-tree.list
      # Creates a file containing a listing of the directory tree.

pbmac@pbmac-server $ : > filename
      # There no command given, just the : character
      # The > truncates file "filename" to zero length.
      # If file not present, creates zero-length file (same effect as 'touch').
```

```
      # The : serves as a dummy placeholder, producing no output.

pbmac@pbmac-server $ > filename
      # There no command given, just the > character
      # The > truncates file "filename" to zero length.
      # If file not present, creates zero-length file (same effect as 'touch').
      # (Same result as ": >", above, but this does not work with some shells.)

pbmac@pbmac-server $ command >> someFile.txt
      # Redirect stdout to a file.
      # Creates the file if not present, otherwise appends to it.


# Single-line redirection commands (affect only the line they are on):
# --------------------------------------------------------------------
# Redirect stdout to file "filename."
   1>filename
pbmac@pbmac-server $ some_command 1>outFile.txt


# Redirect and append stdout to file "filename."
   1>filename
pbmac@pbmac-server $ some_command 1>>outFile.txt

# Redirect stderr to file "filename."
   2>>filename
pbmac@pbmac-server $ some_command 2>errorFile.txt

# Redirect and append stderr to file "filename."
   2>&1
pbmac@pbmac-server $ some_command1 output.txt 2>&

Error messages get sent to same place as standard output.
    >>filename 2>&1
pbmac@pbmac-server $ some_command >>filename 2>&1
      # Appends both stdout and stderr to the file "filename" ...
```

Adapted from:

"Chapter 20. I/O Redirection" by Mendel Cooper, The Linux Documentation Guide is in the Public Domain, CC0

"Read and write data from anywhere with redirection in the Linux terminal " by Seth Kenlon, opensource.com is licensed under CC BY-SA 4.0

"Input Output Redirection in Linux" by Hemusharma196, Geeks for Geeks is licensed under CC BY-SA 4.0

05-E.8: File Output Manipulation is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-E.8.1: File Output Manipulation - pipe

EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

The Pipe

A pipe is a form of redirection (transfer of standard output to some other destination) used to send the output of one command/program/process to another command/program/process for further processing. Linux systems allow stdout of a command to be connected to stdin of another command. You can make it do so by using the pipe character **'|.'**

Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/programs/processes. The command line programs that do the further processing are referred to as filters.

This direct connection between commands/programs/processes allows them to operate simultaneously and permits data to be transferred between them continuously rather than having to pass it through temporary text files or through the display screen.
Pipes are unidirectional **i.e., data flows from left to right through the pipeline.**

Syntax :

```
command_1 | command_2 | command_3 | .... | command_N
```

If we were to issue the **`ls -l`** command against the /dev directory we would get a long list of the files. If we wanted to see a screen of those files, we can use the **`less`** command, which would then hand us a screen of filenames, then we hit the SPACE bar and we get the next screen. We can use all of the **`less`** options to view the output:

```
pbmac@pbmac-server $ ls -l /dev | less
total 0
crw-rw-rw-  1 root  root       14,  12 Aug  4 12:59 adsp
crw-r--r--  1 root  root       10, 235 Aug  4 12:59 autofs
drwxr-xr-x  2 root  root           380 Aug  4 12:59 block
drwxr-xr-x  2 root  root            80 Aug  4 12:58 bsg
crw-------  1 root  root       10, 234 Aug  4 12:58 btrfs-control
drwxr-xr-x  3 root  root            60 Aug  4 12:58 bus
```

So the stdio of the **`ls -l`** command is sent as input to the **`less`** command. Below is another example that shows piping multiple commands together:

```
pbmac@pbmac-server $ more stcap.txt
Montana Helena MT
Arizona Phoenix AZ
South Dakota Pierre SD
Missouri Jefferson City MO
Alaska Juneau AK
Wyoming Cheyenne WY
Nevada Carson City NV
Nevada Reno NV
Nevada Las Vegas NV
pbmac@pbmac-server $ cat stcap.txt | grep NV | wc -l
```

```
3
pbmac@pbmac-server $
```

We have added 2 more cities from Nevada to the file. The entire content is output from the **`cat`** command, and used as input to the **`grep`** command, then that output is used as input to the **`wc`** command with a **`-l`** option. One more example combines pipes and redirection:

```
pbmac@pbmac-server $ cat stcap.txt | grep NV > NVcities.txt
pbmac@pbmac-server $ cat NVcities.txt
Nevada Carson City NV
Nevada Reno NV
Nevada Lasa Vegas NV
pbmac@pbmac-server $
```

Instead of being piped to the **`wc`** command, the output of the **`grep`** command is redirected to a file name NVcities.txt.

Adapted from:
"Piping in Unix or Linux" by Rishabh132, Geeks for Geeks is licensed under CC BY-SA 4.0

05-E.8.1: File Output Manipulation - pipe is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-E.8.2: File Output Manipulation - xargs

The `xargs` Command

`xargs` is a Unix command which can be used to build and execute commands from standard input.

Importance :
Some commands like **`grep`** can accept input as parameters, but some commands accepts arguments. This is where **`xargs`** came into the picture.

Syntax :

```
xargs [ OPTIONS ] [command]
```

`xargs` Command Options

Options	Option Meaning
0	Input items are terminated by null character instead of white spaces.
-a file	Read items from file instead of standard input.
–delimiter = delim	Input items are terminated by a special character.
-E eof-str	Set the end of file string to eof-str.
-I replace-str	Replace occurrences of replace-str in the initial arguments with names read from standard input.
-L max-lines	Use at-most max-lines non-blank input lines per command line.
-p	Prompt the user about whether to run each command line and read a line from terminal.
-r	If the standard input does not contain any nonblanks, do not run the command.
-x	Exit if the size is exceeded.
–help	Print the summary of options to **`xargs`** and exit.
–version	Print the version no. of **`xargs`** and exit.

Simply put **`xargs`** takes each input and applies the command to that input. Below are some examples of **`xargs:`**

```
# the file states.list contains 3 names, we cat the file and pipe the output to xargs
# takes each line from the file and uses it as an argument to mkdir
pbmac@pbmac-server $ cat states.list
Alabama
Wyoming
Washington
pbmac@pbmac-server $ cat states.list | xargs mkdir
pbmac@pbmac-server $ ls
Alabama  states.list  Washington  Wyoming
```

```
# now we use find, searching for type d (directory) and send all the output to xargs r
# the error message is becasue the find command also finds the . directory - the direc
pbmac@pbmac-server $ find . -type d | xargs rmdir
rmdir: failed to remove '.': Invalid argument
pbmac@pbmac-server $ ls
pbmac@pbmac-server $

# there are 3 files ending in .txt, using the find command we pipe the output of the f
# each .txt file.
pbmac@pbmac-server $ ls
one.txt  states.list  three.txt  two.txt
pbmac@pbmac-server $ find . -name "*.txt" | xargs rm
pbmac@pbmac-server $ ls
states.list
```

Adapted from:
"xargs command in Linux with examples" by Mandeep Singh, Geeks for Geeks is licensed under CC BY-SA 4.0

05-E.8.2: File Output Manipulation - xargs is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

05-E.8.3: File Output Manipulation - tee & /dev/null

The `tee` Command

The **`tee`** command reads the standard input and writes it to both the standard output and one or more files. The command is named after the T-splitter used in plumbing. It basically breaks the output of a program so that it can be both displayed and saved in a file. It does both the tasks simultaneously, copies the result into the specified files or variables and also displays the result.

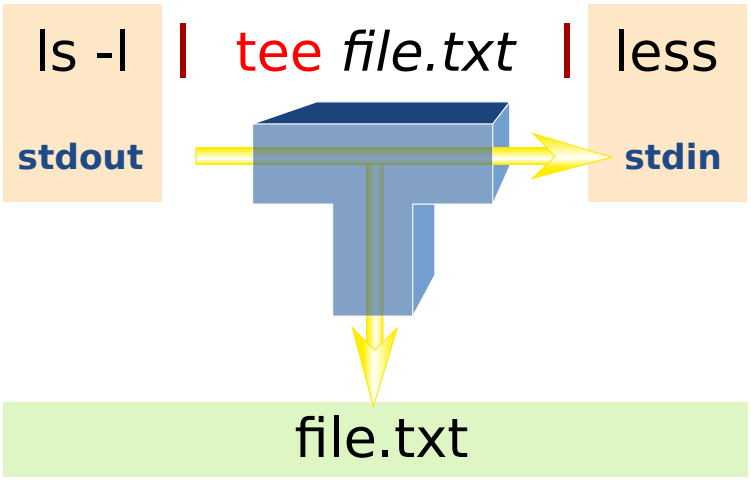


Figure **5.8.3.1**: The tee Command ("Tee.svg" by Sven, Wikimedia Commons is licensed under CC BY-SA 4.0)

SYNTAX:

```
tee [ OPTION ] [FILE]...
```

tee Command Options :

Options	Option Meaning
-a, --append	append to the given FILEs, do not overwrite
-i, --ignore-interrupts	ignore interrupt signals

In each of the following examples, the output is shown on the screen and also placed in the specified file.

```
# Output of the ls command is sent to the terminal as well as the the file output.txt,
pbmac@pbmac-server $ ls | tee output.txt
one.txt
states.list
three.txt
two.txt
pbmac@pbmac-server $ cat output.txt
one.txt
states.list
three.txt
two.txt

# Output from the ls -l command is appended to the output.txt file
```

```
pbmac@pbmac-server $ ls -l | tee -a output.txt
total 8
-rw-r--r-- 1 pbmac pbmac  0 Aug  5 12:31 one.txt
-rw-r--r-- 1 pbmac pbmac 38 Aug  5 12:32 output.txt
-rw-r--r-- 1 pbmac pbmac 27 Aug  5 11:11 states.list
-rw-r--r-- 1 pbmac pbmac  0 Aug  5 12:31 three.txt
-rw-r--r-- 1 pbmac pbmac  0 Aug  5 12:31 two.txt
pbmac@pbmac-server $ cat output.txt
one.txt
states.list
three.txt
two.txt
total 8
-rw-r--r-- 1 pbmac pbmac  0 Aug  5 12:31 one.txt
-rw-r--r-- 1 pbmac pbmac 38 Aug  5 12:32 output.txt
-rw-r--r-- 1 pbmac pbmac 27 Aug  5 11:11 states.list
-rw-r--r-- 1 pbmac pbmac  0 Aug  5 12:31 three.txt
-rw-r--r-- 1 pbmac pbmac  0 Aug  5 12:31 two.txt
pbmac@pbmac-server $
```

Linux /dev/null

All data written on a /dev/null or /dev/zero special file is discarded by the system. Use /dev/null to send any unwanted output from program/command.

Syntax:

```
command >/dev/null
```

It is possible to redirect a command's standard output messages to /dev/null where it is ignored by the shell.

```
command 1>/dev/null
```

This syntax redirects the command error output messages to /dev/null where it is ignored by the shell. OR

```
command 2> /dev/null
OR
command &>/dev/null
```

This syntax redirects both standard output and error output messages to /dev/null where it is ignored by the shell.

Adapted from:
"tee command in Linux with examples" by Anuragrawat1, Geeks for Geeks is licensed under CC BY-SA 4.0
"/dev/null discards unwanted output" by Variety of contributors is licensed under CC BY-NC-SA 3.0

05-E.8.3: File Output Manipulation - tee & /dev/null is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 06: Kernel Module Management

- 06-A.1: Examine the Linux Kernel
- 06-A.2: Linux Kernel
- 06-A.3: Linux Kernel Subsystem
- 06-B.4: Kernel Modules
- 06-B.5: Kernel Module Management - lsmod/modinfo Command
  - 06-B.5.1: Kernel Module Management - insmod/rmmod Command
  - 06-B.5.2: Kernel Module Management - modprobe/depmod Command
- 06-B.6: Kernel Modules - Configuration
- 06-B.7: Kernel Modules - Kernel Parameters
- 06-C.8: Kernel Module Monitoring
- 06-C.9: The dmesg Command

This page titled Module 06: Kernel Module Management is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

06-A.1: Examine the Linux Kernel

Module Objectives

The learning objectives for this module are as follows:

- Identify and describe the purpose of the subsystems of the Linux kernel
- Recognize the proper use of the commands for kernel module management
- Explain the commands used for kernel module configuration
- Describe where to find kernel information
- Inspect dmesg output for troubleshooting and error resolution

The Kernel

The main purpose of a computer is to run a predefined sequence of instructions, known as a **program**. A program under execution is often referred to as a **process**. Now, most special purpose computers are meant to run a single process, but in a sophisticated system such as a general purpose computer, they are intended to run many processes simultaneously. Any kind of process requires hardware resources such as Memory, Processor time, Storage space, etc.

In a general purpose computer running many processes simultaneously, we need a middle layer to manage the distribution of the hardware resources of the computer efficiently and fairly among all the various processes running on the computer. This middle layer is referred to as the **kernel**. Basically the kernel virtualizes the common hardware resources of the computer to provide each process with its own virtual resources. This makes the process seem as if it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes.

Kernel Space vs User Space

Memory is divided into 2 spaces, known as kernel space and user space, sometimes referred to as kernel and user mode.The kernel executes in the memory area referred to as kernel space, where it executes the various services that it provides. Whereas user space is that portion of memory where everything that doesn't operate in kernel mode runs.

Software running in user space has access to a certain portion of the memory that is allocated at the time the process is created. The kernel has access to all of memory. If a process starts up running in user space, it can access a small portion of kernel space via a set of system calls. The kernel developers provided these system calls that can be used by software developers to allow user space applications to access things like open/closing/reading/writing a file.

System memory was divided in this manner to ensure that user processes could not access kernel space. This promotes stability and system security.

Kernel Types

A monolithic kernel, like other kernels, manages system resources between application and hardware, but user services (NOT user space - see below) and kernel services are implemented under same address space. It increases the size of the kernel, thus increasing the size of the operating system as well. This kernel provides CPU scheduling, memory management, file management and other operating system functions through system calls. As both services are implemented under same address space, this makes operating system execution faster.

In a microkernel the user services and kernel services are implemented in different address space. The user services are kept in user address space, and kernel services are kept under kernel address space; thus also reducing the size of kernel and the size of operating system as well.

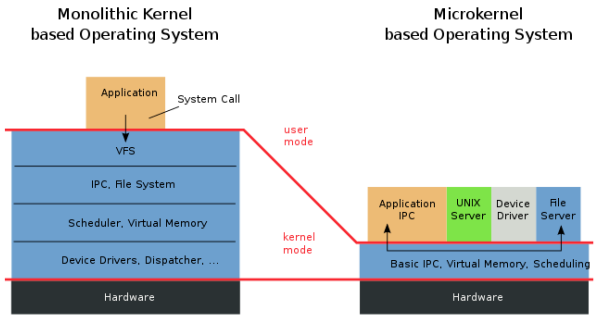


Figure **6.1.1**: Kernel Services. ("Structure of monolithic and microkernel-based operating systems, respectively" by Multiple contributors, Wikipedia is licensed under CC BY-SA 3.0)

Kernel services and user services

Notice in the image on the left, the monolithic kernel has all of the services inside the kernel itself. In the micro kernel some of those services, referred to as user services, have been taken out of the kernel, thereby shrinking the size of the kernel. These user services now reside in the user space memory.

Device Drivers

The term device driver refers to a specific type of software application which controls a specific hardware device. The device driver enables different hardware devices, printers, disk drives, and network interfaces to communicate with the computer's operating system.

Device drivers basically act as an interface between Linux and the hardware devices. This allows application software to access the hardware device easily and without having to interface directly with the hardware.

Adapted from:
"The Linux Kernel" by Arna Maity, Geeks for Geeks is licensed under CC BY-SA 4.0
"Monolithic Kernel and key differences from Microkernel" by akash1295, Geeks for Geeks is licensed under CC BY-SA 4.0
"Microkernel" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Device Driver and It's Purpose" by Satyabrata_Jena, Geeks for Geeks is licensed under CC BY-SA 4.0

06-A.1: Examine the Linux Kernel is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-A.2: Linux Kernel

The Linux Kernel

Linux is the world's largest and most pervasive open source software project in the history of computing. The Linux kernel, developed by contributors worldwide, is a free and open-source, monolithic, modular (i.e., it supports the insertion and removal at runtime of loadable kernel objects), Unix-like operating system kernel.

It is deployed on a wide variety of computing systems, such as embedded devices, mobile devices (including its use in the Android operating system), personal computers, servers, mainframes, and supercomputers.

The Linux kernel was conceived and created in 1991 by Linus Torvalds for his personal computer and with no cross-platform intentions, but has since ported to a wide range of computer architectures. Notwithstanding this, the Linux kernel is highly optimized with the use of architecture specific instructions (ISA), therefore portability isn't as easy as it is with other kernels (e.g., with NetBSD, that as of 2019 supports 59 hardware platforms).

Linux was soon adopted as the kernel for the GNU Operating System, which was created as an open source and free software, and based on UNIX as a by-product of the fallout of the Unix wars. Since then it has spawned a plethora of operating system distributions, commonly also called Linux, although, formally, the term "Linux" refers only to the kernel.

The uname Command

The **uname** command displays the information about the system.

Syntax:

```
uname [ OPTION ]
```

Below are examples of the **uname** command with the -r and -a options. The output of both commands show the kernel version, with the -a (all) option showing much more details.

```
pbmac@pbmac-server $ uname -r
4.15.0-91-generic
pbmac@pbmac-server $ uname -a
Linux pbmac-server 4.15.0-91-generic #92-Ubuntu SMP Fri Feb 28 11:09:48 UTC 2020 x86_6
```

What do the other digits mean here? Here is an explanation:

- 4 – Kernel version
- 15 – Major revision
- 0 – Minor revision
- 91 – Bug fix
- generic – Distribution-specific string. For Ubuntu, it means I'm using the desktop version. For Ubuntu server edition, it would be 'server.'

There is a LOT of other information in this output:

- Linux – Kernel name. If you run the same command on BSD or macOS, the result will be different.
- pbmac-server – Hostname.
- 4.15.0-91-generic – Kernel release (as we saw above).
- #92-Ubuntu SMP Fri Feb 28 11:09:48 UTC 2020 – This means that Ubuntu has compiled 4.15.0-91-generic 92 times. A timestamp for the last compilation is also there.
- x86_64 – Machine architecture.
- x86_64 – Processor architecture.
- x86_64 – Operating system architecture (you can run a 32-bit OS on a 64-bit processor).

- GNU/Linux – Operating system (and no, it won't show the distribution name).

Linux kernels have different support levels depending on version. E.g., version 5.4 released in November 2019, was declared to have Long-Term Support (LTS). It has 6 years of support that way.

Version	Original release date	Current version	Maintainer	EOL
5.9	11 October 2020	5.9.16	Greg Kroah-Hartman	December 2020
5.1	13 December 2020	5.10.128	Greg Kroah-Hartman & Sasha Levin	December 2026
5.11	14 February 2021	5.11.22	Greg Kroah-Hartman	May 2021
5.12	25 April 2021	5.12.19	Greg Kroah-Hartman	July 2021
5.13	27 June 2021	5.13.19	Greg Kroah-Hartman & Sasha Levin	September 2021
5.14	29 August 2021	5.14.21	Greg Kroah-Hartman	November 2021
5.15	31 October 2021	5.15.52	Greg Kroah-Hartman & Sasha Levin	October 2023
5.16	9 January 2022	5.16.20	Greg Kroah-Hartman & Sasha Levin	April 2022
5.17	20 March 2022	5.17.15	Greg Kroah-Hartman & Sasha Levin	June 2022
5.18	22 May 2022	5.18.11 12 July 2022	Greg Kroah-Hartman & Sasha Levin	
5.19-rc6	10 July 2022	5.19-rc6 10 July 2022	Linus Torvalds	

The table is from Wikipedia, and shows the 5.x.y releases. The Current Version column shows the present version of the version specified in the left most column. Several of the releases show an EOL in the far right column. This stands for End of Life, meaning that particular version is no longer supported.

Adapted from:
"Linux kernel" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Linux kernel version history" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"3 Ways to Check Linux Kernel Version in Command Line" by Abhishek Prakash, It's FOSS is licensed under CC BY-SA 4.0

06-A.2: Linux Kernel is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-A.3: Linux Kernel Subsystem

Linux Kernel Subsystems

The **Core Subsystems** of the **Linux Kernel** are as follows:

1. The Process Scheduler
2. The Memory Management Unit (MMU)
3. The Virtual File System (VFS)
4. The Networking Unit
5. Inter-Process Communication Unit

For the purpose of this article we will only be focusing on the first three important subsystems of the Linux Kernel.

The basic functioning of each of the first three subsystems is elaborated below:

- **The Process Scheduler:**
  This kernel subsystem is responsible for fairly distributing the CPU time among all the processes running on the system simultaneously.
- **The Memory Management Unit:**
  This kernel sub-unit is responsible for proper distribution of the memory resources among the various processes running on the system. The MMU does more than just simply provide separate virtual address spaces for each of the processes.
- **The Virtual File System:**
  This subsystem is responsible for providing a unified interface to access stored data across different filesystems and physical storage media.
- **The Network Subsystem**
  This allows Linux systems to connect to other systems over a network. There are a number of possible hardware devices that are supported, and a number of network protocols that can be used. The network subsystem abstracts both of these implementation details so that user processes and other kernel subsystems can access the network without necessarily knowing what physical devices or protocol is being used.
- **Inter-Process Communication**
  Processes communicate with each other and with the kernel to coordinate their activities. Linux supports a number of Inter-Process Communication (IPC) mechanisms. Signals and pipes are two of them but Linux also supports the System V IPC mechanisms named after the Unix ™ release in which they first appeared.

Kernel Space vs User Space

Memory is divided into two spaces, known as kernel space and user space, sometimes referred to as kernel and user mode. The kernel executes in the memory area referred to as kernel space, where it executes the various services that it provides, whereas user space is that portion of memory where everything that doesn't operate in kernel mode runs.

Software running in user space has access to a certain portion of the memory that is allocated at the time the process is created. The kernel has access to all of memory. If a process starts up, running in user space, it can access a small portion of kernel space via a set of system calls. The kernel developers provided these system calls that can be used by software developers to allow user space applications to access things like open/closing/reading/writing a file.

System memory was divided in this manner to ensure that user processes could not access kernel space. This promotes stability and system security.

Adapted from:
"System call" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Memory management" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"The Linux Kernel" by Arna Maity, Geeks for Geeks is licensed under CC BY-SA 4.0

06-A.3: Linux Kernel Subsystem is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-B.4: Kernel Modules

EXAM OBJECTIVES COVERED

1.2 Given a scenario, install, configure, and monitor kernel modules.
1.3 Given a scenario, configure and verify network connection parameters.
4.1 Given a scenario, analyze system properties and remediate accordingly.

Linux Kernel Modules

Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. A module can be configured as built-in or loadable. To dynamically load or remove a module, it has to be configured as a loadable module in the kernel configuration.

There are several advantages that come with using kernel modules:

- The kernel does not have to rebuild your kernel as often. This saves time and prevents the possibility of introducing an error in rebuilding and reinstalling the base kernel. Once you have a working base kernel, it is good to leave it untouched as long as possible.
- It is easier to diagnose system problems. A bug in a device driver which is bound into the kernel can stop the system from booting at all. It can also be really hard to tell which part of the base kernel caused the trouble. If the same device driver is a module, though, the base kernel is up and running before the device driver even gets loaded. If the system dies after the base kernel is up and running, it's an easy matter to track the problem down to the trouble-making device driver and just not load that device driver until the problem is fixed.
- Using modules can save memory, because they are loaded only when the system is actually using them. All parts of the base kernel stay loaded, in real storage, not just virtual storage.
- Modules are much faster to maintain and debug. What would require a full reboot to do with a filesystem driver built into the kernel can be done with a few quick commands using modules. It is possible to try out different parameters or even change the code repeatedly in rapid succession, without waiting for a boot.
- Modules are not slower, by the way, than base kernel modules. Calling either one is simply a branch to the memory location where it resides.

Module Location

The code necessary to create a new kernel with new module included, or old modules removed is usually: **`/lib/modules/$(uname -r)/kernel`** although on some distributions the code is found in **`/usr/lib/modules/$(uname -r)/kernel`**. The /usr/lib and /lib directory is where Linux stores object libraries and shared libraries that are necessary to run certain commands, including kernel code.

These directories are only necessary if you wish to rebuild your kernel with new modules included, or old modules removed. It would be best to seek out assistance from a source knowledgeable with the specific distribution you are working with.

Kernel Modules Subdirectories:

Directory	Description
arch	The arch subdirectory contains all of the architecture specific kernel code. It has further subdirectories, one per supported architecture, for example i386 and alpha.
include Although man distributions place the include files in the /usr/src/$(uname -r)	The include subdirectory contains most of the include files needed to build the kernel code. It too has further subdirectories including one for every architecture supported. The include/asm subdirectory is a soft link to the real include directory needed for this architecture, for example include/asm-i386. To change architectures you need to edit the kernel makefile and rerun the Linux kernel configuration program.
init	This directory contains the initialization code for the kernel and it is a very good place to start looking at how the kernel works.
mm	This directory contains all of the memory management code. The architecture specific memory management code lives down in arch/*/mm/, for example arch/i386/mm/fault.c.
drivers	All of the system's device drivers live in this directory. They are further sub-divided into classes of device driver, for example block.
ipc	This directory contains the kernels inter-process communications code.
modules	This is simply a directory used to hold built modules.
fs	All of the file system code. This is further sub-divided into directories, one per supported file system, for example vfat and ext2.
kernel	The main kernel code. Again, the architecture specific kernel code is in arch/*/kernel.
net	The kernel's networking code.
lib	This directory contains the kernel's library code. The architecture specific library code can be found in arch/*/lib/.
scripts	This directory contains the scripts (for example awk and tk scripts) that are used when the kernel is configured.

Adapted from:
"2. Introduction to Linux Loadable Kernel Modules" by Unlisted, The Linux Documentation Project is licensed under CC BY-SA 3.0

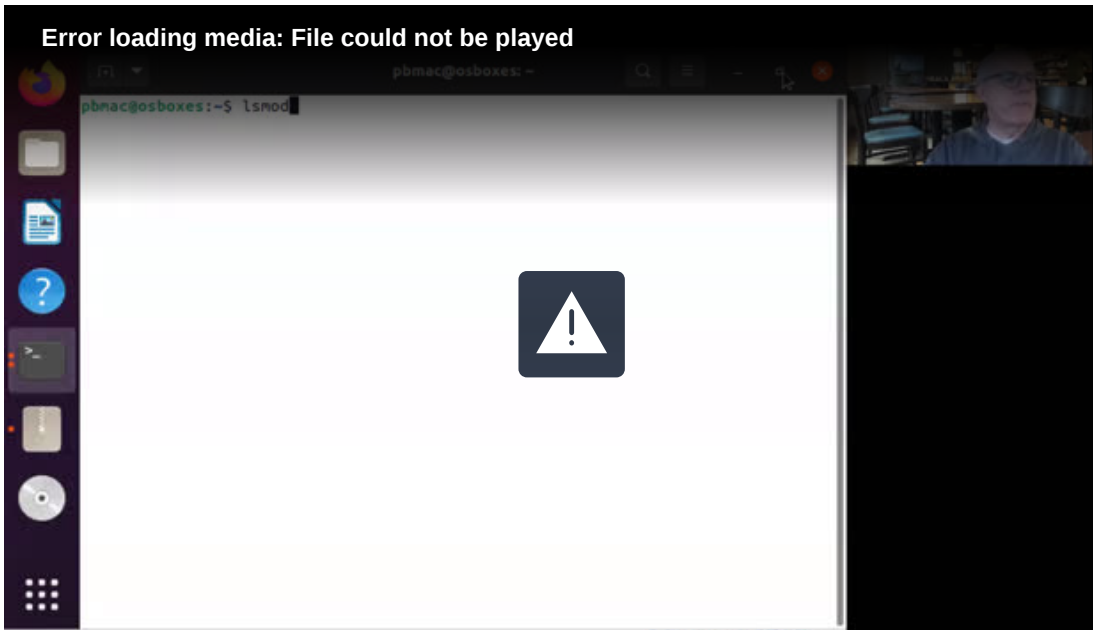
06-B.4: Kernel Modules is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-B.5: Kernel Module Management - lsmod/modinfo Command

The `lsmod` Command

The **`lsmod`** command is used to display the status of modules in the Linux kernel. It results in a list of loaded modules. lsmod is a trivial program which nicely formats the contents of the `/proc/modules` , showing what kernel modules are currently loaded.

This video covers the Linux kernel module commands. While the commands are not overly complex, there are some good examples of how to use them in the video.



Syntax:

```
lsmod
```

Example:

```
pbmac@pbmac-server $ lsmod
Module                  Size  Used by
hid_logitech_hidpp     36864  0
hid_logitech_dj        20480  0
vmw_vsock_vmci_transport    32768  0
vsock                  36864  1 vmw_vsock_vmci_transport
vmw_vmci               69632  1 vmw_vsock_vmci_transport
```

Output Format: There are three columns in output.

- First column is used for '**Name**' of the module file.
- Second column is used for '**size**' of the module in Bytes.
- The third column in output is for '**used by.**' It is basically used to display the number of instances of the module which are used.

The `modinfo` Command

The **`modinfo`** command is used to display the information about a Linux Kernel module. This command extracts the information from the Linux kernel modules given on the command line. If the module name is not a file name, then the **`/lib/modules/$(uname -r)`**directory is searched by default. **`modinfo`** can understand modules from any of the Linux Kernel architecture.

Syntax:

```
modinfo [ OPTION ] [modulename|filename...]
```

modinfo Command Options

Options	Options Meaning
-F, --field	Only print this field value, one per line. This is most useful for scripts. Field names are case-insensitive. Common fields (which may not be in every module) include author, description, license, parm, depends, and alias. There are often multiple parm, alias and depends fields. The special field filename lists the filename of the module.
-b basedir, --basedir basedir	Root directory for modules, / by default.
-k kernel	Provide information about a kernel other than the running one.
-l, --files-with-matches	Suppress normal output; instead print the name of each input file from which output would normally have been printed. The scanning will stop on the first match.
-o, --only-matching	Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.
-v, --invert-match	Invert the sense of matching, to select non-matching lines.
-f FILE, --file=FILE	Obtain patterns from FILE, one per line. If this option is used multiple times or is combined with the -e (--regexp) option, search for all patterns given. The empty file contains zero patterns, and therefore matches nothing.

Taking the name returned in the output of **`lsmod`**, it is possible to find out more about a particular module as seen below.

```
pbmac@pbmac-server $ lsmod
Module                  Size  Used by
...
vboxdrv               483328  2 vboxnetadp,vboxnetflt
pbmac@pbmac-server $ modinfo vboxdrv
filename:       /lib/modules/4.15.0-91-generic/misc/vboxdrv.ko
version:        6.1.6 r137129 (0x002d0001)
license:        GPL
description:    Oracle VM VirtualBox Support Driver
author:         Oracle Corporation
srcversion:     1B117C52DF5B4D7EFB983ED
depends:
retpoline:      Y
name:           vboxdrv
vermagic:       4.15.0-91-generic SMP mod_unload
parm:           force_async_tsc:force the asynchronous TSC mode (int)
```

Adapted from:
"lsmod command in Linux with Examples" by Narendramnitcse, Geeks for Geeks is licensed under CC BY-SA 4.0
"modinfo command in Linux with Examples" by Rahulkumarmandal, Geeks for Geeks is licensed under CC BY-SA 4.0

06-B.5: Kernel Module Management - lsmod/modinfo Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-B.5.1: Kernel Module Management - insmod/rmmod Command

The **insmod** Command

The **insmod** command is used to insert modules into the kernel. Kernel modules are usually used to add support for new hardware (as device drivers) and/or filesystems, or for adding system calls. This command inserts the kernel object file (.ko) into the kernel.

Syntax:

```
insmod [file name] [module-options...]
```

There are only 2 options - on -h for help, and -V fto show the version of **insmod**.

This example shows running the **insmod** command from the /lib/modules/$(uname -r) directory, and that the .ko file exists in the specified directory.

```
pbmac@pbmac-server $ insmod kernel/drivers/net/wireless/airo.ko
pbmac@pbmac-server $ lsmod | grep axnet
Module                  Size  Used by
airo                   66291  0
```

The **rmmod** Command

The **rmmod** command is used to remove a module from the kernel. Most of the users use **modprobe** with the -r option instead of using **rmmod**.

Syntax:

```
rmmod [-f] [-s] [-v] [modulename]
```

The **rmmod** command is extremely simple. You only need to give it the name of a module that you want to unload, and it will remove it. When you run the **lsmod** command looking for the module it should not find it. If you need the name, simply use **lsmod** to find the modules currently loaded, then remove the one you need.

```
pbmac@pbmac-server $ rmmod axnet
pbmac@pbmac-server $ lsmod | grep axnet
pbmac@pbmac-server $
```

Adapted from:
"insmod command in Linux with examples" by Mukkesh Mckenzie, Geeks for Geeks is licensed under CC BY-SA 4.0
"rmmod command in Linux with Examples" by rahulkumarmandal, Geeks for Geeks is licensed under CC BY-SA 4.0

06-B.5.1: Kernel Module Management - insmod/rmmod Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-B.5.2: Kernel Module Management - modprobe/depmod Command

The **modprobe** Command

The **modprobe** command offers more features than the more basic insmod and rmmod utilities. **modprobe** intelligently adds or removes a module from the Linux kernel. Note that for convenience, there is no difference between _ and - in module names (automatic underscore conversion is performed). **modprobe** looks in the module directory /lib/modules/`uname -r` for all the modules and other files, except for the optional configuration files in the /etc/modprobe.d directory (some distributions use the /etc/modprobe.conf file instead).

Syntax:

```
modprobe [ OPTIONS ] [modulename] [module parameters...]
```

Command Options:

Options	Option Meaning
-f --force	Tries to strip any versioning information from the module which might otherwise stop it from loading.
-n, --dry-run, --show	This option does everything but actually insert or delete the modules (or run the install or remove commands). Combined with **-v**, it is useful for debugging problems.
-s --syslog	This option causes any error messages to go through the syslog mechanism (as LOG_DAEMON with level LOG_NOTICE) rather than to standard error.
-v, --verbose	Prints messages about what the program is doing. Usually **modprobe** only prints messages if something goes wrong.
-r	This option causes **modprobe** to remove rather than insert a module.

With **modprobe**, you specify a module by its module name rather than the module filename. You can pass a few options to modify **modprobe**'s behavior. For example, to perform checks and all other operations except the actual module insertions, you can use the -n option. Combined with the -v (verbose) option, this option is useful for debugging problems. The --show-depends option will list ALL the modules that are dependent upon the specified module.

```
pbmac@pbmac-server $ modprobe --show-depends anubis
insmod /lib/modules/4.15.0-91-generic/kernel/crypto/anubis.ko
pbmac@pbmac-server $ modprobe anubis
pbmac@pbmac-server $
```

The **depmod** Command

The **depmod** (**Dep**endency **Mod**ules) command is used to generate a list of dependency description of kernel modules and its associated map files. This analyzes the kernel modules in the directory /lib/modules/kernel-release and creates a "Makefile"-like dependency file named modules.dep based on the symbols present in the set of modules. These modules are generally taken from the directories specified in the configuration file or mentioned on the command line. Then when the stack of modules are added and removed automatically with **modprobe**, no modules are without the other related modules they require. Simultaneously, it creates an associated map correlating the hardware identifiers and the corresponding modules that handle them for the purpose of use by

the hotplug infrastructure. This specifically associated mapping is used to search for and find the correct module when a unit of hardware requests it.

Syntax:

```
depmod [ OPTIONS ]
```

Command Options:

Options	Option Meaning
-a --all	Probes all modules. This option is enabled by default if no file names are given in the command-line.
-A --quick	This option scans to see if any modules are newer than the modules.dep file before any work is done; if not, it silently exits rather than regenerating the files.
-b basedir --basedir basedir	If your modules are not currently in the (normal) directory /lib/modules/version, but in a staging area, you can specify a basedir which is prepended to the directory name. This basedir is stripped from the resulting modules.dep file, so it is ready to be moved into the normal location.
-C --config file or directory	This option overrides the default configuration file at /etc/depmod.conf (or the /etc/depmod.d/ directory if that is not found).
-e --errsyms	When combined with the -F option, this reports any symbols which a module needs which are not supplied by other modules or the kernel. Normally, any symbols not provided by modules are assumed to be provided by the kernel (which should be true in a perfect world).
-F --filesyms System.map	Supplied with the System.map produced when the kernel was built, this allows the -e option to report unresolved symbols.
-n --dry-run	This sends the resulting modules.dep and the various map files to standard output rather than writing them into the module directory.

By default **depmod** looks for the .ko files in /lib/modules/$(uname -r), which is why in the following example a symbolic link is created. Once that file can be found the **depmod** will perform its work, then if necessary run the **modprobe** command to install the module.

```
pbmac@pbmac-server $ ln -s /lib/modules/4.15.0-91-generic/kernel/crypto/md4.ko /lib/mo
pbmac@pbmac-server $ depmod -a
pbmac@pbmac-server $ modprobe md4
```

Linux Kernel Symbols

Kernel symbols are names of functions and variables. Global symbols are those which are available outside the file they are declared in. Global symbols in the Linux kernel currently running on a system are available through `/proc/kallsyms` file. This includes symbols defined inside kernel modules currently loaded.

Global symbols are of two types: those explicitly exported through EXPORT_SYMBOL_GPL and EXPORT_SYMBOL macros, and those which are not declared with `static` C keyword, and hence visible to code which is statically linked with the kernel itself and may be available outside the kernel image.

The first type, explicitly exported ones, are denoted with a capital letter in output of `cat /proc/kallsyms` – e.g., T if the symbol is in text section, i.e. a function name. The second type are denoted with a lowercase letter – e.g., t for a function which isn't exported via EXPORT_SYMBOL_GPL or EXPORT_SYMBOL.

Adapted from:
"depmod command in Linux with examples" by Merlyn Shelley, Geeks for Geeks is licensed under CC BY-SA 4.0

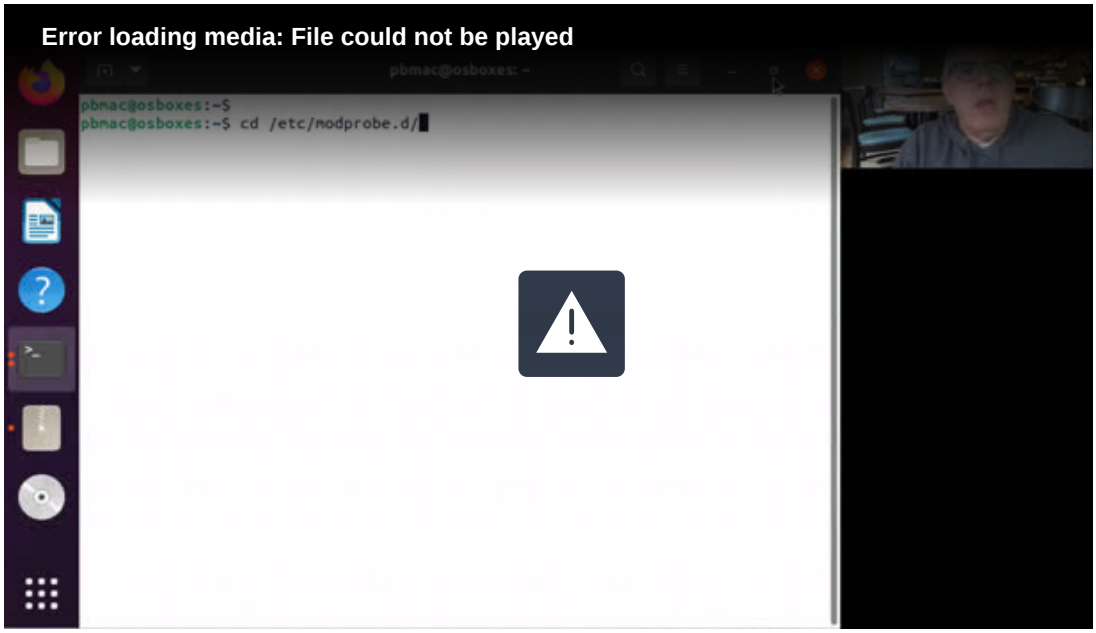
06-B.5.2: Kernel Module Management - modprobe/depmod Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-B.6: Kernel Modules - Configuration

Kernel Module Configuration

The **modprobe** command can add or remove modules. Since modules have dependencies, we need a method of specifying what options are to be used with those modules. All .conf files within the /etc/modprobe.d directory extension specify the options required for the various modules. The configuration files can be used to create aliases (alternate names for a module), or they can override the normal **modprobe** behavior altogether for those with special requirements. The default location of the .conf files are in the **/etc/modprobe.d** directory. In older versions of Linux these configurations were in the **/etc/modprobe.conf** file.

This video discusses a bit about the /etc/modprobe.d directory and the dmesg command, which we will see in a couple of pages.



Loadable modules can be defined in .conf files in the **/etc/modules.d** directory in order to load them during the init portion of the system boot process. Each module is listed one per-line. For example the **/etc/modules.d/networking.conf** file contains a list of 3 modules that should be loaded at boot time.

```
tun
e1000e
brcmfmac
```

To avoid a module from loading, add it by name to a file in **/etc/modprobe.d/** and precede each module name with the blacklist keyword:

```
blacklist uhci_hcd
blacklist nvidia
```

There is also the capability to create aliases for module names. Aliases are alternate names for a module. In the following example we have a **/etc/modprobe.d/myalias.conf** file that contains a very short alias mymod, instead of the long module name.

```
alias mymod really_long_module_name
```

Adapted from:
"Kernel modules" by Multiple contributors, Gentoo is licensed under CC BY-SA 3.0

06-B.6: Kernel Modules - Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-B.7: Kernel Modules - Kernel Parameters

Kernel Parameters

The /proc/sys directory contains a lot of useful and interesting files and directories. Many kernel settings can be manipulated by writing to files in the proc filesystem. A lot of important information can be retrieved from these files. This is especially useful when you are troubleshooting or fine tuning your Linux system.

The /proc/sys directory is divided into subdirectories:

Subdirectory	What it is
abi/	Execution domains & personalities
debug/	<empty>
dev/	Device specific information (eg dev/cdrom/info)
fs/	Specific filesystems filehandle, inode, dentry and quota tuning binfmt_misc <Documentation/admin-guide/binfmt-misc.rst>
kernel/	Global kernel info / tuning miscellaneous stuff
net/	Networking stuff, for documentation look in: <Documentation/networking/>
vm/	Memory management tuning buffer and cache management
user/	Per user per user namespace limits

The `sysctl` Command

The **`sysctl`** is a software utility that reads and modifies the attributes of the system kernel such as its version number, maximum limits, and security settings. It is available both as a system call for compiled programs, and an administrator command for interactive use and scripting.

Syntax:

```
sysctl [ OPTIONS ]
```

Options	Options Meaning
-a, --all	Display all values currently available.
-e, --ignore	Use this option to ignore errors about unknown keys.
-p [FILENAME}, --load[=FILENAME]	Load in sysctl settings from the file specified or /etc/sysctl.conf if none given.
-r, --pattern pattern	Only apply settings that match *pattern*.
-w, --write	Use this option when you want to change a sysctl setting.

To see what the attributes look like use the -a option and the command will output ALL of the currently configured parameters. There are many more than the few shown in the example - the system that this was run on had a total of 936 parameters.

```
pbmac@pbmac-server $ sysctl -a
abi.vsyscall32 = 1
debug.exception-trace = 1
```

```
debug.kprobes-optimization = 1
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
dev.cdrom.check_media = 0
dev.cdrom.debug = 0
dev.cdrom.info = CD-ROM information, Id: cdrom.c 3.20 2003/12/17
```

The sysctl.conf File

The **`sysctl`** preload/configuration file can be created at /etc/sysctl.d/99-sysctl.conf. Current Linux distributions no longer maintain the **`/etc/sysctl.conf file.`** All of the configurations are contained in individual files in the **`/etc/sysctl.d/`** directory.

Settings can be changed through file manipulation or using the sysctl utility. ForThis will change the current value of the parameter until the system is rebooted.

```
pbmac@pbmac-server $ sysctl kernel.sysrq=1
```

To preserve changes between reboots, add or modify the appropriate lines in /etc/sysctl.d/99-sysctl.conf or create an applicable parameter file in /etc/sysctl.d/.

Adapted from:
"Sysctl" by Multiple contributors, ArchWiki is licensed under CC BY-SA 4.0

06-B.7: Kernel Modules - Kernel Parameters is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-C.8: Kernel Module Monitoring

EXAM OBJECTIVES COVERED

1.2 Given a scenario, install, configure, and monitor kernel modules.
2.7 Explain the use and operation of Linux devices.

The /proc Directory

/proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g., system memory, devices mounted, hardware configuration, etc). For this reason, it can be regarded as a control and information center for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory. For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

The most distinctive thing about files in this directory is the fact that all of them have a file size of 0, with the exception of kcore, mtrr and self.

Directories	Contents
/proc/cmdline	Options passed to the kernel by the boot loader at boot time, such as mounting the kernel as read-only.
/proc/cpuinfo	Information about the processor, such as its type, make, model, and performance.
/proc/devices	List of device drivers configured into the currently running kernel.
/proc/filesystems	Filesystems configured into the kernel.
/proc/interrupts	Shows which interrupts are in use, and how many of each there have been.
/proc/ioports	Which I/O ports are in use at the moment.
/proc/meminfo	Information about memory usage, both physical and swap.
/proc/modules	Which kernel modules are loaded at the moment.
/proc/net	Status information about network protocols.
/proc/stat	Various statistics about the system, such as the number of page faults since the system was booted.
/proc/uptime	The length of time the kernel has been running since boot and spent in idle mode (both in seconds).

In Linux, you can also find the kernel information in the file **`/proc/version.`** Just look at the contents of this file:

```
pbmac@pbmac-server $ cat /proc/version
Linux version 4.15.0-91-generic (buildd@lgw01-amd64-013) (gcc version 7.4.0 (Ubuntu 7
#92-Ubuntu SMP Fri Feb 28 11:09:48 UTC 2020
```

You'll see an output similar to what you saw with **`uname`**. Remember what all the digits mean here? Here is an explanation:

- 4 – Kernel version

- 15 – Major revision
- 0 – Minor revision
- 91 – Bug fix
- generic – Distribution-specific string. For Ubuntu, it means I'm using the desktop version. For Ubuntu server edition, it would be 'server.'

There is other information in this output as well:

- Linux – Kernel name. If you run the same command on BSD or MacOS, the result will be different.
- 4.15.0-91-generic – Kernel release (as we saw above).
- buildd@lgw01-amd64-013 – Username who compiled the kernel.
- 7.4.0 – GCC version (GCC is the C compiler used to compile the kernel).
- #92-Ubuntu SMP Fri Feb 28 11:09:48 UTC 2020 – This means that Ubuntu has compiled 4.15.0-91-generic 92 times. A timestamp for the last compilation is also there.

Adapted from:
"3 Ways to Check Linux Kernel Version in Command Line" by Abhishek Prakash, It's FOSS is licensed under CC BY-SA 4.0
"1.14. /proc" by Multiple contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0
"3.7. The /proc filesystem" by Multiple contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0

06-C.8: Kernel Module Monitoring is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

06-C.9: The dmesg Command

The **dmesg** Command

dmesg (there are several sources that disagree on the name: "diagnostic message," “driver message” or “display message” are all options) is a command that prints the message buffer of the kernel. The output of this command typically contains the messages produced by the device drivers.

On most Linux systems, the boot process generates a particularly dense stream of kernel messages. Many administrative issues pertain to whether a desired hardware device is successfully enumerated during the boot process, so the diagnostic process for a failed device often begins by inspecting the **dmesg** output from the kernel identification message to the point where the boot process concludes. Since this buffer can be overwritten by a flood of messages in subsequent operation, many Unix-like distributions store a post-boot copy of the message buffer at **/var/log/dmesg** or similar secure system location.

It is also common to manually consult the current **dmesg** buffer after hot-plugging devices, particularly USB devices (especially thumb drives), to determine whether the device has been recognized, the data rate of the port involved (USB 2 and USB 3.0 plugs sit side by side and are hard to distinguish on many systems), what driver has been assigned, and where the device is made visible in the file system. Many distributions attempt to display device-recognition messages on the desktop, often through a taskbar pop-up, but this is not always reliable or the information presented is incomplete. (Furthermore, to be notified on the desktop, the hot-plugged device must be permitted by the system's security policy.)

Syntax:

```
dmesg [ OPTIONS ]
```

Options:

Option Choice	Meaning
-c, --read-clear	Clear the ring buffer after first printing its contents.
-e, --reltime	Display the local time and the delta in human-readable format. Be aware that conversion to the local time could be inaccurate (see -T for more details).
-f, --facility list	Restrict output to the given (comma-separated) list of facilities.
-H, --human	Enable human-readable output. See also --color, --reltime and --nopager.
-L, --color[=when]	Colorize the output. The optional argument when can be auto, never or always. If the when argument is omitted, it defaults to auto. The colors can be disabled; for the current built-in default see the --help output. See also the COLORS section below.
-l, --level list	Restrict output to the given (comma-separated) list of levels.

There is a LOT of information that is output from dmesg, and most of it takes a a bit of deciphering, but if you are looking for a specific device that is failing or having issues:

```
pbmac@pbmac-server $ dmesg -H
[Aug 4 12:58] Linux version 4.15.0-91-generic (buildd@lgw01-amd64-013) (gcc version
[  +0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.15.0-91-generic root=UUID=6e7
[  +0.000000] KERNEL supported cpus:
```

```
[  +0.000000]   Intel GenuineIntel
[  +0.000000]   AMD AuthenticAMD
[  +0.000000]   Centaur CentaurHauls
[  +0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers
[  +0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[  +0.000000] x86/fpu: Enabled xstate features 0x3, context size is 576 bytes, using
[  +0.000000] e820: BIOS-provided physical RAM map:
[  +0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009a7ff] usable
[  +0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[  +0.000000] BIOS-e820: [mem 0x0000000000100000-0x00000000bfdffbff] usable
[  +0.000000] BIOS-e820: [mem 0x00000000bfdffc00-0x00000000bfe53bff] ACPI NVS
[  +0.000000] BIOS-e820: [mem 0x00000000bfe53c00-0x00000000bfe55bff] ACPI data
[  +0.000000] BIOS-e820: [mem 0x00000000bfe55c00-0x00000000bfffffff] reserved
[  +0.000000] BIOS-e820: [mem 0x00000000f8000000-0x00000000fbffffff] reserved
[  +0.000000] BIOS-e820: [mem 0x00000000fec00000-0x00000000fed003ff] reserved
[  +0.000000] BIOS-e820: [mem 0x00000000fed20000-0x00000000fed9ffff] reserved
[  +0.000000] BIOS-e820: [mem 0x00000000fee00000-0x00000000feefffff] reserved
[  +0.000000] BIOS-e820: [mem 0x00000000ffb00000-0x00000000ffffffff] reserved
[  +0.000000] BIOS-e820: [mem 0x0000000100000000-0x000000033bffffff] usable
[  +0.000000] NX (Execute Disable) protection: active
:
```

Many **dmesg** lines in a traditional system begin with a device name followed by a colon, followed by some detailed text. Often these come in clusters, with the same device showing up on multiple lines in succession. Each cluster is usually associated with a single device enumeration, by one particular device driver (or device facility) associated with the device name.

Adapted from:
"dmesg" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

06-C.9: The dmesg Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 07: Linux Boot Process

07-A.1: Configuring Boot Components

07-A.2: Configuring Boot Components (continued)

07-A.3: Configuring initrd

07-A.4: Boot Process

07-B.5: GRUB2 - Installation

07-B.6: GRUB2 - Configuration

Module 07: Linux Boot Process is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

07-A.1: Configuring Boot Components

EXAM OBJECTIVES COVERED

1.1 Explain Linux boot process concepts

1.4 Given a scenario, manage storage in a Linux environment

Module Objectives

The learning objectives for this module are as follows:

- Explain how to configure boot components in Linux
- Configure initial ramdisk to load a temporary root file system into memory
- Describe the components of the Linux boot process
- Recognize and troubleshoot issues related to a kernel panic during the boot process

The Boot Process

The boot process can be initiated in one of a couple ways. First, if power is turned off, turning on the power will begin the boot process. If the computer is already running a local user, including root or an unprivileged user, the user can programmatically initiate the boot sequence by using the GUI or command line to initiate a reboot. A reboot will first do a shutdown and then restart the computer.

Booting a Linux installation involves multiple stages and software components, including firmware initialization, execution of a boot loader, loading and startup of a Linux kernel image, and execution of various startup scripts and daemons. For each of these stages and components there are different variations and approaches.

The Boot Loader

A boot loader is a small, simple program responsible for loading the Linux kernel with optional kernel parameters and the Linux initial RAM disk, known as initrd. Linux kernel is the core of the Linux operating system, and it starts the init (short for initialization) process, or an init replacement such as systemd, immediately after being loaded. The Linux initial RAM disk provides a temporary storage space for loading critical files into memory before the real root file system can be mounted.

The boot loader phase varies by computer architecture. Since the earlier phases are not specific to the operating system, the BIOS-based boot process for x86 and x86-64 architectures is considered to start when the master boot record (MBR) code is executed in real mode and the first-stage boot loader is loaded. In UEFI systems, a payload, such as the Linux kernel, can be executed directly.

GRUB 2 has two (optionally three) stages and is capable of automatic detection of various operating systems and automatic configuration. The first-stage loader (stage1) is loaded and executed either by the BIOS from the master boot record (MBR) or by another boot loader from the partition boot sector. Its job is to discover and access various file systems that the configuration can be read from later. The optional, intermediate stage loader (stage1.5) is loaded and executed by the first-stage loader in case the second-stage loader is not contiguous, or if the file-system or hardware requires special handling in order to access the second-stage loader. The second-stage loader (stage2) is loaded last and displays the GRUB startup menu that allows the user to choose an operating system or examine and edit startup parameters. After a menu entry is chosen and optional parameters are given, GRUB loads the kernel into memory and passes control to it.

The BIOS

BIOS is an acronym for Basic Input/Output System. It is firmware used to perform hardware initialization during the booting process (power-on startup), and to provide runtime services for operating systems and programs.

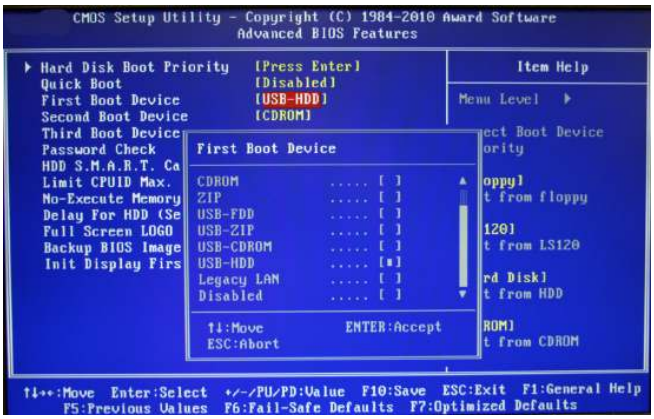


Figure **7.1.1**: BIOS Screen. ("BIOS Settings" by Patrick McClanahan is licensed under CC BY-SA 4.0)

The BIOS firmware comes pre-installed on a personal computer's system board, and it is the first software to run when powered on.

The BIOS in modern PCs initializes and tests the system hardware components, and loads a boot loader from a mass memory device which then initializes an operating system. In the era of DOS, the BIOS provided a hardware abstraction layer for the keyboard, display, and other input/output (I/O) devices that standardized an interface to application programs and the operating system. More recent operating systems do not use the BIOS after loading, instead accessing the hardware components directly.

UEFI

The Unified Extensible Firmware Interface (UEFI) is a specification that defines a software interface between an operating system and platform firmware. UEFI replaces the legacy Basic Input/Output System (BIOS) firmware interface originally present in all IBM PC-compatible personal computers,[1][2] with most UEFI firmware implementations providing support for legacy BIOS services. UEFI can support remote diagnostics and repair of computers, even with no operating system installed.

Just like the BIOS, UEFI is an environment in which to execute a boot loader, and both eventually execute the operating system.

Server Security

As long as we are discussing BIOS/UEFI, its a good time to mention the idea of password protecting the boot sequence. Many Linux system administrators make use of the capability that both BIOS and UEFI have to put a password on the boot sequence. This prevents someone from gaining access to a corporate server, simply rebooting it and taking control. Sometimes it takes a bit of searching around in the BIOS/UEFI menu to find the place to enter a password. As there is greater focus being placed on security - both hardware and software - this is something to consider.

Adapted from:
"Linux startup process" by Multiple contributors, Wikipedia is licensed under CC BY-SA 3.0
"BIOS" by Multiple contributors, Wikipedia is licensed under CC BY-SA 3.0
"Unified Extensible Firmware Interface" by Multiple contributors, Wikipedia is licensed under CC BY-SA 3.0

07-A.1: Configuring Boot Components is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

07-A.2: Configuring Boot Components (continued)

Boot Source Options

In addition to the normal idea of booting from the system's hard drive, there are other options that are used in various situations.

Boot source	Description
ISO	Most Linux distributions can be downloaded as an ISO image. These are images of an actual Linux system. It is possible to boot directly from an ISO file that resides on a hard disk, or on a USB device. It is also possible to view the file system found on an ISO image using an archiving utility.
Network File System (NFS)	Network File Systems (NFS) can be used as a source of the boot files. This simply means that the boot files are not physically loaded on the machine being booted. The files are accessed across the network. This method must be set up both on the local system and the remote NFS server where the files are located. This boot method makes use of NFS, tftp or other network protocols to retrieve the necessary files.
HTTP/FTP	HTTP Boot combines DHCP, DNS, and HTTP to make it possible to boot and deploy systems over the network. HTTP Boot can be used as a high-performance replacement for PXE. HTTP Boot allows you to boot a server from a URI over HTTP, quickly transferring large files such as the Linux kernel and root file system from servers outside of your local network.
PXE	The Preboot eXecution Environment (PXE, also known as Pre-Execution Environment; sometimes pronounced "pixie") is an environment to boot computers using a network interface independently of data storage devices (like hard disks) or installed operating systems.

What are Sectors

When we discuss booting and MBRs we often mention "sectors." The sector is the minimum storage unit of a hard drive. Most disk partitioning schemes are designed to have files occupy an integral number of sectors regardless of the file's actual size. Files that do not fill a whole sector will have the remainder of their last sector filled with zeroes. In practice, operating systems typically operate on blocks of data, which may span multiple sectors. In the image to the right, 'C' points to a sector. Notice that sectors get bigger the closer they get to the outer edge of the disk (of course we are moving towards solid state devices now - this may become a moot point).

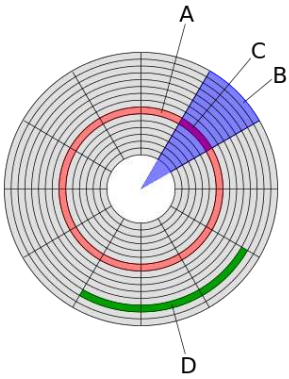


Figure **7.2.1**: Disk structures:

A. Track

B. Geometrical sector

C. Track sector

D. Cluster

("disk sectors" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0)

Master Boot Record

A master boot record (MBR) is a special type of boot sector at the very beginning of partitioned computer mass storage devices like fixed disks or removable drives. The MBR holds the information on how the logical partitions, containing file systems, are organized on that medium. The MBR also contains executable code to function as a loader for the installed operating system—usually by passing control over to the loader's second stage. This MBR code is usually referred to as a boot loader.[1]

The organization of the partition table in the MBR limits the maximum addressable storage space of a partitioned disk to 2 TiB (232 × 512 bytes). Approaches to slightly raise this limit assuming 32-bit arithmetic or 4096-byte sectors are not officially supported, as they fatally break compatibility with existing boot loaders and most MBR-compliant operating systems and system tools, and can cause serious data corruption when used outside of narrowly controlled system environments. Therefore, the MBR-based partitioning scheme is in the process of being superseded by the GUID Partition Table (GPT) scheme in new computers. A GPT can coexist with an MBR in order to provide some limited form of backward compatibility for older systems.

The GUID Partition Table

The GUID Partition Table (GPT) is a standard for the layout of partition tables of a physical computer storage device such as a hard disk drive or solid-state drive, using universally unique identifiers, which are also known as globally unique identifiers (GUIDs). Forming a part of the UEFI standard, it is nevertheless also used for some BIOS systems because of the limitations of master boot record (MBR) partition tables, which use 32 bits for logical block addressing (LBA) of traditional 512-byte disk sectors.

All modern personal computer operating systems support GPT. Some, including macOS and Microsoft Windows on the x86 architecture, support booting from GPT partitions only on systems with EFI firmware, but most Linux distributions can boot from GPT partitions on systems with either firmware interface: the legacy BIOS or the modern EFI.

Adapted from:
"Disk sector" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Master boot record" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"GUID Partition Table" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

07-A.2: Configuring Boot Components (continued) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

07-A.3: Configuring initrd

initrd

`initrd` (initial ramdisk) is a scheme for loading a temporary root file system into memory, which may be used as part of the Linux startup process. **`initrd`** and **`initramfs`** refer to two different methods of achieving this. Both are commonly used to make preparations before the real root file system can be mounted.

Many Linux distributions ship a single, generic Linux kernel image – one that the distribution's developers create specifically to boot on a wide variety of hardware. The device drivers for this generic kernel image are included as loadable kernel modules because statically compiling many drivers into one kernel causes the kernel image to be much larger, perhaps too large to boot on computers with limited memory. This then raises the problem of detecting and loading the modules necessary to mount the root file system at boot time, or for that matter, deducing where or what the root file system is.

An image of this initial root file system (along with the kernel image) must be stored somewhere accessible by the Linux bootloader or the boot firmware of the computer. This can be the root file system itself, a boot image on an optical disc, a small partition on a local disk (a boot partition, usually using ext2 or FAT file systems), or a TFTP server (on systems that can boot from Ethernet).

The bootloader will load the kernel and initial root file system image into memory and then start the kernel, passing in the memory address of the image. At the end of its boot sequence, the kernel tries to determine the format of the image from its first few blocks of data, which can lead either to the initrd or initramfs scheme.

The **`mkinitrd`** Command

The **`mkinitrd`** command creates an initial image used by the kernel for preloading the block device modules (such as IDE, SCSI or RAID) which are needed to access the root filesystem.

Syntax:

```
mkinitrd [ OPTIONS ] [<nitrd-image] <kernel-version>
```

Command Options:

Options	Meaning
--with=<module>	Add the kernel module <module> to the initramfs.
--preload=<module>	Preload the kernel module <module> in the initramfs before any other kernel modules are loaded.
--nocompress	Do not compress the resulting image.
-f, --force	Overwrite existing initramfs file.

We have seen the idea of using the uname command to help us issue commands before. We use it in the following example because we want the resulting image to have a definitive name associating it with the current kernel.

```
pbmac@pbmac-server $ mkinitrd -f /boot/initrd-`uname -r`-new.img `uname -r`
```

The /boot Directory

This directory contains everything required for the boot process except for configuration files not needed at boot time (the most notable of those being those that belong to the GRUB boot-loader) and the map installer. Thus, the /boot directory stores data that is used before the kernel begins executing user-mode programs. This may include redundant (back-up) master boot records, sector/system map files, the kernel and other important boot files and data that is not directly edited by hand. Programs necessary to

arrange for the boot loader to be able to boot a file are placed in /sbin. Configuration files for boot loaders are placed in /etc. The system kernel is located in either / or /boot

/boot Files and Directories	Description
/boot/initrd.img-<kernel version>	In the initrd scheme, the image may be a file system image (optionally compressed), which is made available in a special block device (/dev/ram) that is then mounted as the initial root file system. The driver for that file system must be compiled statically into the kernel.
/boot/initramfs-<kernel version>.img	Some Linux distributions use initramfs as an alternative to initrd. In the initramfs scheme, the image may be a cpio archive (optionally compressed). The archive is unpacked by the kernel into a special instance of a tmpfs that becomes the initial root file system. This scheme has the advantage of not requiring an intermediate file system or block drivers to be compiled into the kernel.
/boot/efi	The EFI (Extensible Firmware Interface) system partition or ESP is a partition on a data storage device (usually a hard disk drive or solid-state drive) that is used by computers adhering to the Unified Extensible Firmware Interface (UEFI). When a computer is booted, UEFI firmware loads files stored on the ESP to start installed operating systems and various utilities.
/boot/grub	Contains the configuration files for the GRUB bootloader. These files are mostly modules (.mod), with configuration stored in grub.cfg.
vmlinux-<kernel version>	On Linux systems, vmlinux is a statically linked executable file that contains the Linux kernel in one of the object file formats supported by Linux, which includes ELF, COFF and a.

What is `dracut`?

As you begin to work with creating and modifying kernels you will find references to the **`dracut`** command in some of the man pages. **`dracut`** is an event driven initramfs infrastructure. **`dracut`** (the tool) is used to create an initramfs image by copying tools and files from an installed system and combining it with the dracut framework, usually found in /usr/lib/dracut/modules.d.

Unlike existing initramfs's, dracut's framework attempts at having as little as possible hard-coded into the initramfs. The initramfs has (basically) one purpose in life -- getting the rootfs mounted so that we can transition to the real rootfs. This is all driven off of device availability. Therefore, instead of scripts hard-coded to do various things, dracut's initramfs depends on udev to create symbolic links to device nodes and then when the rootfs's device node appears, it is mounted and root is switched to it. This helps to keep the time required in the initramfs as minimal as possible so that things like a five-second boot aren't made impossible as a result of the very existence of an initramfs.

Adapted from:
"Initial ramdisk" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"/boot/" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"EFI system partition" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"dracut" by Various contributors, The Linux Kernel Archives is licensed under CC BY-SA 3.0

07-A.3: Configuring initrd is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

07-A.4: Boot Process

The Boot Process

The Linux startup process is the multi-stage initialization process performed during booting a Linux installation. It is in many ways similar to the BSD and other Unix-style boot processes, from which it derives.

Booting a Linux installation involves multiple stages and software components, including firmware initialization, execution of a boot loader, loading and startup of a Linux kernel image, and execution of various startup scripts and daemons.

1. The BIOS performs startup tasks specific to the actual hardware platform. The hardware is enumerated and the hardware which is necessary for boot is initialized. Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
2. The MBR / GPT loads and executes the boot code from the configured boot device. The MBR / GPT loads and executes the GRUB boot loader.
3. The boot loader (GRUB2) often presents the user with a menu of possible boot options and has a default option, which is selected after some time passes. Once the selection is made, the boot loader loads the kernel into memory, supplies it with some parameters and gives it control.
4. The kernel, if compressed, will decompress itself. It then sets up system functions such as essential hardware and memory paging, and calls start_kernel() which performs the majority of system setup (interrupts, the rest of memory management, device and driver initialization, etc.). It then starts up, separately, the idle process, scheduler, and the init process, which is executed in user space.
5. The init either consists of scripts that are executed by the shell (sysv, bsd, runit) or configuration files that are executed by the binary components (systemd, upstart). Init has specific levels (sysv, bsd) or targets (systemd), each of which consists of specific set of services (daemons). These provide various non-operating system services and structures and form the user environment. A typical server environment starts a web server, database services, and networking.
6. The typical desktop environment begins with a daemon, called the display manager, that starts a graphic environment which consists of a graphical server that provides a basic underlying graphical stack and a login manager that provides the ability to enter credentials and select a session. After the user has entered the correct credentials, the session manager starts a session. A session is a set of programs such as UI elements (panels, desktops, applets, etc.) which, together, can form a complete desktop environment.

What is a Kernel Panic

A kernel panic is a safety measure taken by an operating system's kernel upon detecting an internal fatal error, in which it either is unable to safely recover or cannot have the system continue to run without having a much higher risk of major data loss. The term is largely specific to Unix and Unix-like systems.

A panic may occur as a result of a hardware failure or a software bug in the operating system. In many cases, the operating system is capable of continued operation after an error has occurred. However, the system is in an unstable state and rather than risking security breaches and data corruption, the operating system stops to prevent further damage and facilitate diagnosis of the error and, in usual cases, restart.

After recompiling a kernel binary image from source code, a kernel panic while booting the resulting kernel is a common problem if the kernel was not correctly configured, compiled or installed. Add-on hardware or malfunctioning RAM could also be sources of fatal kernel errors during start up, due to incompatibility with the OS or a missing device driver. A kernel may also go into panic() if it is unable to locate a root file system. During the final stages of kernel userspace initialization, a panic is typically triggered if the spawning of init fails. A panic might also be triggered if the init process terminates, as the system would then be unusable.

Adapted from:
"Linux startup process" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Kernel panic" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

07-A.4: Boot Process is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

07-B.5: GRUB2 - Installation

GNU GRUB

GNU GRUB (short for GNU GRand Unified Bootloader, commonly referred to as GRUB) is a boot loader package from the GNU Project. GRUB is the reference implementation of the Free Software Foundation's Multiboot Specification, which provides a user the choice to boot one of multiple operating systems installed on a computer or select a specific kernel configuration available on a particular operating system's partitions.

GNU GRUB was developed from a package called the Grand Unified Bootloader (a play on Grand Unified Theory). It is predominantly used for Unix-like systems. The GNU operating system uses GNU GRUB as its boot loader, as do most Linux distributions and the Solaris operating system on x86 system.

Startup on systems using BIOS firmware:

- 1st stage: boot.img is written to the first 440 bytes of the Master Boot Record (MBR in sector 0), or optionally in a partition's boot sector (PBR / VBR). It addresses diskboot.img by a 64-bit LBA address, thus it can load from above the 2 GiB limit of the MBR. The actual sector number is written by grub-install.
- 2nd stage: diskboot.img is the first sector of core.img (called stage 1.5 in GRUB Legacy) with the sole purpose to load the rest of core.img identified by LBA sector numbers also written by grub-install.
- On MBR partitioned disks: core.img is stored in the empty sectors (if available) between the MBR and the first partition. Recent operating systems suggest a 1 MiB gap here for alignment (2047*512 byte or 255*4KiB sectors). This gap used to be 62 sectors (31 KiB) as a reminder of the sector number limit of C/H/S addressing used by Bios before 1998, therefore core.img is designed to be smaller than 32 KiB.
- On GPT partitioned disks: partitions are not limited to 4, thus core.img is written to its own tiny (1 MiB), filesystem-less BIOS boot partition.
- 3rd stage: core.img enters 32-bit protected mode, uncompresses itself (the kernel of grub and filesystem modules to reach /boot/grub), then loads /boot/grub/<platform>/normal.mod from the partition configured by grub-install. If the partition index has changed, GRUB will be unable to find the normal.mod, and presents the user with the GRUB Rescue prompt, where the user “can” find and load normal.mod, or the linux kernel.
- The /boot/grub directory can be located on any partition (GRUB can read many filesystems, including NTFS). Depending on how it was installed it's either in the root partition of the distribution, or a separate /boot partition.
- 4th stage: normal.mod (equivalent of stage 2 in GRUB Legacy) parses /boot/grub/grub.cfg, optionally loads modules (eg. for graphical UI) and shows the menu.

Startup on systems using UEFI firmware:

- Common on motherboards since ca. 2012.
- /efi/<distro>/grubx64.efi is installed as a file in the EFI System Partition, and booted by the firmware directly, without a boot.img in sector 0.
- /boot/grub/ can be installed on the EFI System Partition as well.

After startup

GRUB presents a menu where the user can choose from operating systems (OS) found by grub-install. GRUB can be configured to automatically load a specified OS after a user-defined timeout. If the timeout is set to zero seconds, pressing and holding ⇧ Shift while the computer is booting makes it possible to access the boot menu.[9]

In the operating system selection menu GRUB accepts a couple of commands:

- By pressing e, it is possible to edit kernel parameters of the selected menu item before the operating system is started. The reason for doing this in GRUB (i.e. not editing the parameters in an already booted system) can be an emergency case: the system has failed to boot. Using the kernel parameters line it is possible, among other things, to specify a module to be disabled (blacklisted) for the kernel. This could be required if the specific kernel module is broken and thus prevents boot-up. For example, to blacklist the kernel module nvidia-current, one could append modprobe.blacklist=nvidia-current at the end of the kernel parameters.

- By pressing c, the user enters the GRUB command line. The GRUB command line is not a regular Linux shell, like e.g. bash, and accepts only certain GRUB-specific commands, documented by various Linux distributions.[10]

Once boot options have been selected, GRUB loads the selected kernel into memory and passes control to the kernel. Alternatively, GRUB can pass control of the boot process to another boot loader, using chain loading. This is the method used to load operating systems that do not support the Multiboot Specification or are not supported directly by GRUB.

The `grub2-install` Command

`grub2-install` installs GRUB onto a device. This includes copying GRUB images into the target directory (generally */boot/grub*), and on some platforms may also include installing GRUB onto a boot sector. If grub2 is being installed on a UEFI system make sure to use the **`--efi-directory`** option so the files get placed in the proper directory.

Syntax:

grub-install [OPTIONS]... INSTALL_DEVICE

Notice the command is grub-install even though it installs grub2.

Command Options:

Option	Option Meaning
--modules=MODULES	Pre-load modules specified by MODULES.
--install-modules=MODULES	Install only MODULES and their dependencies. The default is to install all available modules.
--themes=THEMES	Install THEMES. The default is to install the starfield theme, if available.
--fonts=FONTS	Install FONTS. The default is to install the unicode font.
--locales=LOCALES	Install only locales listed in LOCALES. The default is to install all available locales.
--compress=no,xz,gz,lzo	Compress GRUB files using the specified compression algorithm.
--directory=DIR	Use images and modules in DIR.
--grub-mkimage=FILE	Use FILE as grub-mkimage. The default is /usr/bin/grub-mkimage.
--boot-directory=DIR	Use DIR as the boot directory. The default is /boot. GRUB will put its files in a subdirectory of this directory named grub.
--target=TARGET	Install GRUB for TARGET platform. The default is the platform grub-install is running on.
--grub-setup=FILE	Use FILE as grub-setup. The default is /usr/bin/grub-setup.
--grub-mkrelpath=FILE	Use FILE as grub-mkrelpath. The default is /usr/bin/grub-mkrelpath.
--grub-probe=FILE	Use FILE as grub-probe. The default is /usr/bin/grub-mkrelpath.
--allow-floppy	Make the device also bootable as a floppy. This option is the default for /dev/fdX devices. Some BIOSes will not boot images created with this option.
--recheck	Delete any existing device map and create a new one if necessary.
--force	Install even if problems are detected.
--force-file-id	Use identifier file even if UUID is available.
--disk-module=*MODULE*	Use MODULE for disk access. This allows you to manually specify either biosdisk or native disk access. This option is only available on the BIOS target platform.
--removable	Treat the target device as if it is removable. This option is only available on the EFI target platform.
--bootloader-id=*ID*	Use ID as the bootloader ID. This option is only available on the EFI target platform.
--efi-directory=*DIR*	Use DIR as the EFI System Partition root. This option is only available on the EFI target platform.
INSTALL_DEVICE	Install GRUB to the block device INSTALL_DEVICE.

Adapted from:
"GNU GRUB.svg" by Multiple Contributors, Wikimedia Commons is licensed under CC BY-SA 3.0

07-B.5: GRUB2 - Installation is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

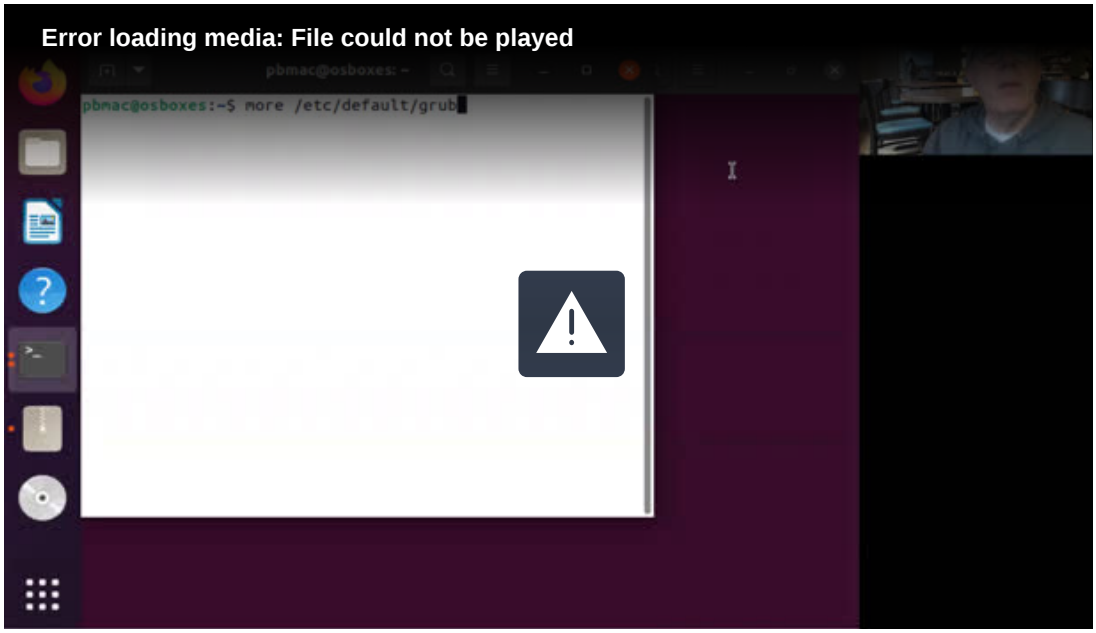
07-B.6: GRUB2 - Configuration

The grub.cfg File

The **grub.cfg** file is the GRUB configuration file. It is generated by the **grub2-mkconfig** program using a set of primary configuration files and the grub default file as a source for user configuration specifications. The **/boot/grub2/grub.cfg** file is first generated during Linux installation and regenerated when a new kernel is installed.

The **grub.cfg** file contains Bash-like code and a list of installed kernels in an array ordered by sequence of installation. For example, if you have four installed kernels, the most recent kernel will be at index 0, the previous kernel will be at index 1, and the oldest kernel will be index 3. If you have access to a grub.cfg file you should look at it to get a feel for what one looks like. The grub.cfg file is just too large to be included in this article.

The video discusses how to configure GRUB. It is important you understand how to use GRUB, and the ramifications of not understanding what you are doing.



GRUB defaults file

Configuration of the original GRUB was fairly simple and straightforward. I would just modify **/boot/grub/grub.conf** and be good to go. I could still modify GRUB2 by changing **/boot/grub2/grub.cfg**, but the new version is considerably more complex than the original GRUB. In addition, grub.cfg may be overwritten when a new kernel is installed, so any changes may disappear. However, the GNU.org GRUB Manual does discuss direct creation and modification of **/boot/grub2/grub.cfg.**

Changing the configuration for GRUB2 is fairly easy once you actually figure out how to do it. I only discovered this while researching GRUB2 for a previous article. The secret formula is in the /etc/default directory, with a file called, naturally enough, grub, which is then used in concert with a simple terminal command. The /etc/default directory contains configuration files for a few programs such as Google Chrome, useradd, and grub.

The **/etc/default/grub** file is very simple. The grub defaults file has a number of valid key/value pairs listed already. You can simply change the values of existing keys or add other keys that are not already in the file. Listing 1, below, shows an unmodified **/etc/default/grub** file.

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g'
   /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="rd.lvm.lv=fedora_fedora25vm/root
   rd.lvm.lv=fedora_fedora25vm/swap
```

```
   rd.lvm.lv=fedora_fedora25vm/usr rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

Section 5.1 of the GRUB Manual contains information about all of the possible keys that can be included in the grub file. I have never needed to do anything other than modifying the values of some of the keys that are already in the grub default file. Let's look at what each of these keys means as well as some that don't appear in the grub default file.

- **GRUB_TIMEOUT** The value of this key determines the length of time that the GRUB selection menu is displayed. GRUB offers the capability to keep multiple kernels installed simultaneously and choose between them at boot time using the GRUB menu. The default value for this key is five seconds, but I usually change that to 10 seconds to allow more time to view the choices and make a selection.
- **GRUB_DISTRIBUTOR** This key defines a sed expression that extracts the distribution release number from the /etc/system-release file. This information is used to generate the text names for each kernel release that appears in the GRUB menu, such as "Fedora." Due to variations in the structure of the data in the system-release file between distributions, this sed expression may be different on your system.
- **GRUB_DEFAULT** Determines which kernel is booted by default. That is, the "saved" kernel which is the most recent kernel. Other options here are a number which represents the index of the list of kernels in **grub.cfg**. Using an index such as 3, however, to load the fourth kernel in the list will always load the fourth kernel in the list even after a new kernel is installed. So using an index will load a different kernel after a new kernel is installed. The only way to ensure that a specific kernel release is booted is to set the value of **GRUB_DEFAULT** to the name of the desired kernel, like 4.8.13-300.fc25.x86_64.
- **GRUB_SAVEDEFAULT** Normally, this option is not specified in the grub defaults file. Normal operation when a different kernel is selected for boot; that kernel is booted only that one time. The default kernel is not changed. When set to "true" and used with **GRUB_DEFAULT=saved** this option saves a different kernel as the default. This happens when a different kernel is selected for boot.
- **GRUB_DISABLE_SUBMENU** Some people may wish to create a hierarchical menu structure of kernels for the GRUB menu screen. This key, along with some additional configuration of the kernel stanzas in **grub.cfg** allows creating such a hierarchy. For example, the one might have the main menu with "production" and "test" sub-menus where each sub-menu would contain the appropriate kernels. Setting this to "false" would enable the use of sub-menus.
- **GRUB_TERMINAL_OUTPUT** In some environments it may be desirable or necessary to redirect output to a different display console or terminal. The default is to send output to the default terminal, usually the "console" which equates to the standard display on an Intel class PC. Another useful option is to specify "serial" in a data center or lab environment in which serial terminals or Integrated Lights Out (ILO) terminal connections are in use.
- **GRUB_TERMINAL_INPUT** As with **GRUB_TERMINAL_OUTPUT**, it may be desirable or necessary to redirect input from a serial terminal or ILO device rather than the standard keyboard input.
- **GRUB_CMDLINE_LINUX** This key contains the command line arguments that will be passed to the kernel at boot time. Note that these arguments will be added to the kernel line of grub.cfg for all installed kernels. This means that all installed kernels will have the same arguments when booted. I usually remove the "rhgb" and "quiet" arguments so that I can see all of the very informative messages output by the kernel and systemd during the boot and startup.
- **GRUB_DISABLE_RECOVERY** When the value of this key is set to "false," a recovery entry is created in the GRUB menu for every installed kernel. When set to "true" no recovery entries are created. Regardless of this setting, the last kernel entry is always a "rescue" option. However, I encountered a problem with the rescue option, which I'll talk more about below.

The /etc/grub.d Directory

The main set of configuration files for grub.cfg is located in the /etc/grub.d directory. Each of the files in that directory contains GRUB code that is collected into the final grub.cfg file. The numbering scheme used in the names of these configuration files is designed to provide ordering so that the final grub.cfg file is assembled into the correct sequence. Each of these files has a comment to denote the beginning and end of the section, and those comments are also part of the final grub.cfg file so that it is possible to see from which file each section is generated. The delimiting comments look like this:

```
### BEGIN /etc/grub.d/10_linux ###

### END /etc/grub.d/10_linux ###
```

These files should not be modified unless you are a GRUB expert and understand what the changes will do. Even then you should always keep a backup copy of the original, working grub.cfg file. The specific files, 40_custom and 41_custom are intended to be used to generate user modifications to the GRUB configuration. You should still be aware of the consequences of any changes you make to these files and maintain a backup of the original grub.cfg file.

You can also add your own files to the /etc/grub.d directory. One reason for doing that might be to add a menu line for a non-Linux operating system. Just be sure to follow the naming convention to ensure that the additional menu item is added either immediately before or after the 10_linux entry

in the configuration file.

You can also add your own files to the /etc/grub.d directory. One reason for doing that might be to add a menu line for a non-Linux operating system. Just be sure to follow the naming convention to ensure that the additional menu item is added either immediately before or after the 10_linux entry in the configuration file.

How to Customize GRUB2 Boot Menu

GRUB 2 constructs the menu via a series of scripts, with each script building a portion of the menu. These scripts include **`/etc/grub.d/00_header`** and **`/etc/grub.d/05_header`**. The GRUB 2 menu configuration file (grub.cfg) includes section remarks showing which script is responsible for creating the section. The /etc/default/grub file contains most of the user-defined variables such as display time, menu resolution and default menu entry. Most custom menu users will want to allow these components to continue to operate normally.

A sample custom menu called **`40_custom`** is provided in the **`/etc/grub.d/`** folder. This file can be used or copied. The current **`40_custom`** file contains only lines which are not actually imported into **`grub.cfg`**. In general, users should honor the comments and leave these lines alone, adding custom entries below the existing lines. The 40_customer file looks something like:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
```

One common customization is the addition of a password to protect the GRUB2 menu. The password can be generated by using the **`grub2-mkpasswd-pbkdf2`** command. Enter this command, type in the desired password twice as requested and it generates the password.

```
pbmac@pbmac-server $ sudo grub-mkpasswd-pbkdf2
Enter password:
Reenter password:
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.E71841FEC7692B19F7819BEDD0ED9B28C690C973
53A2F74EEA5B3A8B12C00BBADEE5E50B2ADD85B63C1661156109D894A4705AF5E782A0471FF9.DC9A1D683BE14ACA4C0D
C349DE6EDD9FDAAC311BC754F9812932F518FD15EC62A3AD9A7557E76DBB7E1B5EF835EC9B4204778C1559C9942DB14F
```

This password can then be entered into the **`40_customer`** file. You need to enter a valid user id in the "set superuser=" line, then enter the "password_pbkdf2," followed by the userid then paste the generated password.

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
set superuse="pbmac"
password_pbkdf2 pbmac grub.pbkdf2.sha512.10000.E71841FEC7692B19F7819BEDD0ED9B28C690C97375A384D807
53A2F74EEA5B3A8B12C00BBADEE5E50B2ADD85B63C1661156109D894A4705AF5E782A0471FF9.DC9A1D683BE14ACA4C0D
C349DE6EDD9FDAAC311BC754F9812932F518FD15EC62A3AD9A7557E76DBB7E1B5EF835EC9B4204778C1559C9942DB14F
```

The grub-mkconfig Command

Depending on the distribution of Linux you are running the command might be **`grub-mkconfig`**, and on other distros it is **`grub2-mkconfig`**. After completing the desired configuration it is necessary to generate the **`/boot/grub2/grub.cfg`** file.

This is a simple command - with a single option -o, to output the results to the specified file.

This is accomplished with the following command.

```
grub-mkconfig -o /boot/grub2/grub.cfg
```

This command takes the configuration files located in **`/etc/grub.d`** in sequence to build the grub.cfg file, and uses the contents of the grub defaults file to modify the output to achieve the final desired configuration. The **`grub-mkconfig`** command attempts to locate all of the installed

kernels and creates an entry for each in the 10_Linux section of the **`grub.cfg`** file. It also creates a "rescue" entry to provide a method for recovering from significant problems that prevent Linux from booting.

It is strongly recommended that you do not edit the grub.cfg file manually because any direct modifications to the file will be overwritten the next time a new kernel is installed or **`grub-mkconfig`** is run manually.

Adapted from
"An introduction to GRUB2 configuration for your Linux machine" by David Both, opensource.com is licensed under CC BY-SA 4.0
"Grub2/Passwords" by Eric Heintzmann, CommunityHelpWiki is licensed under CC BY-SA 3.0
"Grub2/Setup" by sudodus, CommunityHelpWiki is licensed under CC BY-SA 3.0
"Grub2/CustomMenus" by damage3025, CommunityHelpWiki is licensed under CC BY-SA 3.0

07-B.6: GRUB2 - Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 08: How to Manage System Components

- 08-A.1: Localization Options - Configuration
- 08-A.2: Localization - date/timdatectl Command
- 08-A.3: Localization - localectl Command
- 08-B.4: GUI Configuration
- 08-B.5: GUI Desktop Environments
- 08-B.6: Remote Desktop Access
- 08-B.7: Remote Console - SSH/X Forwarding
- 08-C.8: Managing Linux Services
  - 08-C.8.1: Managing Linux Services - systemctl
  - 08-C.8.2: Managing Linux Services - hostnamectl
  - 08-C.8.3: Managing Linux Services - SysVinit
  - 08-C.8.4: Managing Linux Services - inittab & init.d
- 08-D.9: Process Troubleshooting
  - 08-D.9.1: Process Troubleshooting - ps command
  - 08-D.9.2: Process Troubleshooting - top command
  - 08-D.9.3: Process Troubleshooting - lsof Command
  - 08-D.9.4: Process Troubleshooting - Process Priorities
  - 08-D.9.5: Process Troubleshooting - Background / Foreground Processes
  - 08-D.9.6: Process Troubleshooting - nohup and kill Commands
- 08-E.10: CPU and Memory Troubleshooting
  - 08-E.10.1: CPU and Memory Troubleshooting - sar & Memory Issues
  - 08-E.10.2: CPU and Memory Troubleshooting - free / vmstat Commands
  - 08-E.10.3: CPU and Memory Troubleshooting - OOM & swap

Module 08: How to Manage System Components is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-A.1: Localization Options - Configuration

EXAM OBJECTIVES COVERED

1.6 Given a scenario, configure localization options.

Objectives of this Module

In this lesson, you will:

- Configure localization settings in Linux
- Recognize the purpose of the graphical user interface
- Understand remote desktop access and identify remote desktop packages
- Use remote console and SSH tunneling

What is Localization

When dealing with GNU/Linux systems in an international context or for a specific country or region, both localization (abbreviated to *l10n*) and internationalization (abbreviated to *i18n*) play an important part. Localization allows administrators and users to select the language of choice on the platform, time zone selection, character ordering and more. In Linux, localization is supported in various levels ranging from kernel support up to end user application support.

Localization is the process of creating or adapting a product to a specific locale, i.e., to the language, cultural context, conventions and market requirements of a specific target market. With a properly localized product, a user can interact with this product using his/her own language and cultural conventions. It also means that all user-visible messages and all user documentation (printed and electronic) use the language and cultural conventions of the user. Finally, the properly localized product meets all regulatory and other requirements of the user's country/region.

Internationalization is a way of designing and producing products that can be easily adapted to different locales. This requires extracting all language, country/region and culturally dependent elements from a product. In other words, internationalization is the process of developing an application whose feature design and code design do not make assumptions based on a single locale, and whose source code simplifies the creation of different local editions of a program.

The /usr/share/zoneinfo Database

The /usr/share/zoneinfo files are made up of a set of directories, subdirectories and files that contain information about all the various time zones that Linux can be set to. These files come from the tz database, which is a collaborative compilation of information about the world's time zones, primarily intended for use with computer programs and operating systems. The tz database is also known as tzdata, the zoneinfo database, or IANA time zone database, and occasionally as the Olson database, referring to the founding contributor, Arthur David Olson. The database attempts to record historical time zones and all civil changes since 1970, the Unix time epoch. It includes transitions such as daylight saving time, and also records leap seconds.

Looking at the directory that contains the US time zones we see that there is a file for every time time zone that is currently in effect in the US.

```
pbmac@pbmac-server $ ls /usr/share/zoneinfo/US
Alaska    Arizona  Eastern       Hawaii          Michigan  Pacific      Samoa
Aleutian  Central  East-Indiana  Indiana-Starke  Mountain  Pacific-New
```

Two Time Zone Files

There are two files that contain time zone information on most Linux systems. There is the /etc/localtime, a symbolic link to the /usr/share/zoneinfo file, which is the actual zone information for each time zone as mentioned above. There is also the /etc/timezone file, which is a text file containing the location within the /usr/share/zoneinfo directory of the current time zone file. These two files should always be in sync, but some utilities only change one of the files, so if you are changing or setting the time zone, make sure that these files are synced and point to the same information.

Adapted from:

"tz database" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"Internationalization and localization" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"Localization" by Mutiple contributors, Wikipedia is licensed under CC BY-SA 3.0

08-A.1: Localization Options - Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-A.2: Localization - date/timdatectl Command

Types of System Clocks

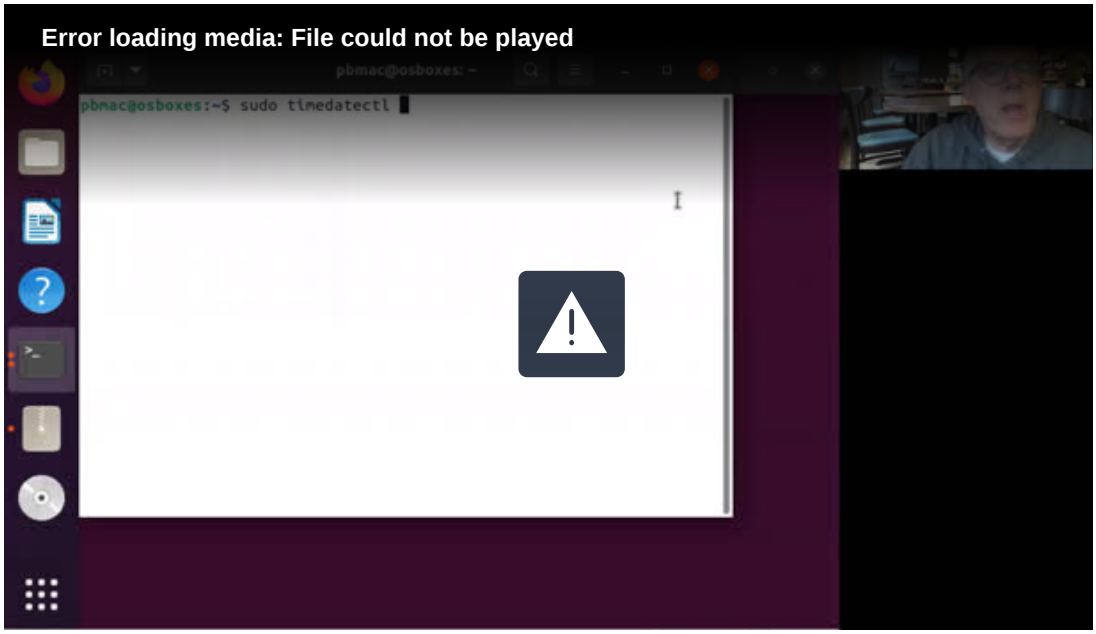
Modern operating systems distinguish between the following two types of clocks:

- A real-time clock (RTC), commonly referred to as a hardware clock, (typically an integrated circuit on the system board) that is completely independent of the current state of the operating system and runs even when the computer is shut down.
- A system clock, also known as a software clock, that is maintained by the kernel and whose initial value is based on the real-time clock. Once the system is booted and the system clock is initialized, the system clock is completely independent of the real-time clock.

The system time is always kept in Coordinated Universal Time (UTC) and converted in applications to local time as needed. Local time is the actual time in your current time zone, taking into account daylight savings time (DST). The real-time clock can use either UTC or local time. UTC is recommended.

Linux offers three command line tools that can be used to configure and display information about the system date and time: the **`date`** command; the **`timedatectl`** utility, which is part of **`systemd`**; and the **`hwclock`** utility for accessing the hardware clock.

The video delves into localization configuration. It covers topics that an administrator need to know to manage localizations.



The `date` Command

The **`date`** command is used to display the system date and time. It is also used to set date and time of the system. By default the date command displays the date in the time zone for which the system is configured. You must be the super-user (root) to change the date and time.

Syntax:

```
date [ OPTION ] [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

Command Options:

Option	Option Meaning
-d, --date=,STRING/	display time described by STRING, not 'now'
-s, --set=,STRING/	set time described by STRING
-u, --utc, --universal	print or set Coordinated Universal Time (UTC)

Usually the **`date`** command is used to simply display the current date and time, as shown in the example. However, there are other options that can give other output as well.

```
pbmac@pbmac-server $ date
Tue Aug 11 17:29:40 PDT 2020
pbmac@pbmac-server $ date -u
Wed Aug 12 00:29:43 UTC 2020
pbmac@pbmac-server $ date --date="1 month ago"
Sat Jul 11 17:30:19 PDT 2020
pbmac@pbmac-server $ date --date="last week"
Tue Aug  4 17:30:43 PDT 2020
pbmac@pbmac-server $ date --date="10 days ago"
Sat Aug  1 17:31:02 PDT 2020
```

In addition to the options outlined above, there is a syntax that can be used to display the date and or time in a user specified format.

```
%D: Display date as mm/dd/yy.
%d: Display the day of the month (01 to 31).
%a: Displays the abbreviated name for weekdays (Sun to Sat).
%A: Displays full weekdays (Sunday to Saturday).
%h: Displays abbreviated month name (Jan to Dec).
%b: Displays abbreviated month name (Jan to Dec).
%B: Displays full month name(January to December).
%m: Displays the month of year (01 to 12).
%y: Displays last two digits of the year(00 to 99).
%Y: Display four-digit year.
%T: Display the time in 24 hour format as HH:MM:SS.
%H: Display the hour.
%M: Display the minute.
%S: Display the seconds.
```

Using these syntactical components it is possible to display the date and time and even day of the week. Notice that if you string more than one you need to enclose them in quotation marks.

```
pbmac@pbmac-server $ date +%D
08/11/20
pbmac@pbmac-server $ date "+%D %T"
08/11/20 17:44:02
pbmac@pbmac-server $ date "+%Y-%m-%d"
2020-08-11
pbmac@pbmac-server $ date "+%A %B %d %T %y"
Tuesday August 11 17:44:24 20
```

The `timedatectl` Command

The **`timedatectl`** utility is distributed as part of the **`systemd`** system and service manager and allows you to review and change the configuration of the system clock. You can use this tool to change the current date and time, set the time zone, or enable automatic synchronization of the system clock with a remote server.

Syntax:

```
timedatectl [ OPTIONS ] {COMMAND}
```

Command Options:

https://eng.libretexts.org/@go/page/32994

Option	Option Meaning
--no-ask-password	Do not query the user for authentication for privileged operations.
--adjust-system-clock	If set-local-rtc is invoked and this option is passed, the system clock is synchronized from the RTC again, taking the new setting into account. Otherwise, the RTC is synchronized from the system clock.
-M, --machine=	Execute operation on a local container. Specify a container name to connect to.
-H, --host=	Execute the operation remotely.

Command Line Commands

There is also a set of sub-commands that can be used with **`timedatectl`** to perform a variety of actions.

Sub-command	Explanation
status	Show current settings of the system clock and RTC, including whether network time synchronization is active. If no command is specified, this is the implied default.
set-time [TIME]	Set the system clock to the specified time. This will also update the RTC time accordingly. The time may be specified in the format "2012-10-30 18:17:16."
set-timezone [TIMEZONE]	Set the system time zone to the specified value. Available time zones can be listed with list-timezones. If the RTC is configured to be in the local time, this will also update the RTC time. This call will alter the /etc/localtime symlink. See localtime for more information.
list-timezones	List available time zones, one per line. Entries from the list can be set as the system time zone with set-timezone.

So, to use the above mentioned commands, we simply combine them with the **`timedatectl`** command itself and if appropriate we can add the options as well.

```
pbmac@pbmac-server $ timedatectl
                      Local time: Wed 2020-08-12 11:49:59 PDT
                  Universal time: Wed 2020-08-12 18:49:59 UTC
                        RTC time: Wed 2020-08-12 18:49:59
                       Time zone: America/Los_Angeles (PDT, -0700)
       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
pbmac@pbmac-server $ timedatectl set-time 23:26:00
pbmac@pbmac-server $ timedatectl set-time "2013-06-02 23:26:00"
pbmac@pbmac-server $ timedatectl set-timezone America/Los_Angeles
```

The **`hwclock`** Command

`hwclock` is a utility for accessing the hardware clock, also referred to as the Real Time Clock (RTC). The hardware clock is independent of the operating system you use and works even when the machine is shut down. This utility is used for displaying the time from the hardware clock. **`hwclock`** also contains facilities for compensating for systematic drift in the hardware clock.

The hardware clock stores the values of: year, month, day, hour, minute, and second. It is not able to store the time standard, local time or Coordinated Universal Time (UTC), nor set the Daylight Saving Time (DST).

The **`hwclock`** utility saves its settings in the **`/etc/adjtime`** file, which is created with the first change.

Syntax:

```
hwclock [FUNCTION] [OPTION...]
```

Command Options:

Option	Option Meaning
a, --adjust	Add or subtract time from the Hardware Clock to account for systematic drift since the last time the clock was set or adjusted.
-s, --hctosys	Set the System Clock from the Hardware Clock.
-set	Set the Hardware Clock to the time given by the --date option, and update the timestamps in /etc/adjtime.
w, --systohc	Set the Hardware Clock from the System Clock, and update the timestamps in */etc/adjtime*.

Adapted from:
"date command in Linux with examples" by saloni1297, Geeks for Geeks is licensed under CC BY-SA 3.0
"Chapter 3. Configuring the Date and Time" by Multiple Contributors, Fedora Docs is licensed under CC BY-SA 3.0
"3.3. Using the hwclock Command" by Multiple Contributors, Fedora Docs is licensed under CC BY-SA 3.0

08-A.2: Localization - date/timdatectl Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-A.3: Localization - localectl Command

The `localectl` Command

In computing, a locale is a set of parameters that defines the user's language, region, and any special variant preferences that the user wants to see in their user interface. Usually a locale identifier consists of at least a language code and a country/region code.

These settings can be made by modifying the /etc/locale.conf (although some distributions use the /etc/default/locale file) configuration file or by using the **`localectl`** utility. Also, you can use the graphical user interface to perform the task.

System-wide locale settings are stored in the /etc/locale.conf (or /etc/default/locale) file, which is read at early boot by the systemd daemon. The locale settings configured in /etc/locale.conf are inherited by every service or user, unless individual programs or individual users override them.

Syntax:

```
localectl [ OPTIONS ] {COMMAND}
```

As with the **`timedatectl`** command, the **`localectl`** command has typical Linux type options as well as sub-commands that specify what the command is supposed to perform.

Sub-Commands	Explanation
status	Show current settings of the system locale and keyboard mapping. If no command is specified, this is the implied default.
set-locale LOCALE, set-locale VARIABLE=LOCALE...	Set the system locale.
list-locales	List available locales useful for configuration with set-locale.
list-keymaps	List available keyboard mappings for the console, useful for configuration with set-keymap.

The following options can be combined with the above mentioned sub-commands -examples can be found below.

Command Options:

Option	Option Meaning
-H, --host=	Execute the operation remotely.
--no-ask-password	Do not query the user for authentication for privileged operations.
--no-convert	If set-keymap or set-x11-keymap is invoked and this option is passed, then the keymap will not be converted from the console to X11, or X11 to console, respectively.

A couple of examples to show the usage:

```
pbmac@pbmac-server $ localectl status
   System Locale: LANG=en_US.UTF-8
```

```
       VC Keymap: us
      X11 Layout: n/a

to select a specific English locale, but if you are not sure if it is available
on the system, you can check that by listing all English locales
pbmac@pbmac-server $ localectl list-locales | grep en_
en_AG
en_AG.utf8
en_AU
en_AU.iso88591
en_AU.utf8
en_BW
en_BW.iso88591
en_BW.utf8
...

to set British English as your default locale, first find the name of this
locale by using list-locales. Then, as root, type the command in the following form:
pbmac@pbmac-server $ localectl set-locale LANG=en_GB.utf8
```

Character Sets, Coded Character Sets, and Encoding

Historically, the terms "character encoding", "character map", "character set" and "code page" were synonymous in computer science, as the same standard would specify a repertoire of characters and how they were to be encoded into a stream of code units – usually with a single character per code unit. But now the terms have related but distinct meanings, due to efforts by standards bodies to use precise terminology when writing about and unifying many different encoding systems. Regardless, the terms are still used interchangeably, with character set being nearly ubiquitous.A character set or repertoire comprises the set of characters one might use for a particular purpose – be it those required to support Western European languages in computers, or those a Chinese child will learn at school in the third grade (nothing to do with computers).

A coded character set is a set of characters for which a unique number has been assigned to each character. Units of a coded character set are known as code points. A code point value represents the position of a character in the coded character set. For example, the code point for the letter á in the Unicode coded character set is 225 in decimal, or 0xE1 in hexadecimal notation. (Note that hexadecimal notation is commonly used for referring to code points, and will be used here.) A Unicode code point can have a value between 0x0000 and 0x10FFFF.

Coded character sets are sometimes called code pages.

A "code page" usually means a byte-oriented encoding, but with regard to some suite of encodings (covering different scripts), where many characters share the same codes in most or all those code pages. Well-known code page suites are "Windows" (based on Windows-1252) and "IBM"/"DOS" (based on code page 437), see Windows code page for details. Most, but not all, encodings referred to as code pages are single-byte encodings (but see octet on byte size.)

IBM's Character Data Representation Architecture (CDRA) designates entities with coded character set identifiers (CCSIDs), each of which is variously called a "charset", "character set", "code page", or "CHARMAP".

The term "code page" does not occur in Unix or Linux where "charmap" is preferred, usually in the larger context of locales.

In contrast to a "coded character set", a "character encoding" is a map from abstract characters to code words. A "character set" in HTTP (and MIME) parlance is the same as a character encoding (but not the same as CCS).

"Legacy encoding" is a term sometimes used to characterize old character encodings, but with an ambiguity of sense. Most of its use is in the context of Unicodification, where it refers to encodings that fail to cover all Unicode code points, or, more generally,

using a somewhat different character repertoire: several code points representing one Unicode character, or versa (see e.g. code page 437). Some sources refer to an encoding as legacy only because it preceded Unicode. All Windows code pages are usually referred to as legacy, both because they antedate Unicode and because they are unable to represent all 221 possible Unicode code points.

Adapted from:
"Chapter 2. System Locale and Keyboard Configuration" by Multiple Contributors, Fedora Docs is licensed under CC BY-SA 3.0
"Character encoding" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

08-A.3: Localization - localectl Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-B.4: GUI Configuration

EXAM OBJECTIVES COVERED

2.8 Compare and contrast Linux graphical user interfaces.

GUI

The graphical user interface (GUI) is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLI), which require commands to be typed on a computer keyboard.

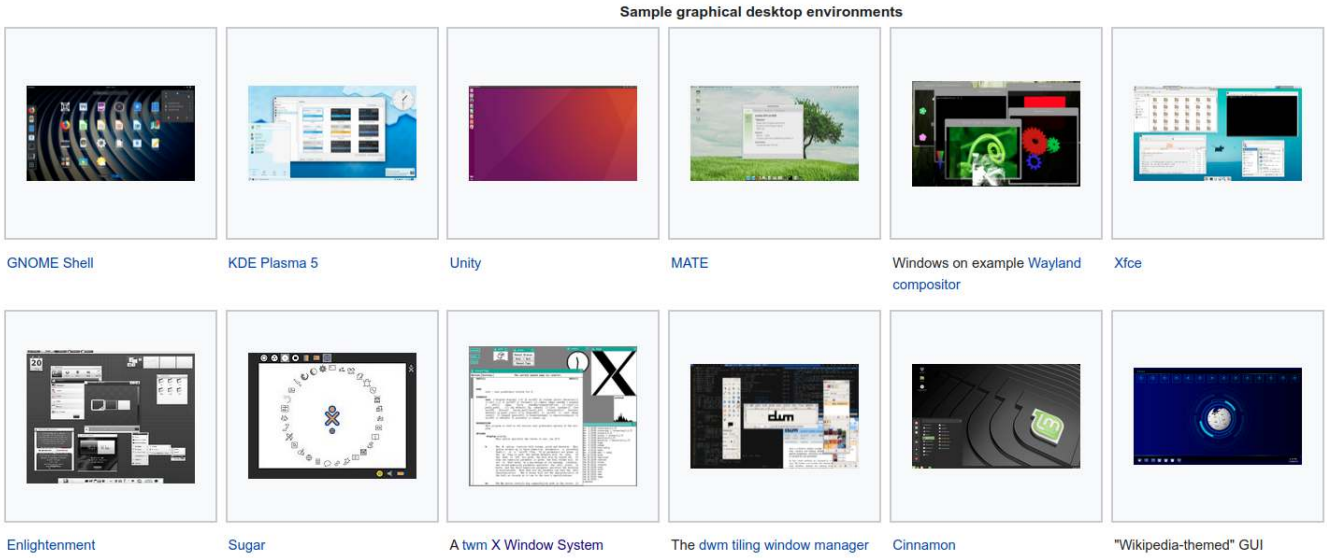


Figure **8.4.1**: Linux GUI. ("GUI" by Patrick McClanahan is licensed under CC BY-SA 4.0)

The actions in a GUI are usually performed through direct manipulation of the graphical elements. Beyond computers, GUIs are used in many handheld mobile devices such as MP3 players, portable media players, gaming devices, smartphones and smaller household, office and industrial controls. The term GUI tends not to be applied to other lower-display resolution types of interfaces, such as video games, or not to include flat screens like volumetric displays. The term is restricted to the scope of two-dimensional display screens able to describe generic information, in the tradition of the computer science research at the Xerox Palo Alto Research Center.

Display Servers

A display server or window server is a program whose primary task is to coordinate the input and output of its clients to and from the rest of the operating system, the hardware, and each other. The display server communicates with its clients over the display server protocol, a communications protocol, which can be network-transparent or simply network-capable.

The display server is a key component in any graphical user interface, specifically the windowing system.

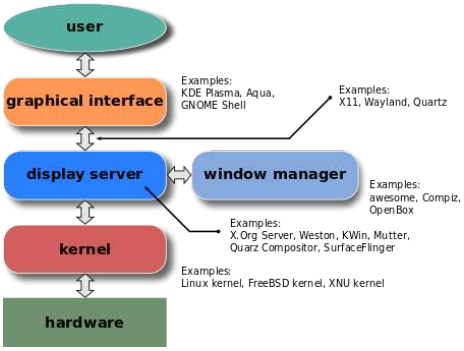


Figure **8.4.1**: Layers of a GUI based on a windowing system.
("Schema_of_the_layers_of_the_graphical_user_interface.svg" by ScotXW, Wikimedia Commons is licensed under CC BY-SA 3.0)

X Window System

The **X Window System** (**X11**, or simply **X**) is a windowing system for bitmap displays, common on Unix-like operating systems.

X provides the basic framework for a GUI environment: drawing and moving windows on the display device and interacting with a mouse and keyboard. X does not mandate the user interface – this is handled by individual programs. As such, the visual styling of X-based environments varies greatly; different programs may present radically different interfaces.

X originated at the Project Athena at Massachusetts Institute of Technology (MIT) in 1984. The X protocol has been at version 11 (hence "X11") since September 1987. The X.Org Foundation leads the X project, with the current reference implementation, X.Org Server, available as free and open-source software under the MIT License and similar permissive licenses.

X primarily defines protocol and graphics primitives – it deliberately contains no specification for application user-interface design, such as button, menu, or window title-bar styles. Instead, application software – such as window managers, GUI widget toolkits and desktop environments, or application-specific graphical user interfaces – define and provide such details. As a result, there is no typical X interface and several different desktop environments have become popular among users.

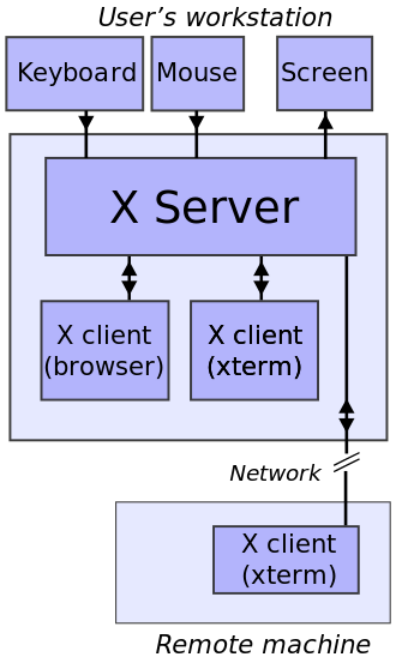


Figure **8.4.1**: A graphic depicting how the X Window System operates.
("X_client_server_example.svg" by Efitu, Wikimedia Commons is licensed under CC BY-SA 3.0)

A window manager controls the placement and appearance of application windows. This may result in desktop interfaces reminiscent of those of Microsoft Windows or of the Apple Macintosh (examples include GNOME 2, KDE, Xfce) or have radically different controls (such as a tiling window manager, like wmii or Ratpoison). Some interfaces such as Sugar or Chrome OS eschew the desktop metaphor altogether, simplifying their interfaces for specialized applications. Window managers range in sophistication and complexity from the bare-bones (*e.g.*, twm, the basic window manager supplied with X, or evilwm, an extremely light window-manager) to the more comprehensive desktop environments such as Enlightenment and even to application-specific window-managers for vertical markets such as point-of-sale.

Many users use X with a desktop environment, which, aside from the window manager, includes various applications using a consistent user-interface. Popular desktop environments include GNOME, KDE Software Compilation and Xfce. The UNIX 98 standard environment is the (CDE). The freedesktop.org initiative addresses interoperability between desktops and the components needed for a competitive X desktop.

The X.Org Project

The X.Org project provides an open source implementation of the X Window System. The development work is being done in conjunction with the freedesktop.org community. The X.Org Foundation is the educational non-profit corporation whose Board serves this effort, and whose members lead this work.

Xorg (commonly referred as simply X) is the most popular display server among Linux users. Its ubiquity has led to making it an ever-present requisite for GUI applications, resulting in massive adoption from most distributions. See the Xorg Wikipedia article or visit the Xorg website for more details.

Wayland

Wayland is a communication protocol that specifies the communication between a display server and its clients, as well as a C library implementation of that protocol. A display server using the Wayland protocol is called a Wayland compositor, because it additionally performs the task of a compositing window manager.

Wayland is developed by a group of volunteers as a free and open-source community-driven project with the aim of replacing the X Window System with a modern, secure, simpler windowing system in Linux and other Unix-like operating systems. The project's

source code is published under the terms of the MIT License, a permissive free software license.

As part of its efforts, the Wayland project also develops a reference implementation of a Wayland compositor called Weston.

Within the Linux community there are those who support X.Org, and those who support the use of Wayland. If you search for "X.Org vs Wayland" you can find numerous articles that discuss the comparison from different viewpoints.

Adapted from:
"Graphical user interface" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Display server" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"X Window System" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Xorg" by Multiple Contributors, Wikipedia is in the Public Domain, CC0

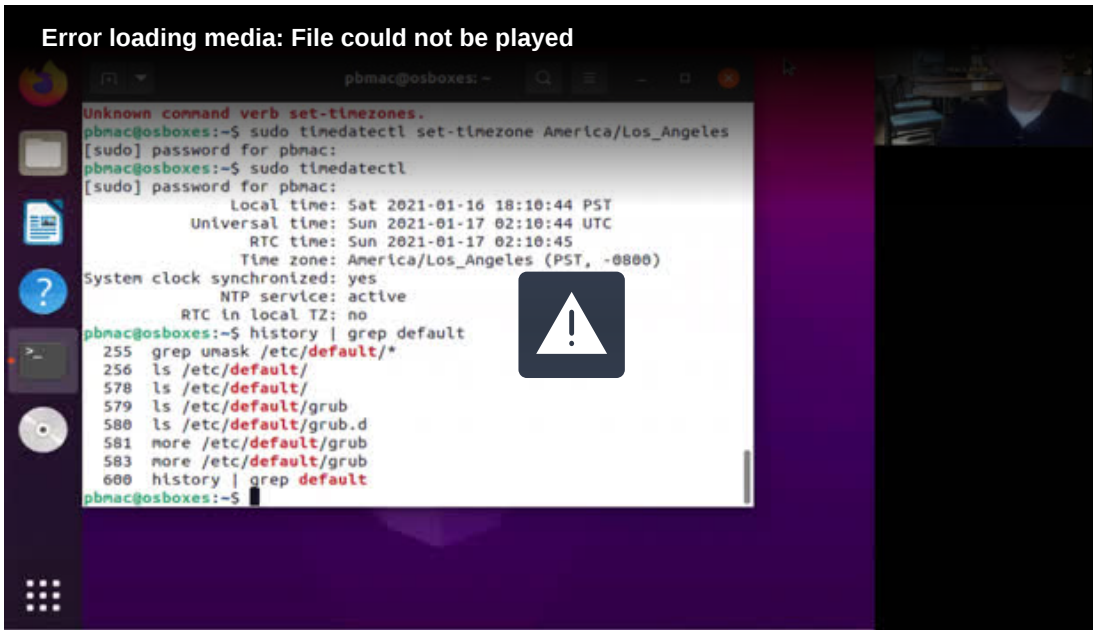
08-B.4: GUI Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-B.5: GUI Desktop Environments

Desktop Environment

A desktop environment bundles together a variety of components to provide common graphical user interface elements such as icons, toolbars, wallpapers, and desktop widgets. Additionally, most desktop environments include a set of integrated applications and utilities. Most importantly, desktop environments provide their own window manager, which can however usually be replaced with another compatible one.

The video discusses the capabilites that Linux provides to all certain setting to be changed to provide greater accessibility for users who need things such as screen readers, larger text, sound keys etc.



The user is free to configure their GUI environment in any number of ways. Desktop environments simply provide a complete and convenient means of accomplishing this task. Note that users are free to mix-and-match applications from multiple desktop environments. For example, a KDE user may install and run GNOME applications such as the Epiphany web browser, should he/she prefer it over KDE's Konqueror web browser. One drawback of this approach is that many applications provided by desktop environment projects rely heavily upon their DE's respective underlying libraries. As a result, installing applications from a range of desktop environments will require installation of a larger number of dependencies. Users seeking to conserve disk space often avoid such mixed environments, or choose alternatives which depend on only a few external libraries.

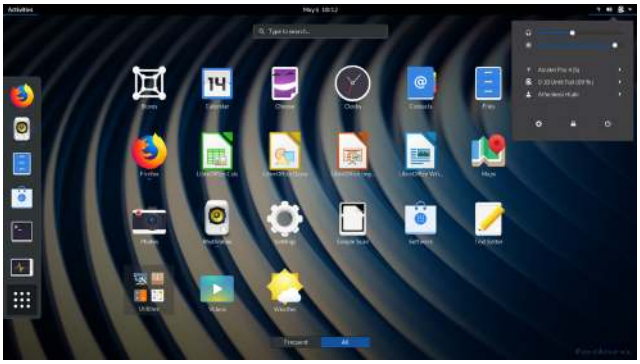


Figure **8.5.1**: Gnome 3.21. ("NOME_3.32.1.png" by Athenkosi Hlubi, Wikimedia Commons is licensed under CC BY-SA 3.0)

Furthermore, KDE-provided applications tend to integrate better with their native environments. Superficially, mixing environments with different widget toolkits will result in visual discrepancies (that is, interfaces will use different icons and widget styles). In terms of usability, mixed environments may not behave similarly (e.g. single-clicking versus double-clicking icons; drag-and-drop functionality) potentially causing confusion or unexpected behavior.

Prior to installing a desktop environment, a functional X server installation is required. See Xorg for detailed information. Some desktop environments may also support Wayland as an alternative to X, but most of these are still experimental.

The Best Desktop Environments:

Desktop Environment	Details
KDE - known as Plasma	KDE is one of the most popular desktop environments out there. You may also refer it as the "**Plasma**" desktop. Even though it's not my primary choice, it is highly customizable and extremely lightweight. It looks like KDE has managed to beat XFCE as one of the lightest desktop environments. Not just limited to Jason's test, you will find numerous benchmark comparisons and tests that point to the same result. KDE also makes it easier to connect your phone with your Linux system using KDE Connect. You will also find Plasma's browser integration that connects your phone directly to your browser for establishing quick communication. Overall, it looks like KDE is an incredibly lightweight desktop environment while being one of the most flexible as well. Some Linux distros using KDE as the default are openSUSE, Kubuntu and KDE Neon. You may also refer to one of our tutorials to install KDE on Ubuntu, if that's what you want.
MATE	MATE Desktop Environment is based on GNOME 2. MATE was initially developed for the users who were disappointed with the latest iteration of GNOME shell — GNOME 3. Even though it's based on the good-old GNOME 2, the MATE team has improved the desktop environment on a lot of grounds. To get some idea, you might want to check out what Ubuntu MATE 20.04 has to offer. If we take the example of Ubuntu MATE 20.04, MATE desktop is suitable for almost everyone. And, of course, especially for the ones who loved GNOME 2 but hate the new GNOME. In addition to the user experience, it is also worth noting that it is a lightweight desktop environment as well. MATE comes with a collection of basic applications and includes a number of built-in useful tools.
GNOME	GNOME is arguably the most popular desktop environment out there. Many of the popular Linux distros use GNOME as their default desktop environment and it has some popular forks, such as Cinnamon. GNOME is designed to be easy to use and customizable. The user interface aims to provide a unique experience (kind of tailored for both mobile and desktops). Unfortunately, GNOME isn't a lightweight desktop environment. So, it's not a great choice to go with if you are looking to install a Linux distribution on older computers or systems with less than 4 Gigs of RAM. It's good to see that GNOME is also focusing on the performance side of things with their recent GNOME 3.36 release. So, if you want a good user experience with something that looks different from the likes of a traditional Windows layout, GNOME should be the perfect pick.
Cinnamon	Cinnamon, a fork of GNOME 3, was initially developed to be and is the default desktop environment for Linux Mint. It is known for its similarities with the Windows user interface which helps new Linux users get comfortable using easily. Cinnamon tries to present itself as a modern desktop environment while offering a traditional user interface. And, being somewhat light on resources makes Cinnamon a balanced choice for many Linux users.
Budgie	Budgie desktop has been developed by the Solus project. In case you didn't know, Solus is an independently developed Linux distro where they utilize GNOME technologies such as GTK to develop the Budgie desktop environment. It's a mixed bag of modern UI and a traditional user interface. It's not exactly resource heavy but not a complete lightweight desktop environment either. Ubuntu Budgie — an official Ubuntu flavor - features the Budgie desktop as the default. Not a lot of major distributions support Budgie desktop

Desktop Environment	Details
	environment out of the box. So, you'd be better off using Solus or Ubuntu Budgie for the most part.

Adapted from:
"Desktop environment" by Multiple Contributors, Wikipedia s licensed under CC BY-SA 3.0
"The Best Desktop Environments For Linux (We Tested Them So That You Don't Have To)" by Ankush Das, It's FOSS is licensed under CC BY-SA 4.0

08-B.5: GUI Desktop Environments is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-B.6: Remote Desktop Access

Remote Desktop

Remote desktop refers to a software or operating system feature that allows a personal computer's desktop environment to be run remotely on one system (usually a PC, but the concept applies equally to a server), while being displayed on a separate client device. Remote desktop applications have varying features. Some allow attaching to an existing user's session (i.e., a running desktop) and "remote controlling," either displaying the remote control session or blanking the screen. Taking over a desktop remotely is a form of remote administration.

Remote access can also be explained as remote control of a computer by using another device connected via the internet or another network. This is widely used by many computer manufacturers and large businesses' help desks for technical troubleshooting of their customers' problems.

Remote desktop software captures the mouse and keyboard inputs from the local computer (client) and sends them to the remote computer (server). The remote computer in turn sends the display commands to the local computer. When applications with many graphics including video or 3D models need to be controlled remotely, a remote workstation software that sends the pixels rather than the display commands must be used to provide a smooth, like-local experience. HP Remote Graphics Software is one such remote workstation solution.

Remote Desktop Packages

There are numerous remote desktop software packages; some are proprietary and some are open source.

Remote desktop sharing is accomplished through a common client/server model. The client, or VNC viewer, is installed on a local computer and then connects via a network to a server component, which is installed on the remote computer. In a typical VNC session, all keystrokes and mouse clicks are registered as if the client were actually performing tasks on the end-user machine.

Remote Desktop Software	Comments
Teamviewer	TeamViewer is a proprietary software application for remote control, desktop sharing, online meetings, web conferencing and file transfer between computers. TeamViewer is available for Microsoft Windows, macOS, Linux, Chrome OS, iOS, Android, Windows RT, Windows Phone 8 and BlackBerry operating systems. It is also possible to access a machine running TeamViewer with a web browser. While the main focus of the application is remote control of computers, collaboration and presentation features are included.
Remmina	Remmina is a remote desktop client for POSIX-based computer operating systems. It supports the Remote Desktop Protocol (RDP), VNC, NX, XDMCP, SPICE and SSH protocols. Remmina is in the package repositories for Debian versions 6 (Squeeze) and later and for Ubuntu versions since 10.04 (Lucid Lynx). As of 11.04 (Natty Narwhal), it replaced tsclient as Ubuntu's default remote desktop client. The FreeBSD ports/package collection also contains it as a separate port and additional protocol-specific plugin ports.
Vinagre	Vinagre is a VNC, SSH, RDP and SPICE client for the GNOME desktop environment. It was included in GNOME 2.22. It has several features, like the ability to connect to multiple servers simultaneously and to switch between them using tabs, VNC servers browsing and bookmarking. In version 2.29, Vinagre added controlling frame compression, better scaling and color depth. Version 2.30 added improved SSH tunneling and better support for copy/paste features between client and server. Vinagre version 3.0 will operate with GNOME 3.0. As of 2012, features such as frame rate, file transfer and audio support have yet to become available.
TigerVNC	TigerVNC is VNC server and client software, started as a fork of TightVNC in 2009. Red Hat, Cendio AB, and TurboVNC maintainers started this fork because RealVNC had focused on their enterprise non-open VNC and no TightVNC update had appeared since 2006. The past few years however, Cendio AB, who use it for their product ThinLinc, is the main contributor to the project. TigerVNC is fully open-source, with development and discussion done via publicly accessible mailing lists and repositories. Compared to TightVNC, TigerVNC adds encryption for all supported operating systems (not just Linux), but it removes scaling the remote display into the client window, file transfer, and changing options while connected. TigerVNC focuses on performance and on remote display functionality. TigerVNC became the default VNC implementation in Fedora shortly after its creation.

Adapted from:
"TeamViewer" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"Remmina" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Vinagre" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"TigerVNC" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

08-B.6: Remote Desktop Access is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-B.7: Remote Console - SSH/X Forwarding

Console Redirection

Most Linux distributions can be configured for connections via the serial console port, which enables administration of a machine even if it has no keyboard, mouse, monitor, or network attached to it. A basic environment for this scenario is two machines connected using a serial cable (9-pin connector cable). The administering machine can be any Unix/Linux or Windows machine with a terminal emulator program (PuTTY or Minicom, for example).

In situations where a systems administrator may not have easy access to the Linux system, it can be configured to allow for remote administrative access, including booting the system, even providing access to the BIOS/UEFI. This configuration is often employed in a stat center setting, where there may be numerous servers without a physical keyboard/monitor/mouse.

SSH Tunneling

Port forwarding via SSH (SSH tunneling) creates a secure connection between a local computer and a remote machine through which services can be relayed. Because the connection is encrypted, SSH tunneling is useful for transmitting information that uses an unencrypted protocol, such as IMAP, VNC, or IRC.

Types of Port Forwarding

SSH's port forwarding feature can smuggle various types of Internet traffic into or out of a network. This can be used to avoid network monitoring or sniffers, or bypass badly configured routers on the Internet. Note: You might also need to change the settings in other programs (like your web browser) in order to circumvent these filters.

There are three types of port forwarding with SSH:

- Local port forwarding: connections from the SSH client are forwarded via the SSH server, then to a destination server.
- Remote port forwarding: connections from the SSH server are forwarded via the SSH client, then to a destination server.
- Dynamic port forwarding: connections from various programs are forwarded via the SSH client, then via the SSH server, and finally to several destination

Local port forwarding is the most common type. For example, local port forwarding lets you bypass a company firewall that blocks Wikipedia. Remote port forwarding is less common. For example, remote port forwarding lets you connect from your SSH server to a computer on your company's intranet.

Dynamic port forwarding is rarely used. For example, dynamic port forwarding lets you bypass a company firewall that blocks web access altogether. Although this is very powerful, it takes a lot of work to set up, and it's usually easier to use local port forwarding for the specific sites you want to access.

Port-forwarding is a widely supported technique and a feature found in all major SSH clients and servers, although not all clients do it the same way. For help on using a specific client, consult the client's documentation. For example, the PuTTY manual has a section on port forwarding in PuTTY

To use port forwarding, you need to make sure port forwarding is enabled in your server. You also need to tell your client the source and destination port numbers to use. If you're using local or remote forwarding, you need to tell your client the destination server. If you're using dynamic port forwarding, you need to configure your programs to use a SOCKS proxy server. Again, exactly how to do this depends on which SSH client you use, so you may need to consult your documentation.

X Forwarding

The X Window System (also known as X11, or just X) is a software package and network protocol that lets you interact locally, using your personal computer's display, mouse, and keyboard, with the graphical user interface (GUI) of an application running on a remote networked computer.

X forwarding can be used in an SSH session on one computer to securely run graphical applications (X clients) located on remote systems.

Accessibility Options

It is possible to make the Linux operating system usable by as many people as possible across ages, languages and physical abilities. This includes providing an accessible platform with high quality assistive tools, and ensuring that other applications work well with these. The assistive tools on Linux are often part of the default installation.

Options Common to Linux

- Screen reader - Orca is common.
- High contrast themes, complete with custom icons and mouse cursor.
- Keyboard modifiers including on screen keyboards.
- It is possible to control applications using voice commands. Julius speech recognition engine can be used to recognize voice commands and execute pre-set commands.
- For users that cannot click with a hardware button, there is a utility named Mousetweaks that enables them to perform the various clicks by software.

For systems that may not include accessibility tools as part of the default install, most of the tools are open source and freely available for download and install.

Adapted from:
"SSH/OpenSSH/PortForwarding" by Pasi Lallinaho, CommunityHelpWiki is licensed under CC BY-SA 4.0
"Accessibility" by Christopher M. Penalver, **CommunityHelpWiki** is licensed under CC BY-SA 4.0

08-B.7: Remote Console - SSH/X Forwarding is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-C.8: Managing Linux Services

EXAM OBJECTIVES COVERED

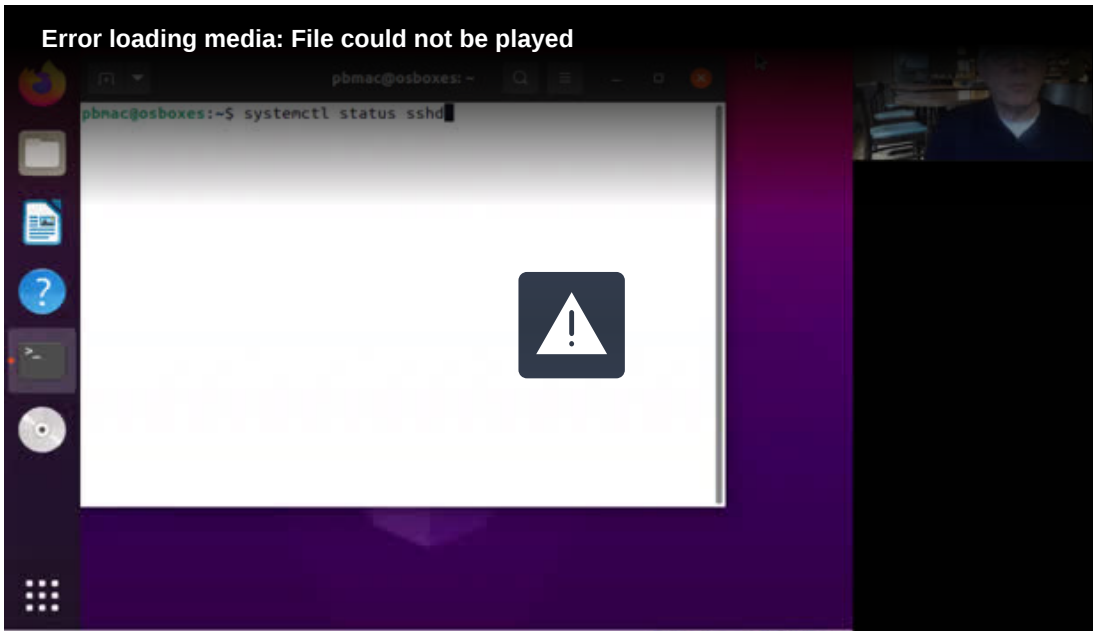
2.4 Given a scenario, manage services.

4.1 Given a scenario, analyze system properties and remediate accordingly.

Daemons and Services

Services are essential processes that usually run in the background, rather than being under the direct control of an interactive user, waiting for requests from other software programs, or to carry out essential tasks at the appropriate time. Many services are implemented as daemons.

The video talks about the Linux systemctl command and how to control system services.



In Linux, a daemon is a program that runs as a background process. Traditionally, the process names of a daemon end with the letter d, for clarification that the process is in fact a daemon, and for differentiation between a daemon and a normal computer program. For example, **`syslogd`** is a daemon that implements system logging facility, and **`sshd`** is a daemon that serves incoming SSH connections.

```
pbmac@ubuntu $ systemctl list-unit-files --type service -all
UNIT FILE                                  STATE          VENDOR PRESET
accounts-daemon.service                    enabled        enabled
alsa-restore.service                       static         enabled
alsa-state.service                         static         enabled
alsa-utils.service                         masked         enabled
anacron.service                            enabled        enabled
apparmor.service                           enabled        enabled
apport-autoreport.service                  static         enabled
apport-forward@.service                    static         enabled
apport.service                             generated      enabled
```

This example shows some of the services running on a Linux system. Services can be listed by using the systemctl command. There are numerous options that provide a variety of output formats

In a Linux environment, the parent process of a daemon is often, but not always, the **`init`** process. A daemon is usually created either by a process forking a child process and then immediately exiting, thus causing **`init`** to adopt the child process, or by the **`init`** process directly launching the daemon. In addition, a daemon launched by forking and exiting typically must perform other operations, such as dissociating the process from any controlling terminal (tty). Such procedures are often implemented in various convenience routines such as daemon(3) in Unix.

Since 2015, the majority of Linux distributions have adopted **systemd**, having replaced other systems such as the UNIX System V and BSD **init** systems. **systemd** has faced mixed reception from Linux users over a variety of concerns.

Systems often start daemons at boot time which will respond to network requests, hardware activity, or other programs by performing some task.

Managing Daemons and Services

As a Linux administrator it is important to understand services. There is an entire lifecycle that services, and actually all Linux processes, go through. Many services get started at boot time, while others are initiated at specific times or days to perform a necessary task. We will take a look at configuring and managing these services and daemons through the use of a couple of important Linux commands.

In the Linux realm there are two methods that are used to initialize Linux. One is named SysVinit, the other is **systemd**.

As was mentioned in the previous module, Linux Boot Process, towards the end of Stage 2 GRUB2 loads the selected kernel into memory and turns control of the computer over to the kernel. All of the kernels are in a self-extracting, compressed format to save space. The kernels are located in the /boot directory, along with an initial RAM disk image, and device maps of the hard drives.

After the selected kernel is loaded into memory and begins executing, it must first extract itself from the compressed version of the file before it can perform any useful work. Once the kernel has extracted itself, it loads one of the initialization methods, and turns control over to it.

The startup process follows the boot process and brings the Linux computer up to an operational state in which it is usable for productive work.

systemd

systemd is a software suite that provides an array of system components for Linux, and is the mother of all processes. It is responsible for bringing the Linux host up to a state in which productive work can be done. Some of its functions, which are far more extensive than the old init program, are to manage many aspects of a running Linux host, including mounting filesystems, and starting and managing system services required to have a productive Linux host.

Its main aim is to unify service configuration and behavior across Linux distributions; **systemd's** primary component is a "system and service manager"—an init system used to bootstrap user space and manage user processes. It also provides replacements for various daemons and utilities, including device management, login management, network connection management, and event logging. The name **systemd** adheres to the Unix convention of naming daemons by appending the letter d.

systemd daemons make it is easier to supervise and control processes and parallelized job execution. **systemd** also provides the **systemctl** mmand and cgroups to make system administration easier: 1) **systemctl** provides the administrator with more detailed error messages including runtime and start-up errors; and cgroups - "control groups" -allow for the grouping of processes into a hierarchy for easier management.

Unit files: What are They?

The unit files on your system determine how **systemd** will start and run. Each corresponds to a single activity or component — or unit in **systemd** terms. Each unit file is a simple text file describing a unit, what it does, what needs to run before or afterward, and other details.

Unit files can be stored in a few different places on your system. **systemd** looks for system unit files in this order:

1. /etc/systemd/system
2. /run/systemd/system
3. /usr/lib/systemd/system

Unit files in the earlier directories override later ones. This is a useful scheme, because it lets you make changes in the /etc directory, where configuration is expected. You should avoid making changes in /usr. Your system installs package data there that’s not expected to change.

systemd can also run in a user context, and manage resources per user in addition to system-side. Unit files for user units are stored similarly in /etc/systemd/user, /run/systemd/user, and /usr/lib/systemd/user. The order of precedence works similarly.

You can examine some details about these unit files using the **systemctl** command. If you want to see a list of all the unit files installed on the system, run this command:

```
systemctl list-unit-files
```

Each unit file contains options in the form OptionName=value, separated into sections marked like [SectionName]. These options describe how the unit works, and how **systemd** deals with it.

There are numerous types of units **systemd** understands. The two most common for system owners to deal with are service units and target units. To list unit files on your system of each of these types, use the **systemctl** command:

```
systemctl list-unit-files --type service
systemctl list-unit-files --type target
```

Service units

These are the units that describe a process **systemd** can start and monitor. Service units are the most common units you'll use daily. They are controlled using **systemctl** as root:

```
systemctl [command] NAME.service
```

Typical commands include:

- start: starts a **systemd** unit
- stop: attempts to "nicely" end a service
- status: provides detailed information on a service
- restart: restarts (stops and then starts) the specified service
- enable: hooks (links) a unit to various places, for instance to run at boot
- disable: unhooks (unlinks) a unit, so it is not activated

The following example is the sshd.service unit file. It allows **systemd** to control the sshd daemon that allows remote access to your system via SSH (Secure Shell).

```
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
```

This unit features both some common unit file options, and service unit-specific options. Common options seen here include a description and where to find documentation. There is obviously more in this unit file than this article can explain. But as you can probably guess from the ExecStart option, this unit runs the sshd daemon when started.

Service Owned Processes

One interesting feature of **systemd** is that it monitors processes it starts with service units. To find out what processes are being monitored, use the **systemctl** status command. For instance, here is output from that command checking the status of the sshd.service unit:

```
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2015-10-20 14:51:24 EDT; 1 weeks 0 days ago
   Docs: man:sshd(8)
         man:sshd_config(5)
   Main PID: 1090 (sshd)
   CGroup: /system.slice/sshd.service
           └─1090 /usr/sbin/sshd -D
```

We can see that the process started by this service has process ID (PID) 1090. This process will continue to be monitored by **systemd**.

Notice also that the service has placed this process in a specially named control group. The control group (or "cgroup") allows **systemd** to manage associated processes together. In future articles in the series, we'll explore how this feature allows you to tune performance and resource limits.

Thanks to knowing the processes owned by this service, **systemd** can also help you manage errant services. Normally when you stop a service, you'll use the **systemctl** command as mentioned earlier. For example, to stop the web server service:

```
systemctl stop httpd.service
```

But what if the service doesn't respond or cooperate? In this case, **systemctl** has a built in kill switch:

```
systemctl kill httpd.service
```

Target Units

Target units are used to link and group other units together to describe a desired system state. Some of these units may be services. Others may be additional target units with their own groups of units.

Here's an example, the multi-user.target unit file:

```
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

Notice that this target unit file doesn't contain a command to execute. Instead, it functions purely as a way to connect other units (in this case, mostly targets) together. In this case:

- The multi-user.target requires that basic.target run successfully when multi-user.target is run.
- If the rescue.service or rescue.target units run, they will cause this unit to stop, and vice versa.
- The multi-user.target unit starts after basic.target starts, and after rescue.service and rescue.target are run, if those are started.

Additionally this target unit includes the AllowIsolate option. This option allows your system to treat the multi-user.target unit as a boot target, using the **systemctl** isolate command.

Target units group other units together not just with their unit file contents. A target can also have a .wants directory that links to units which will be started along with the target. For instance, the /usr/lib/systemd/system/multi-user.target file has an associated folder /usr/lib/systemd/system/multi-user.target.wants. This directory contains links to units (not just services but other targets as well) that will run when this target is run. Each of those may have its own dependencies, or unit file options like Requires above, as well.

Adapted from:
"Daemons" by Unknown, Wikipedia is licensed under CC BY-NC-SA 3.0
"systemd" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"An introduction to the Linux boot and startup processes" by David Both, opensource.com is licensed under CC BY-SA 4.0
"Using the systemctl command to manage systemd units" by David Both, opensource.com is licensed under CC BY-SA 4.0
"systemd unit file basics" by Bryan Sutherland, Fedora Magazine is licensed under CC BY-SA 3.0

08-C.8: Managing Linux Services is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-C.8.1: Managing Linux Services - systemctl

The systemctl Command

`systemctl` is used to examine and control the state of “systemd” system and service manager.

`systemd` is system and service manager for Unix like operating systems (most of the distributions, not all). As the system boots up, the first process created, i.e., init process with PID = 1, is **`systemd`** system that initiates the userspace services.

Syntax :

```
systemctl [ OPTIONS ] COMMAND [NAME...]
```

Command Options:

Option	Meaning
-t, --type=	The argument in this case should be comma-separated list of unit types such as service and socket.
--state=	This is used to show only specified states, for example --state=failed, will show only failed units.
-a, --all	Show all units, regardless of their states.
-r, --recursive	While listing units, show units of local containers also.
--show-types	Shows types of sockets along with showing sockets.
--job-mode	This option controls how to deal with already queued jobs in case of queuing a new job.
--fail	Similar to --job-mode=fail.
-i, --ignore-inhibitors	When you request system shutdown or sleep state, it ignores inhibitor locks.
-q, --quite	Suppresses printing of results of various commands and the hints about the truncated lines.
--no-block	Don't wait synchronously for process to finish.
--user	Talk to service manager of calling user, instead of system.
--no-wall	Don't send wall message before power-off, halt or reboot.
-f, --force	When used with enable, override any conflicting symlinks.
--message=	When used with halt, poweroff, reboot, kexec, you can display a short message specifying reason of operation.
-n, --lines	When used with status, controls the number of journal lines to show.
-H, --host=	Execute operation remotely.

systemctl has numerous sub-commands to assist in managing the various services. The following example shows the command's syntax, the command, followed by a sub-command (in the example it is stop), and then the service (NetworkManager.service):

```
systemctl stop NetworkManager.service
OR
systemctl stop NetworkManager
```

Below are some of the common sub-commands available. See the **systemctl** man page for more details.

systemctl Sub-commands:

Subcommand	Meaning
systemctl start foobar.service	Used to start a service (not reboot persistent).
systemctl stop foobar.service	Used to stop a service (not reboot persistent).
systemctl restart foobar.service	Used to stop and then start a service.
systemctl reload foobar.service	When supported, reloads the config file without interrupting pending operations.
systemctl condrestart foobar.service	Restarts if the service is already running.
systemctl status foobar.service	Tells whether a service is currently running.
ls /lib/systemd/system/*.service /etc/systemd/system/*.service	Used to list the services that can be started or stopped.
systemctl enable foobar.service	Turns the service on, for start at next boot, or other trigger.
systemctl disable foobar.service	Turns the service off for the next reboot, or any other trigger.
systemctl is-enabled foobar.service	Used to check whether a service is configured to start or not in the current environment.

Some example output - notice that the command only used "ssh" and not "sshd.service" - this is valid syntax.

```
pbmac@pbmac-server $ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enab
   Active: active (running) since Wed 2020-08-19 08:11:43 PDT; 1 weeks 1 days ag
 Main PID: 1132 (sshd)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/ssh.service
           └─1132 /usr/sbin/sshd -D
```

Adapted from:
"systemctl in Unix" by Mandeep Singh, Geeks for Geeks is licensed under CC BY-SA 4.0

08-C.8.1: Managing Linux Services - systemctl is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-C.8.2: Managing Linux Services - hostnamectl

The hostnamectl Command

`hostnamectl` command provides a proper API used to control Linux system hostname and change its related settings. The command also helps to change the hostname without actually locating and editing the /etc/hostname file on a given system.

Syntax:

```
hostnamectl [ OPTIONS ] COMMAND ....
```

Where COMMAND can be any of the following:

- status: Used to check current hostname settings.
- set-hostname NAME: Used to set system hostname.
- set-icon-name NAME: Used to set icon name for host.
- set-chassis NAME: Set chassis type for host.

Types of hostname:

- Static: Assigned by system admin and it is used to initialize the kernel hostname during boot time.
- Dynamic or Transient: Assigned by mDNS server or DHCP server during run time.
- Pretty: A high-level hostname assigned by system admin or end-user.

Command Options:

Option	Meaning
--no-ask-password	Do not query the user for authentication for privileged operations.
-H, --host=	Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":", and then a container name, separated by "/", which connects directly to a specific container on the specified host.
-M, --machine=	Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made

Adapted from:
"hostnamectl command in Linux with Examples" by Suraj1994, Geeks for Geeks is licensed under CC BY-SA 4.0

08-C.8.2: Managing Linux Services - hostnamectl is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-C.8.3: Managing Linux Services - SysVinit

SysVinit Runlevels compared to systemd Targets

The following table is a comparison of the SysVinit run levels (we will talk about them in a moment) and the **`systemd`** targets that we talked about previously.

SysV Runlevel	systemd Target	Notes
0	runlevel0.target, poweroff.target	Halt the system.
1, s, single	runlevel1.target, rescue.target	Single user mode.
2, 4	runlevel2.target, runlevel4.target, multi-user.target	User-defined/Site-specific runlevels. By default, identical to 3.
3	runlevel3.target, multi-user.target	Multi-user, non-graphical. Users can usually login via multiple consoles or via the network.
5	runlevel5.target, graphical.target	Multi-user, graphical. Usually has all the services of runlevel 3 plus a graphical login.
6	runlevel6.target, reboot.target	Reboot
emergency	emergency.target	Emergency shell

What is Single User Mode

An important run level is single user mode (run level 1), in which only the system administrator is using the machine and as few system services as possible, including logins, are running. Single user mode is necessary for a few administrative tasks such as running fsck on a /usr partition, since this requires that the partition be unmounted, and that can't happen unless just about all system services are killed.

A running system can be taken to single user mode by using **`telinit`** to request run level 1. At bootup, it can be entered by giving the word **`single`** or **`emergency`** on the kernel command line: the kernel gives the command line to **`init`** as well, and init understands from that word that it shouldn't use the default run level. (The kernel command line is entered in a way that depends on how you boot the system.)

Booting into single user mode is sometimes necessary so that one can run **`fsck`** by hand, before anything mounts or otherwise touches a broken /usr partition (any activity on a broken filesystem is likely to break it more, so **`fsck`** should be run as soon as possible).

The bootup scripts **`init`** runs will automatically enter single user mode, if the automatic **`fsck`** at bootup fails. This is an attempt to prevent the system from using a filesystem that is so broken that **`fsck`** can't fix it automatically. Such breakage is relatively rare, and usually involves a broken hard disk or an experimental kernel release, but it's good to be prepared.

init

When initializing a Linux system with sysVinit, **`init`** is the last step of the kernel boot sequence. It looks for the file /etc/inittab to see if there is an entry for initdefault. It is used to determine initial run-level of the system.

The **`telinit`** and **`runlevel`** Commands

`init` is the program responsible for altering the run level which can be called using **`telinit`** command.

For example, to change a runlevel from 3 to runlevel 5, which will actually allow the GUI to be started in multi-user mode, the **`telinit`** command can be used as:

```
# using telinit to change
# runlevel from 3 to 5

pbmac@pbmac-server # telinit 5
```

NOTE : The changing of runlevels is a task for the super user and not the normal user; that's why it is necessary to be logged in as super user for the successful execution of the above **`telinit`** command, or you can use **sudo** command as:

```
# using sudo to execute telinit
pbmac@pbmac-server # sudo telinit 5
```

The default runlevel for a system is specified in /etc/initab file which will have an entry id : 5 : initdefault if the default runlevel is set to 5 or will have an entry id : 3 : initdefault if the default runlevel is set to 3.

Need for Changing the Runlevel

- There can be a situation when you may find trouble logging in if you don't remember the password or because of the corrupted /etc/passwd file (have all the user names and passwords); in this case the problem can be solved by booting into a single user mode *i.e* runlevel 1.
- You can easily halt the system by changing the runlevel to 0 by using **`telinit`** 0.

Now verify the currently running target. It should be the same as the default target. You can still use the old method, which displays the old SystemV runlevels. Note that the previous runlevel is on the left; it is N (which means None), indicating that the runlevel has not changed since the host was booted. The number 5 indicates the current target, as defined in the old SystemV terminology:

```
pbmac@pbmac-server # runlevel
N 5
```

Note that the runlevel man page indicates that runlevels are obsolete and provides a conversion table.

Adapted from:
"9.5. Booting in single user mode" by Alex Weeks, Joanna Oja, Lars Wirzenius, Stephen Stafford, The Linux Documentation Project is in the Public Domain, CC0
"Run Levels in Linux" by Dimpy Varshni, Geeks for Geeks is in the Public Domain, CC0

08-C.8.3: Managing Linux Services - SysVinit is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-C.8.4: Managing Linux Services - inittab & init.d

The /etc/inittab File

When init starts up, it reads the **`/etc/inittab`** configuration file. While the system is running, it will re-read it, if sent the HUP signal (kill -HUP 1); this feature makes it unnecessary to boot the system in order to make changes to the init configuration take effect.

The **`/etc/inittab`** file is a bit complicated. We'll start with the simple case of configuring getty lines. Lines in /etc/inittab consist of four colon-delimited fields:

```
id:runlevels:action:process
```

The fields are described below. In addition, **`/etc/inittab`** can contain empty lines, and lines that begin with a number sign (`#'); these are both ignored.

id - This identifies the line in the file. For getty lines, it specifies the terminal it runs on (the characters after /dev/tty in the device file name). For other lines, it doesn't matter (except for length restrictions), but it should be unique.

runlevels - The run levels the line should be considered for. The run levels are given as single digits, without delimiters. (Run levels are described in the next section.)

action - What action should be taken by the line, e.g., respawn to run the command in the next field again, when it exits, or once to run it just once.

process - The Linux command to run.

What is init.d?

All these services work on several scripts and these scripts are stored in **`/etc/init.d`**. **`init`** is a daemon which is the first process of the Linux system. Then other processes, services, and daemons, are started by **`init`**. So init.d is a configuration database for the **`init`** process.

There are numerous files in the /etc/init.d directory. Each of these files represent a service, that is a daemon, that may or may not be running on the system. The files are simply Linux shell scripts that can be run either by **`init`**, or manually from the command line. The scripts all have some portions that are the same. If the service is to be started the "start" function is called, if it is to be stopped, then the "stop" function is called. So, each script has several functions to respond appropriately to the various sub-commands of service: start, stop, status, etc.

The /etc/init.d Directory

In Linux there are several services that can be started and stopped manually in the system. Some of these services are ssh, HTTP, tor, apache, etc. To start and run these services we used to simply type **service "service name" start/stop/status/restart.**

Example:

```
service ssh start
```

And to check if this service is running we type the command:

```
service ssh status
```

In this simple manner, we are using service management in Linux but what actually happens and how it actually works is in the background.

Some documentation sources suggest that administrators run the scripts directly as in:

```
/etc/init.d/ssh start
```

However, it is better to use the **service** command, since it knows where the scripts are located, and it is less typing.

The /etc/rc.local Directory

There is also a directory /etc/rc.local. It starts to get confusing as to what are all these directories and files for.

- **init.d** is the directory that stores services control scripts, which control the starting and stopping of services such as httpd, sshd, etc.
- **rc.local** is a service that allows running of arbitrary scripts as part of the system startup process.

So, if you have local scripts that you as the administrator want to run when the system is started, you can place them in the /etc/rc.local directory and they get run automatically.

The **service** Command

The **service** command is used to run a System V init script. Usually all system V init scripts are stored in /etc/init.d directory and service commands can be used to start, stop, and restart the daemons and other services under Linux. All scripts in /etc/init.d accept and support at least the start, stop, and restart commands.

Syntax:

```
service SCRIPT COMMAND [ OPTIONS ]
```

The SCRIPT is the name of one the scripts that exists in /etc/init.d. The allowable COMMAND parameters depends on the SCRIPT, but the scripts should all at least have a "start" and a "stop" capability. The COMMAND and OPTION parameters are passed directly to the script itself, where they are interpreted. In the following table each row shows the service command, "name," which is the service that is being queried (such as httpd orsshd), and then the sub-command.

Sub-commands	Meaning
service name **start**	Starts a service
service name **stop**	Stops a service
service name **restart**	Restarts a service
service name **reload**	Reloads a configuration
service name **status**	Checks whether a service is running
service –status- all	Displays the status of all services

The **chkconfig** Command

The **chkconfig** command is quite similar to the **service** command, and is used to list all available services and view or update their run level settings. It lists current startup information of services or any particular service, updating runlevel settings of service and adding or removing service from management.

Syntax:

```
chkconfig [ OPTIONS ] [ SERVICE ]  sub-command
```

In the following table the only OPTION used is --list, though there are several other options - consult your **chkcofig** man page for additional information. In these examples, replace SERVICE with the appropriate service name.

Sub-command	Meaning
chkconfig SERVICE on	Enables a service
chkconfig SERVICE off	Disables a service
chkconfig –list SERVICE	Checks to see if a service is enabled
chkconfig –list	List all services and check if they are enabled
chkconfig SERVICE reset	Restore the service's startup priority and other information to the recommended settings
chkconfig –level 5 SERVICE on	Enable the specified service at the specified runlevel (in this case runlevel 5)
chkconfig –level 5 SERVICE off	Disable the specified service at the specified runlevel (in this case runlevel 5)

Adapted from:
"What is init.d in Linux Service Management?" by Madhusudan_Soni, Geeks for Geeks is licensed under CC BY-SA 4.0
"Service command" by Multiple Contributors, nixCraft is licensed under CC BY-NC-SA 3.0

08-C.8.4: Managing Linux Services - inittab & init.d is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-D.9: Process Troubleshooting

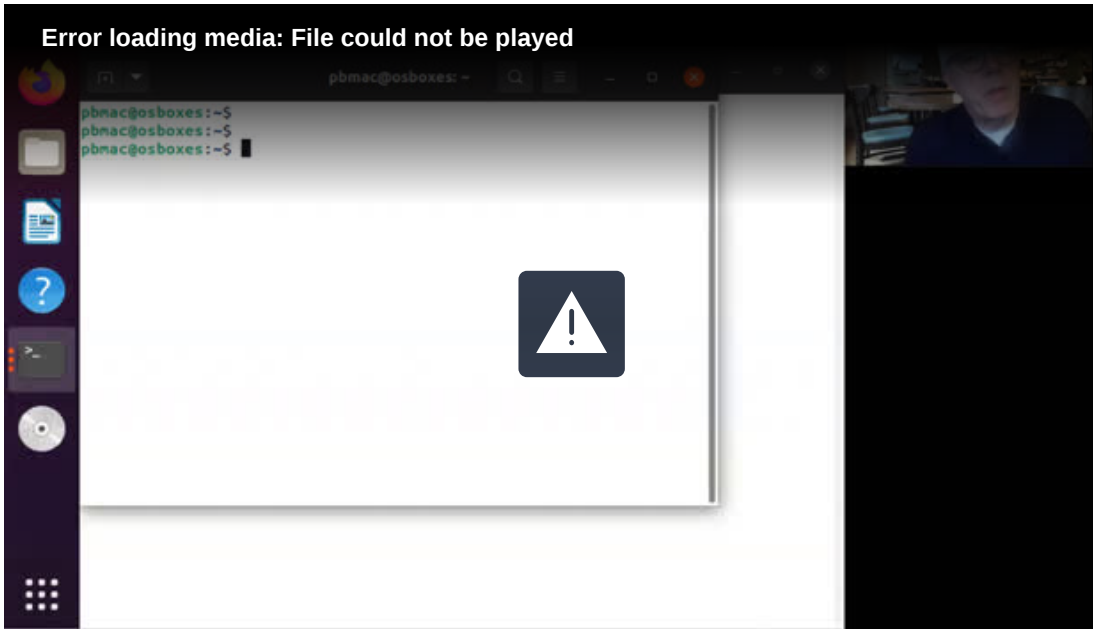
EXAM OBJECTIVES COVERED

2.4 Given a scenario, manage services.
2.6 Given a scenario, automate and schedule jobs.
4.2 Given a scenario, analyze system processes in order to optimize performance.

The Linux Process

Everything that happens in Linux is a process. Some software actually creates numerous processes during the execution of the code. Processes can get stuck. Referred to as a hung process, they can die unexpectedly, they can not terminate when they should, they can consume or cause other processes to wait while they complete...there are numerous issues that can occur with a process that impacts the overall system.

Processes are important in Linux, so the video discusses processes and things system administrators need to be familiar with.



Process States

A process goes through a lifecycle from "new" to "terminated." It is important to know what state a process is in so that you, as the system administrator, know what actions to take.

As a process executes it changes state according to its circumstances. Linux processes are in one the following states:

- **Running** - The process is either running (it is the current process in the system) or it is ready to run (it is waiting to be assigned to one of the system's CPUs).
- **Waiting** - The process is waiting for an event or for a resource.
  - Linux differentiates between two types of waiting processes:
    - **Interruptible** waiting processes can be interrupted by signals, such as input from the terminal.
    - **Uninterruptible** waiting processes are waiting directly on hardware conditions and cannot be interrupted under any circumstances.
- **Stopped** - The process has been stopped, usually by receiving a signal. A process that is being debugged can be in a stopped state.
- **Zombie** - This is a halted process which, for some reason, still has a task_struct data structure in the task vector. It is what it sounds like, a dead process.

Process ID's

A program/command when executed, a special instance is provided by the system to the process. This instance consists of all the services/resources that may be utilized by the process under execution.

- Whenever a command is issued in Linux it creates/starts a new process. For example, when the command **pwd** is issued, which is used to list the current directory location the user is in, a process starts.

- Through a five digit ID number, Linux keeps account of the processes. This number is called process ID or PID. Each process in the system has a unique PID.
- Used up PID's can be used again for a newer process since all the possible combinations are used.
- At any point in time, no two processes with the same PID exist in the system because it is the PID that Unix uses to track each process.

To find the PID of a process there are several commands that can be used. The most common is the **ps** command.

```
pbmac@pbmac-server $ ps
  PID TTY          TIME CMD
 7582 pts/1    00:00:00 ps
11845 pts/1    00:00:00 bash
```

This outputs the PID, the terminal session it is associated with, the CPU time consumed by the command, and the command itself. The most common **ps** options are **ps -ef,** or **ps-ax.**

In order to troubleshoot process issues it is imperative to be able to find the process ID. With the PID you can then perform process management commands, or if necessary, the process can be terminated by the use of the PID.

The **pgrep** Command

When using the pgrep command, it looks through the currently running processes and lists the PIDs which matches the selection criteria to stdout. All the criteria have to match. For example, **pgrep -u root sshd** will only list the processes called sshd AND owned by **root**. On the other hand, **pgrep -u root,daemon** will list the processes owned by root OR daemon.

Syntax:

```
pgrep [ OPTIONS ] {pattern}
```

If you needed to find the process ID of a certain command that is running you can simply use the command name. Or perhaps you need all of the processes being run by a specific user, which you can do.

```
pbmac@pbmac-server $ pgrep sshd
1064
pbmac@pbmac-server $ psgrep chrome
30773
pbmac@pbmac-server $ ps -ef | grep 30773
pbmac    30773  3106  0 Sep10 tty2     00:00:01 /opt/google/chrome/chrome
pbmac@pbmac-server $
```

Adapted from:
"Introduction of Process Management" by Nilesh Singh 2, Geeks for Geeks is licensed under CC BY-SA 4.0

08-D.9: Process Troubleshooting is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-D.9.1: Process Troubleshooting - ps command

The **ps** Command

Linux provides a utility called **ps** (which stands as abbreviation for **"**Process Status**"**) for viewing information related with the processes on a system**.** The **ps** command is used to list the currently running processes and their PIDs along with some other information, which depends on different options. It reads the process information from the virtual files in /proc file-system. /proc contains virtual files, which is the reason it's referred as a virtual file system.

ps provides numerous options for manipulating the output according to our need.

Syntax:

```
ps [ OPTIONS ]
```

Command Options:

Option	Meaning
-a	Write information for all processes associated with terminals. Implementations may omit session leaders from this list.
-A	Write information for all processes.
-d	Write information for all processes, except session leaders.
-e	Write information for all processes. (Equivalent to -A.)
-f	Generate a full listing. (See the STDOUT section for the contents of a full listing.)
-g grouplist	Write information for processes whose session leaders are given in grouplist. The application shall ensure that the grouplist is a single argument in the form of a <blank> or <comma>-separated list.
-G grouplist	Write information for processes whose real group ID numbers are given in grouplist. The application shall ensure that the grouplist is a single argument in the form of a <blank> or <comma>-separated list.
-l	Generate a long listing. (See STDOUT for the contents of a long listing.)
-n namelist	Specify the name of an alternative system namelist file in place of the default. The name of the default file and the format of a namelist file are unspecified.
-o format	Write information according to the format specification given in format. This is fully described in the STDOUT section. Multiple -o options can be specified; the format specification shall be interpreted as the <space>-separated concatenation of all the format option-arguments.
-p proclist	Write information for processes whose process ID numbers are given in proclist. The application shall ensure that the proclist is a single argument in the form of a <blank> or <comma>-separated list.

Option	Meaning
-t termlist	Write information for processes associated with terminals given in termlist. The application shall ensure that the termlist is a single argument in the form of a <blank> or <comma>-separated list. Terminal identifiers shall be given in an implementation-defined format. On XSI-conformant systems, they shall be given in one of two forms: the device's filename (for example, tty04) or, if the device's filename starts with tty, just the identifier following the characters tty (for example, "04").
-u userlist	Write information for processes whose user ID numbers or login names are given in userlist. The application shall ensure that the userlist is a single argument in the form of a <blank> or <comma>-separated list. In the listing, the numerical user ID shall be written unless the -f option is used, in which case the login name shall be written.
-U userlist	Write information for processes whose real user ID numbers or login names are given in userlist. The application shall ensure that the userlist is a single argument in the form of a <blank> or <comma>-separated list.

```
pbmac@pbmac-server $ ps
  PID TTY          TIME CMD
11845 pts/1    00:00:00 bash
17636 pts/1    00:00:00 ps
```

The result contains four columns of information:
PID – the unique process ID.
TTY – terminal type that the user running this command is logged into.
TIME – amount of CPU in minutes and seconds that the process has been running.
CMD – name of the command that launched the process.

Note – Sometimes when we execute **`ps`** command, it shows TIME as 00:00:00. It is nothing but the total accumulated CPU utilization time for any process and 00:00:00 indicates no CPU time has been given by the kernel till now. In the above example we found that for bash no CPU time has been given. This is because bash is just a parent process for different processes which needs bash for their execution, and bash itself is not utilizing any CPU time till now.

There are numerous options for the **ps** command - and the output is different with the different options.

```
pbmac-server $ ps -ax
PID TTY      STAT   TIME COMMAND
  1 ?        Ss     0:27 /sbin/init
  2 ?        S      0:00 [kthreadd]
  6 ?        I<     0:00 [mm_percpu_wq]
  7 ?        S      0:07 [ksoftirqd/0]
```

Notice the TTY is a question mark, because this process is NOT associated with a terminal session. The STAT column refers to what state (as previously discussed) the process is in.

```
pbmac@pbmac-server $ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Sep09 ?        00:00:27 /sbin/init
root         2     0  0 Sep09 ?        00:00:00 [kthreadd]
root         6     2  0 Sep09 ?        00:00:00 [mm_percpu_wq]
root         7     2  0 Sep09 ?        00:00:07 [ksoftirqd/0]
```

Using the -ef options gives a slightly different output:

UID - the user who owns this process
PPID - the parent process ID - when the PPID is 0 that means the process was started by the boot process.

There are quite a number of options - the **ps -ax** and **ps -ef** commands shown above are quite common. You are encouraged to view the **ps** man page and familiarize yourself with some of the other options, as there are just too many to cover here.

The **ps** command accepts three styles of options:

1. UNIX options, which may be grouped and must be preceded by a dash.
2. BSD options, which may be grouped and must not be used with a dash.
3. GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and **ps** implementations that this **ps** is compatible with.

Note that **ps -aux** is distinct from **ps aux**. The POSIX and UNIX standards require that **ps -aux** print all processes owned by a user named **x**, as well as printing all processes that would be selected by the **-a** option. If the user named **x** does not exist, this **ps** may interpret the command as **ps aux** instead and print a warning. This behavior is intended to aid in transitioning old scripts and habits. It is fragile, subject to change, and thus should not be relied upon.

Adapted from:
"ps command in Linux with Examples" by Mausami_Inhe, Geeks for Geeks is licensed under CC BY-SA 4.0

08-D.9.1: Process Troubleshooting - ps command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-D.9.2: Process Troubleshooting - top command

The `top` Command

The **`top`** command is used to show the active Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux kernel.

Running this command will open an interactive command mode window where the top half portion will contain the statistics of processes and resource usage. The lower half contains a list of the currently running processes. Pressing q will simply exit the command mode.

```
top - 18:27:57 up 5 days, 32 min,  1 user,  load average: 2.64, 2.53, 2.05
Tasks: 363 total,   2 running, 313 sleeping,   0 stopped,   0 zombie
%Cpu(s):  8.5 us,  1.9 sy,  0.0 ni, 88.7 id,  0.9 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 12228648 total,  1462820 free,  9597568 used,  1168260 buff/cache
KiB Swap:     8188 total,     1284 free,     6904 used.  1672064 avail Mem
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
22935 pbmac     20   0   43000   3768   3152 R  11.8  0.0   0:00.04 top
   16 root      20   0       0      0      0 S   5.9  0.0   0:08.33 ksoftirqd/1
 3478 pbmac     20   0 5135080 476252  77796 S   5.9  3.9 199:36.14 chrome
16330 pbmac     20   0 4974592 149692  17480 S   5.9  1.2  54:41.58 chrome
22890 pbmac     20   0 4792848 125380  82060 S   5.9  1.0   0:02.57 chrome
    1 root      20   0  160616   3636    508 S   0.0  0.0   0:28.53 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.06 kthreadd
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_+
    7 root      20   0       0      0      0 S   0.0  0.0   0:08.06 ksoftirqd/0
    8 root      20   0       0      0      0 I   0.0  0.0   3:35.28 rcu_sched
    9 root      20   0       0      0      0 I   0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.03 migration/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.62 watchdog/0
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:00.55 watchdog/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.03 migration/1
```

The columns are labeled:

- PID: Shows task's unique process ID.
- USER: User name of owner of task.
- PR: Stands for priority of the task.
- NI: Represents a Nice Value of task. A negative nice value implies higher priority, and positive nice value means lower priority.
- VIRT: Total virtual memory used by the task.
- RES: Represents the amount of actual physical memory a process is consuming.
- SHR: Represents the Shared Memory size (kb) used by a task.
- S: The state the process is in.
- %CPU: Represents the CPU usage.
- %MEM: Shows the Memory usage of task.
- TIME+: CPU Time, the same as 'TIME,' but reflecting more granularity through hundredths of a second.

- COMMAND: The command that is being run.

Above the list of processes, there's a whole bunch of other useful information. Some of these details may look strange and confusing, but once you take some time to step through each one, you'll see they are very useful stats to pull up in a pinch.

The first row contains general system information:

- top: this is simply the command name...followed by:
- XX:YY:XX : the time, updated every time the screen updates.
- up (then X day, YY:ZZ): the system's uptime, or how much time has passed since the system turned on.
- load average (then three numbers): the system load over the last one, five, and 15 minutes, respectively.

The second row (Tasks) shows information about the running tasks, and it's fairly self-explanatory. It shows the total number of processes and the number of running, sleeping, stopped, and zombie processes. This is literally a sum of the S (state) column described above.

The third row (%Cpu(s)) shows the CPU usage separated by types. The data are the values between screen refreshes. The values are:

- us : user processes
- sy : system processes
- ni : nice user processes
- id : the CPU's idle time; a high idle time means there's not a lot going on otherwise
- wa : wait time, or time spent waiting for I/O completion
- hi : time spent waiting for hardware interrupts
- si : time spent waiting for software interrupts
- st : "time stolen from this VM by the hypervisor"

You can collapse the Tasks and %Cpu(s) rows by typing t (for "toggle").

The fourth (KiB Mem) and fifth rows (KiB Swap) provide information for memory and swap. These values are:

- total
- used
- free

But also:

- memory buffers
- swap cached Mem

Adapted from:
"Tips for using the top command in Linux" by Katie McLaughlin, opensource.com is licensed under CC BY-SA 4.0

08-D.9.2: Process Troubleshooting - top command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-D.9.3: Process Troubleshooting - lsof Command

The **systemd-analyze** Command

Improving the boot performance of a system can provide reduced boot wait times and a means to learn more about how certain system files and scripts interact with one another. Linux provides a tool called **systemd-analyze** that can be used to show timing details about the boot process, including an svg plot showing units waiting for their dependencies. You can see which unit files are causing your boot process to slow down.

To see how much time was spent in kernelspace and userspace on boot, simply use:

```
$ systemd-analyze
```

Tip: If you boot via UEFI and use a boot loader which implements systemd's Boot Loader Interface (which currently systemd-boot and GRUB do), **systemd-analyze** can additionally show you how much time was spent in the EFI firmware and the boot loader itself.

To list the started unit files, sorted by the time each of them took to start up:

```
$ systemd-analyze blame
```

The output looks like the example below:

```
pbmac@pbmac-server $ systemd-analyze blame
         13.085s udisks2.service
         11.744s systemd-fsck@dev-disk-by\x2duuid-ad74f5a8\x2d188a\x2d4282\x2d93
         10.343s uml-utilities.service
         10.275s NetworkManager-wait-online.service
          9.368s NetworkManager.service
          7.723s systemd-fsck@dev-disk-by\x2duuid-3709f1b7\x2d6417\x2d413e\x2da4
          7.439s accounts-daemon.service
          6.079s ModemManager.service
          5.377s colord.service
          5.168s dev-sda2.device
          5.044s dev-sda1.device
          4.864s apt-daily.service
          4.745s snapd.service
          4.393s systemd-journal-flush.service
```

The first column is simply the time it took to execute a service, and the second column is the service that was being started.

The **lsof** Command

Linux considers everything as files and maintains folders. So “Files or a File ” is very important in Linux. While working in Linux system there might be several files and folders which are being used; some of them would be visible and some not. The **lsof** command stands for List of Open Files. This command provides a list of files that are currently open. Basically, it gives the information to find out the files which are open by which process. With one go, it lists out all open files to the output console. It cannot only list common regular files but it can list a directory, a block special file, a shared library, a character special file, a regular pipe, a named pipe, an internet socket, a UNIX domain socket, and many others. It can be combined with **grep** command and can be used to do advanced searching and listing.

Syntax:

```
$lsof [ OPTIONS ][user name]
```

There are numerous options for this command. It is suggested that the user look at the man pages for specifics about which options you want to use.

The **lsof** command prints each file that has been opened by a process.The output information is:

- The name of the command or process
- The PID
- The user who issued the command
- The file descriptor - FD
- The type of the file
- The device where the file exists
- The size of the file
- The name of the file

```
pbmac@pbmac-server $ lsof
COMMAND     PID           USER   FD      TYPE             DEVICE  SIZE/OFF       NODE
systemd       1           root  cwd       DIR               8,1      4096          2
systemd       1           root  rtd       DIR               8,1      4096          2
systemd       1           root  txt       REG               8,1   1612152      13162
systemd       1           root  mem       REG               8,1   1700792      13196
systemd       1           root  mem       REG               8,1    121016      13100
systemd       1           root  mem       REG               8,1     43304      13085
systemd       1           root  mem       REG               8,2     34872      40167
systemd       1           root  mem       REG               8,1     18680      13167
```

Adapted from:
"Improving performance/Boot process" by Multiple contributors is in the Public Domain, CC0"
lsof command in Linux with Examples" by Bhumika_Rani, Geeks for Geeks is licensed under CC BY-SA 4.0

08-D.9.3: Process Troubleshooting - lsof Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-D.9.4: Process Troubleshooting - Process Priorities

Working with nice and renice Command

The nice command in Linux helps in execution of a program/process with modified scheduling priority. It launches a process with a user-defined scheduling priority. In this, if we give a process a higher priority, then Kernel will allocate more CPU time to that process, whereas the renice command allows you to change and modify the scheduling priority of an already running process. Linux Kernel schedules the process and allocates CPU time accordingly for each of them.

- To check the nice value of a process:

```
pbmac@pbmac-server $ ps -el | grep terminal
0 S  1000 13197  2034  0  80   0 - 201301 poll_s ?       00:00:14 gnome-terminal-
```

The highlighted value is the nice value of the process - in this case 0.

- To set the priority of a process:

```
pbmac@pbmac-server $ nice -10 gnome-terminal
pbmac@pbmac-server $ ps -el | grep terminal
4 S     0 13197  2034  1  70 -10 - 160436 poll_s ?       00:00:00 gnome-terminal-
```

This will set the nice value of the specified process.

- To change the priority of the running process:

```
pbmac@pbmac-server $ sudo renice -n 15 -p 13197
pbmac@pbmac-server $ ps -el | grep terminal
4 S     0 13197  2034  0  95  15 - 160836 poll_s ?       00:00:00 gnome-terminal-
```

This will change the priority of the process with PID 13197

- To change the priority of all programs of a specific group.

```
pbmac@pbmac-server $ sudo renice -n 10 -g 4
```

This command will set all the processes of GID 4 priority to 10.

- To change the priority of all programs of a specific user:

```
pbmac@pbmac-server $ sudo renice -n 10 -u 2
```

This will set all the processes of UID 2 to 10.

Adapted from:
"Nice and Renice Command in Linux with Examples" by manav014, Geeks for Geeks is licensed under CC BY-SA 4.0

08-D.9.4: Process Troubleshooting - Process Priorities is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-D.9.5: Process Troubleshooting - Background / Foreground Processes

Foreground Process

A foreground process is a process where a command or task is run directly from GUI or the command line and waits for it to finish. Some foreground processes utilize a user interface that allows for ongoing user interaction, others execute a task and simply wait while the task completes. If it is a long running task, in some instances you are not allowed to use the interface or the terminal until the task completes.

A simple example would be the **`ls`** command from the terminal. Until the listing of files is complete you are not able to enter additional commands from that terminal.

Background Process

However, when a background process is launched the shell does not have to wait for the process to complete before it can run more processes. More background processes can be started, one after the other - it is only limited by the amount of memory available. To run a command as a background process, type the command and add a space and an ampersand to the end of the command.

Managing Foreground/Background Processes

In the following example comments have been placed between the command to help explain what is happening:

```
# the yes command is executed, output is redirectd to /dev/null
# a CONTROL-Z is entered from the keyboard - Stopping the execution of the process.
pbmac@pbmac-server $ yes >/dev/null
^Z
[2]+  Stopped                 yes > /dev/null
# Using the jobs command we can see we have a Stopped job,
# the CONTROL-Z stops the job
pbmac@pbmac-server $ jobs
[2]+  Stopped                 yes > /dev/null

# Now - start another process - using the & at the end we
# place this command in the background at the time of execution
# the output shows us the job number and the PID
pbmac@pbmac-server $ yes >/dev/null &
[3] 16229

# Using the jobs command we now see we have 2 command
# one is Stopped, one is Running in the background.
pbmac@pbmac-server $ jobs
[2]+  Stopped                 yes > /dev/null
[3]-  Running                 yes > /dev/null &

# Using the bg command we place job number 2 in the background
pbmac@pbmac-server $ bg %2
[2]+ yes > /dev/null &

# Both jobs are now running in the background
pbmac@pbmac-server $ jobs
[2]-  Running                 yes > /dev/null &
```

```
[3]+  Running                 yes > /dev/null &

# We can kill a job using the job number
pbmac@pbmac-server $ kill %3
[3]+  Terminated              yes > /dev/null
pbmac@pbmac-server $ jobs
[2]+  Running                 yes > /dev/null &

# At this point we only have a single job running
# using the fg comand we bring it back to the foreground
# then kill it with a CONTROL-C
pbmac@pbmac-server $ fg %2
yes > /dev/null
^C
pbmac@pbmac-server $
```

Adapted from:
"Process Control Commands in Unix/Linux" by Shivani Ghughtyal, Geeks for Geeks is licensed under CC BY-SA 4.0

08-D.9.5: Process Troubleshooting - Background / Foreground Processes is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-D.9.6: Process Troubleshooting - nohup and kill Commands

The **nohup** Command

The **nohup** command executes some other program that is specified as an argument and ignores all SIGHUP (hangup) signals. SIGHUP is a signal that is sent to a process when its controlling terminal is closed.

For instance, if a user is connected via SSH, and the connection drops or the user logs out, the session is terminated, and all the processes executed from the terminal will terminate. This is where the **nohup** command can be used - it ignores all hangup signals, and the process will continue to run.

Syntax:

```
nohup command arguments
```

The `kill, pkill, and killall` Command

There are three different commands in Linux which are used to terminate processes manually. **`kill`** command sends a signal to a process which terminates the process. If the user doesn't specify any signal which is to be sent along with **`kill`** command then default TERM signal is sent that terminates the process.

The **`kill`** and **`pkill`** commands send signals to processes directing them to terminate. Each signal has a number, name, and an associated event. Below are some of the most commonly used signals with their functionalities. A list of the various Linux signals can be found by issuing the **`kill`** command with the -l option.

```
pbmac@pbmac-server $ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT       4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE        9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE      14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT      19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG       24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH     29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1   36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6   41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11  46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14  51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9   56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4   61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Both the **`kill`** and **`killall`** commands use PIDs to determine which process is to receive the signal. The **`killall`** command terminates all programs that match a specified name.

```
pbmac@pbmac-server $ ps -ef | grep sleep
root      1337  1218  0 07:33 pts/0    00:00:00 sleep 400
pbmac@pbmac-server $ kill -9 1337
pbmac@pbmac-server $ ps -ef | grep sleep
pbmac@pbmac-server $
```

When using the **`pkill`** command the use of the command name that is to receive the signal is used.

```
pbmac@pbmac-server $ pkill -9 firefox
```

Kill Signals

Signal Name	Signal Value	Signal Meaning
SIGINT	2	This signal is the same as pressing CTRL-C. On some systems, "delete" + "break" sends the same signal to the process. The process is interrupted and stopped. However, the process can ignore this signal.
SIGKILL	9	The SIGKILL signal forces the process to stop executing immediately. The program cannot ignore this signal. This process does not get to clean-up either.
SIGTERM	15	This signal requests a process to stop running. This signal can be ignored. The process is given time to gracefully shutdown. When a program gracefully shuts down, that means it is given time to save its progress and release resources. In other words, it is not forced to stop. SIGINT is very similar to SIGTERM.
SIGSTOP	17, 19, 23	This signal makes the operating system pause a process's execution. The process cannot ignore the signal.
SIGSTP	18, 20, 24	This signal is like pressing CTRL-Z. This makes a request to the terminal containing the process to ask the process to stop temporarily. The process can ignore the request.

For other signals, information can be found on the **`kill`** man page, the **`signal`** man page, and various web sites.

08-D.9.6: Process Troubleshooting - nohup and kill Commands is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-E.10: CPU and Memory Troubleshooting

EXAM OBJECTIVES COVERED

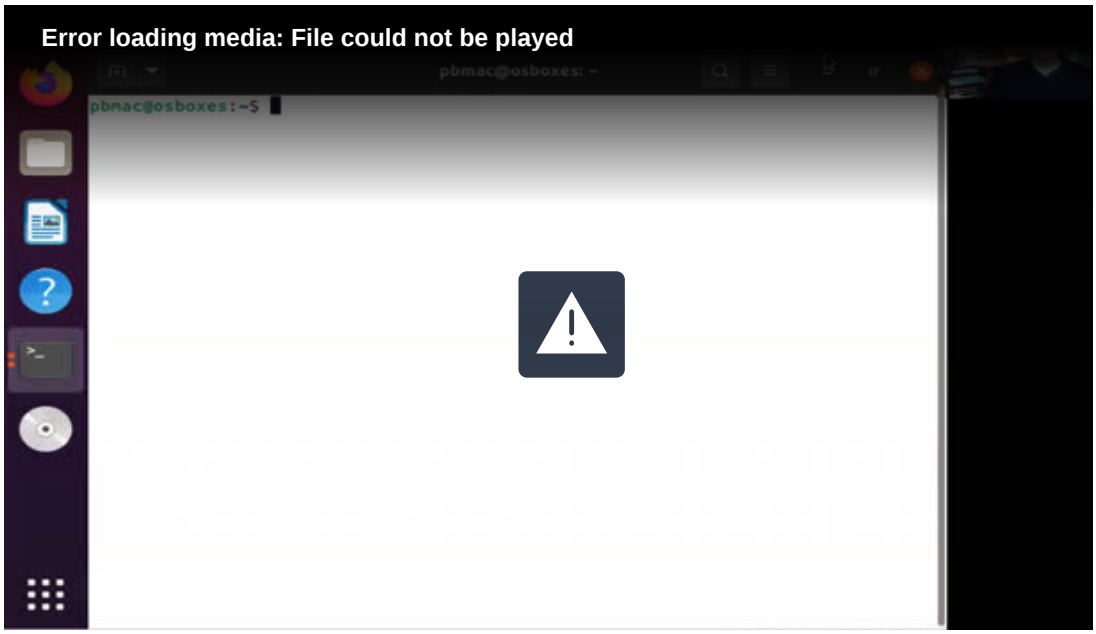
4.1 Given a scenario, analyze system properties and remediate accordingly.

CPU Issues

There are numerous potential issues that could affect or be caused by the CPU. These issues may include:

- The CPU may be under-performing, because it may be overloaded
- On multi-systems it is possible that one of the CPUs/Cores is not functioning properly
- Due to the type and number of processes, one process may be consuming large amounts of CPU time, leaving inadequate processor time for other processes
- The BIOS may not have virtualization or hyperthreading turned on
- For some reason the CPU may be sitting idle too much of the time

This video discusses the /proc directory as well as some other system administration commands.



The /proc/cpuinfo File

/proc/cpuinfo is a short, read-only, plain text file that contains information about the CPUs (central processing units) on a computer.

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 23
model name      : Intel(R) Core(TM)2 Duo CPU     E8400  @ 3.00GHz
stepping        : 10
microcode       : 0xa0b
cpu MHz         : 2704.609
cache size      : 6144 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
```

```
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush
```

Some of the more useful fields in the example output are:

- **processor** – Provides each processor with an identifying number. If you have one processor it will display a 0. If you have more than one processor it will display all processor information separately counting the processors using zero notation.
- **cpu family** – Authoritatively tells you the type of processor you have in the system. If your computer is an Intel-based system, simply place the number in front of “86” to determine the value. This is helpful to determine the type of architecture of an older system and is helpful in determining which compiled RPM package would best suit that system.
- **model name** – Gives you the common name of the processor, including the project name.
- **cpu MHz** – Shows the processor’s precise speed, in megahertz, to the thousandth decimal point.
- **cache size** – Tells you the amount of level 2 memory cache available to the processor.
- **flags** – Defines a number of different processor attributes, such as the presence of a floating-point unit (FPU) and the ability to process MMX instructions.

The `uptime` Command

The **`uptime`** command is used to find out how long the system is active (running). This command returns a set of values that involve the current time and the amount of time system is in running state, number of users currently logged into, and the load time for the past 1, 5, and 15 minutes respectively.

Syntax:

```
uptime [ OPTIONS ]
```

A couple of examples show the output in two different formats. There are additional formats based on other options, which can be found on the **`uptime`** man page.

```
pbmac@pbmac-server $ uptime
 16:53:01 up 5 days, 22:57,  1 user,  load average: 1.51, 1.35, 1.18
pbmac@pbmac-server $ uptime -p
up 5 days, 22 hours, 57 minutes
```

Adapted from:
"Linux | Uptime command with examples" by mistersubha, Geeks for Geeks is licensed under CC BY-SA 4.0

08-E.10: CPU and Memory Troubleshooting is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-E.10.1: CPU and Memory Troubleshooting - sar & Memory Issues

The **sar** Command

The **sar** command can be used to monitor Linux system's resources like CPU usage, memory utilization, I/O devices consumption, network monitoring, Disk usage, process and thread allocation, battery performance, plug and play devices, processor performance, file system and more. Linux system monitoring and analyzing aids in understanding system resource usage, which can help to improve system performance to handle more requests.

By default **sar** command displays results on the output screen; in addition results can also be stored in the file specified by the -o filename option. Any user can collect information about system performance using system activities flags. The **sar** command will show only CPU monitoring activity if any flag is not specified by the user.

```
pbmac@pbmac-server $ sar -u 2 5
Linux 4.15.0-91-generic (pbmac-server)     09/15/2020     _x86_64_    (2 CPU)

05:26:47 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
05:26:49 PM     all     13.99      0.00      1.02      1.02      0.00     83.97
05:26:51 PM     all      6.94      0.00      0.51      0.00      0.00     92.54
05:26:53 PM     all      7.42      0.00      1.02      1.53      0.00     90.03
05:26:55 PM     all      6.89      0.00      1.28      0.51      0.00     91.33
05:26:57 PM     all      7.73      0.00      0.77      1.55      0.00     89.95
Average:        all      8.60      0.00      0.92      0.92      0.00     89.55
```

Common Memory Issues

There are numerous issues that can be caused by memory problems, or that are related to memory issues.

- It is possible to load down a system to the point that there is not enough total memory to service all processes at once.
- If the system is a busy system, or does not have sufficient memory, then there may not be enough free memory to service new processes.
- There is the potential for processes to be unable to access memory even though it is available.
- Swap partitions can up too much or not enough storage space.
- Other memory issues can impact the system, but there are tools to help find and resolve memory issues.

The /proc/meminfo File

This file contains all the data about the memory usage. It provides the current memory usage details rather than old stored values. It is one of the more commonly used files in the /proc directory, as it reports a large amount of valuable information about the systems' RAM usage.

The following sample /proc/meminfo virtual file is from a system with 256 MB of RAM and 512 MB of swap space:

```
MemTotal:       255908 kB
MemFree:         69936 kB
Buffers:         15812 kB
Cached:         115124 kB
SwapCached:          0 kB
Active:          92700 kB
Inactive:        63792 kB
HighTotal:           0 kB
```

```
HighFree:            0 kB
LowTotal:       255908 kB
LowFree:         69936 kB
SwapTotal:      524280 kB
SwapFree:       524280 kB
Dirty:               4 kB
Writeback:           0 kB
Mapped:          42236 kB
Slab:            25912 kB
Committed_AS:   118680 kB
PageTables:       1236 kB
VmallocTotal:  3874808 kB
VmallocUsed:      1416 kB
VmallocChunk:  3872908 kB
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:     4096 kB
```

Much of the information here is used by the **free**, **top**, and **ps** commands. In fact, the output of the **free** command is similar in appearance to the contents and structure of **/proc/meminfo**. But by looking directly at **/proc/meminfo**, more details are revealed:

- MemTotal — Total amount of physical RAM, in kilobytes.
- MemFree — The amount of physical RAM, in kilobytes, left unused by the system.
- Buffers — The amount of physical RAM, in kilobytes, used for file buffers.
- Cached — The amount of physical RAM, in kilobytes, used as cache memory.
- SwapCached — The amount of swap, in kilobytes, used as cache memory.
- Active — The total amount of buffer or page cache memory, in kilobytes, that is in active use. This is memory that has been recently used and is usually not reclaimed for other purposes.
- Inactive — The total amount of buffer or page cache memory, in kilobytes, that is free and available. This is memory that has not been recently used and can be reclaimed for other purposes.
- HighTotal and HighFree — The total and free amount of memory, in kilobytes, that is not directly mapped into kernel space. The HighTotal value can vary based on the type of kernel used.
- LowTotal and LowFree — The total and free amount of memory, in kilobytes, that is directly mapped into kernel space. The LowTotal value can vary based on the type of kernel used.
- SwapTotal — The total amount of swap available, in kilobytes.
- SwapFree — The total amount of swap free, in kilobytes.
- Dirty — The total amount of memory, in kilobytes, waiting to be written back to the disk.
- Writeback — The total amount of memory, in kilobytes, actively being written back to the disk.
- Mapped — The total amount of memory, in kilobytes, which have been used to map devices, files, or libraries using the mmap command.
- Slab — The total amount of memory, in kilobytes, used by the kernel to cache data structures for its own use.
- Committed_AS — The total amount of memory, in kilobytes, estimated to complete the workload. This value represents the worst case scenario value, and also includes swap memory.
- PageTables — The total amount of memory, in kilobytes, dedicated to the lowest page table level.
- VMallocTotal — The total amount of memory, in kilobytes, of total allocated virtual address space.
- VMallocUsed — The total amount of memory, in kilobytes, of used virtual address space.
- VMallocChunk — The largest contiguous block of memory, in kilobytes, of available virtual address space.

- HugePages_Total — The total number of hugepages for the system. The number is derived by dividing Hugepagesize by the megabytes set aside for hugepages specified in /proc/sys/vm/hugetlb_pool. This statistic only appears on the x86, Itanium, and AMD64 architectures.
- HugePages_Free — The total number of hugepages available for the system. This statistic only appears on the x86, Itanium, and AMD64 architectures.
- Hugepagesize — The size for each hugepages unit in kilobytes. By default, the value is 4096 KB on uniprocessor kernels for 32 bit architectures. For SMP, hugemem kernels, and AMD64, the default is 2048 KB. For Itanium architectures, the default is 262144 KB. This statistic only appears on the x86, Itanium, and AMD64 architectures.

Adapted from:
"C.2.18. /proc/meminfo" by Multiple Contributors, Fedora Docs is licensed under CC BY-SA 4.0

08-E.10.1: CPU and Memory Troubleshooting - sar & Memory Issues is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-E.10.2: CPU and Memory Troubleshooting - free / vmstat Commands

The `free` Command

Linux provides the **`free`** command which displays the total amount of free space available, along with the amount of memory used and swap memory in the system, and also the buffers used by the kernel. The **`free`** command actually gets this information from the /proc/meminfo file discussed above.

Syntax:

```
free [ OPTION ]
```

As **`free`** displays the details of the memory related to your system, its syntax doesn't need any arguments to be passed, but only options which you can use according to your wish.

```
pbmac@pbmac-server $ free
              total        used        free      shared  buff/cache   available
Mem:       12228648    10094296      549120      656608     1585232     1195616
Swap:          8188        8188           0
```

When no option is used then **`free`** command produces the columnar output as shown above, where column:

- **total** displays the total installed memory (MemTotal and SwapTotal *i.e* present in /proc/meminfo).
- **used** displays the used memory.
- **free** displays the unused memory.
- **shared** displays the memory used by tmpfs(Shmen *i.e*present in /proc/meminfo and displays zero in case not available).
- **buffers** displays the memory used by kernel buffers.
- **cached** displays the memory used by the page cache and slabs (Cached and Slab available in /proc/meminfo).
- **buffers/cache** displays the sum of buffers and cache.

Command Options:

Options	Meaning
-b, – -bytes	Displays the memory in bytes.
-k, – -kilo	Displays the amount of memory in kilobytes (default).
-m, – -mega	Displays the amount of memory in megabytes.
-g, – -giga	Displays the amount of memory in gigabytes.
– – tera	Displays the amount of memory in terabytes.
-h, – -human	Shows all output columns automatically scaled to shortest three digit unit and displays the units also of print out. The units used are B(bytes), K(kilos), M(megas), G(gigas), and T(teras).
-l, – -lohi	Shows the detailed low and high memory statistics.
-o, – -old	Disables the display of the buffer adjusted line.
-s, – -seconds	Allows you to display the output continuously after s seconds delay. In actual, the usleepsystem call is used for microsecond resolution delay times.
-t, – -total	Adds an additional line in the output showing the column totals.

The `vmstat` Command

The vmstat command in Linux gives information about processes, memory, paging, block IO, disk and CPU scheduling.

Syntax:

```
vmstat [ OPTIONS ][delay [count]]
```

Delay: This is the time interval in between two updates. It can be termed as a sampling period after each interval of the sampling period report that will be printed with the updated details. If there will be no delay specified only one report will be printed with average value since boot.

Count: This is the number of updates which is printed after each interval (sampling period). In the absence of count and delay is specified, the default value of count is infinite.

Command Optionssssss

Options	Meaning
delay	The delay between updates in seconds. If no delay is specified, only one report is printed with the average values since boot.
count	Number of updates. In absence of count, when delay is defined, default is infinite.
-a, --active	Display active and inactive memory, given a 2.5.41 kernel or better.
-f, --forks	The -f switch displays the number of forks since boot.
-m, --slabs	Displays slabinfo.
-n, --one-header	Display the header only once rather than periodically.
-s, --stats	Displays a table of various event counters and memory statistics. This display does not repeat.
-d, --disk	Report disk statistics (2.5.70 or above required).
-t, --timestamp	Append timestamp to each line
-w, --wide	Wide output mode. The output is wider than 80 characters per line.

```
pbmac@pbmac-server $ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0   8188 1114688  54256 1023088    0    0    20    69   15    0  8  2 89  1  0
```

There are two main stats shown in this example: 1) Memory stats - which includes the swap data and 2) CPU stats.

- Free: Specifies the amount of free memory/idle memory spaces which are not being used.
- si: Memory that is swapped in every second from disk in kilobytes.
- so: Memory that is swapped out every second to disk in kilobytes.
- In the given figure we can see the process, memory, swap in memory, swap out memory, io, system and cpu update.

Adapted from:
"free Command in Linux with examples" by Kartik Thakral, Geeks for Geeks is licensed under CC BY-SA 4.0"
vmstat command in Linux with Examples" by Bhumika_Rani, Geeks for Geeks is licensed under CC BY-SA 4.0

08-E.10.2: CPU and Memory Troubleshooting - free / vmstat Commands is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

08-E.10.3: CPU and Memory Troubleshooting - OOM & swap

The Out of Memory Manager (OOM)

The OOM manager checks to see if there is enough available memory to verify that the system is truly out of memory and if so, selects a process to kill. This is a controversial part of the VM and it has been suggested that it be removed on many occasions. Regardless of whether it exists in the latest kernel, it still is a useful system to examine as it touches off a number of other subsystems.

Checking Available Memory

For certain operations, such as expanding the heap or remapping an address space, the system will check if there is enough available memory to satisfy a request.

Unless the system administrator has specified that the system should over-commit memory, the amount of available memory will be checked. To determine how many pages are potentially available, Linux sums up the following bits of data:

- Total page cache as page cache is easily reclaimed
- Total free pages because they are already available
- Total free swap pages as user space pages may be paged out
- Total pages managed by `swapper_space` although this double-counts the free swap pages. This is balanced by the fact that slots are sometimes reserved but not used
- Total pages used by the dentry cache as they are easily reclaimed
- Total pages used by the inode cache as they are easily reclaimed

If the total number of pages added here is sufficient for the request, `vm_enough_memory()` returns true to the caller. If false is returned, the caller knows that the memory is not available and usually decides to return `-ENOMEM` to userspace.

Determining OOM Status

When the machine is low on memory, old page frames will be reclaimed, but despite reclaiming pages it may find that it was unable to free enough pages to satisfy a request even when scanning at highest priority.

Unfortunately, it is possible that the system is not out memory and simply needs to wait for IO to complete or for pages to be swapped to backing storage. This is unfortunate not because the system has memory, but because the function is being called unnecessarily, opening the possibly of processes being unnecessarily killed. Before deciding to kill a process, it goes through the following checklist:

- Is there enough swap space left? If yes, not OOM
- Has it been more than 5 seconds since the last failure? If yes, not OOM
- Have we failed within the last second? If no, not OOM
- If there haven't been 10 failures at least in the last 5 seconds, we're not OOM
- Has a process been killed within the last 5 seconds? If yes, not OOM

Only after the above tests have passed is a process selected to kill.

To avoid OOM killing processes there are two steps that can be taken:

1. The fastest and easiest option is to upgrade the memory in the server.
2. Reduce memory usage to stop any processes running that are not needed. This would entail finding all the processes that are started at boot time, looking at what processes are run by cron, and what users are running.

Swap Space Configurations

What is swap?

Swap space is the area on a hard disk. It is a part of your machine's Virtual Memory, which is a combination of accessible physical memory (RAM) and the swap space. Swap holds memory pages that are temporarily inactive. Swap space is used when your operating system decides that it needs physical memory for active processes and the amount of available (unused) physical memory

is insufficient. When this happens, inactive pages from the physical memory are then moved into the swap space, freeing up that physical memory for other uses. Note that the access time for swap is slower, depending on the speed of the hard drive. Do not consider it to be a complete replacement for the physical memory. Swap space can be a dedicated swap partition (recommended), a swap file, or a combination of swap partitions and swap file(s).

Types of Linux Swap

Linux provides for two types of swap space. By default, most Linux installations create a swap partition, but it is also possible to use a specially configured file as a swap file. A swap partition is just what its name implies—a standard disk partition that is designated as swap space by the **`mkswap`** command.

A swap file can be used if there is no free disk space in which to create a new swap partition or space in a volume group where a logical volume can be created for swap space. This is just a regular file that is created and pre-allocated to a specified size. Then the **`mkswap`** command is run to configure it as swap space.

The `mkswap` Command

The **`mkswap`** sets up a Linux swap area on a device or in a file.

Syntax:

```
mkswap [ OPTIONS ] device [size]
```

Command Options:

Options	Option Meaning
-a, --all	All devices marked as "swap" in */etc/fstab* are made available, except for those with the "noauto" option. Devices that are already being used as swap are silently skipped.
-e, --ifexists	Silently skip devices that do not exist.
--show	Display a definable table of swap areas. See the --help output for a list of available columns.

If a system doesn't have swap space or if the swap space is not adequate enough, a swap file can be created on Linux. You can create multiple swap files as well.

```
pbmac@pbmac-server $ sudo fallocate -l 1G /swapfile
pbmac@pbmac-server $ sudo chmod 600 /swapfile
pbmac@pbmac-server $ sudo mkswap /swapfile
pbmac@pbmac-server $ sudo swapon /swapfile
pbmac@pbmac-server $ swapon --show
NAME       TYPE   SIZE USED PRIO
/swapfile  file 1024M   0B   -2
```

You can see in the above example the use of the **`swapon`** command. There are a couple of options available in swapon.

There is also a **`swapoff`** command. This command disable devices and files for paging and swapping. The most common option is -a which deactivates all known swap devices and files (as found in /proc/swaps or /etc/fstab).

Adapted from:
"Chapter 13 Out Of Memory Management" by Unknown, Geeks for Geeks is in the Public Domain, CC0
"SwapFaq" by Jose Bartolome, Ubuntu Community Help Wiki is licensed under CC BY-SA 4.0
"An introduction to swap space on Linux systems" by David Both, opensource.com is licensed under CC BY-SA 4.0

08-E.10.3: CPU and Memory Troubleshooting - OOM & swap is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 09: Managing Linux Devices

- 09-A.1: Supported Linux Devices
  - 09-A.1.1: Supported Linux Devices (cont'd)
  - 09-A.1.2: Supported Linux Devices (continued)
- 09-B.2: Configuring Linux Devices
  - 09-B.2.1: Configuring Linux Devices - Printing
- 09-C.3: Monitoring Devices in Linux
- 09-D.4: Troubleshooting: Hardware Issues

Module 09: Managing Linux Devices is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

09-A.1: Supported Linux Devices

EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

Objectives of this Module

In this lesson, you will:

- Identify the different types of devices that support the Linux OS.
- Learn to configure devices.
- Understand how to monitor devices.
- Troubleshoot various issues having to do with hardware devices.

Reminder: Device Drivers

Instead of putting code to manage the hardware controllers in the system into every application, the code is kept in the Linux kernel. The software that handles or manages a hardware controller is known as a device driver. The Linux kernel device drivers are, essentially, a shared library of privileged, memory resident, low level hardware handling routines. It is Linux's device drivers that handle the peculiarities of the devices they are managing.

One of the basic features is that it abstracts the handling of devices. All hardware devices look like regular files; they can be opened, closed, read and written using the same, standard, system calls that are used to manipulate files. Every device in the system is represented by a device special file; for example the first IDE disk in the system is represented by /dev/hda. For block (disk) and character devices, these device special files are created by the mknod command and they describe the device using major and minor device numbers. Network devices are also represented by device special files but they are created by Linux as it finds and initializes the network controllers in the system. All devices controlled by the same device driver have a common major device number. Device drivers are necessary, otherwise it would be impossible to use a device, or at least, to use it efficiently.

What is a Thin Client?

In computer networking, a thin client is a simple (low-performance) computer that has been optimized for establishing a remote connection with a server-based computing environment. The server does most of the work, which can include launching software programs, performing calculations, and storing data. This contrasts with a fat client or a conventional personal computer; the former is also intended for working in a client–server model but has significant local processing power, while the latter aims to perform its function mostly locally.[citation needed]

Thin clients occur as components of a broader computing infrastructure, where many clients share their computations with a server or server farm. The server-side infrastructure uses cloud computing software such as application virtualization, hosted shared desktop (HSD) or desktop virtualization (VDI). This combination forms what is known as a cloud-based system, where desktop resources are centralized at one or more data centers. The benefits of centralization are hardware resource optimization, reduced software maintenance, and improved security.

- Example of hardware resource optimization: Cabling, bussing and I/O can be minimized while idle memory and processing power can be applied to user sessions that most need it.
- Example of reduced software maintenance: Software patching and operating system (OS) migrations can be applied, tested and activated for all users in one instance to accelerate roll-out and improve administrative efficiency.
- Example of improved security: Software assets are centralized and easily fire-walled, monitored and protected. Sensitive data is uncompromised in cases of desktop loss or theft.

Thin client hardware generally supports common peripherals, such as keyboards, mice, monitors, jacks for sound peripherals, and open ports for USB devices (e.g., printer, flash drive, webcam). Some thin clients include (legacy) serial or parallel ports to support older devices, such as receipt printers, scales or time clocks. Thin client software typically consists of a graphical user interface (GUI), cloud access agents, a local web browser, terminal emulators (in some cases), and a basic set of local utilities.

Devices: USB

Universal Serial Bus (USB) is an industry standard that establishes specifications for cables and connectors and protocols for connection, communication and power supply (interfacing) between computers, peripherals and other computers. A broad variety of USB hardware exists, including several different connectors, of which USB-C is the most recent.

Released in 1996, the USB standard is currently maintained by the USB Implementers Forum (USB-IF). There have been four generations of USB specifications: USB 1.x, USB 2.0, USB 3.x and USB4.

USB was designed to standardize the connection of peripherals to personal computers, both to communicate with and to supply electric power. It has largely replaced interfaces such as serial ports and parallel ports, and has become commonplace on a wide range of devices. Examples of peripherals that are connected via USB include computer keyboards and mice, video cameras, printers, portable media players, disk drives and network adapters.

USB connectors have been increasingly replacing other types as charging cables of portable devices. Pretty much every peripheral device uses the USB connectors now.

Devices: Wireless

LPWAN provides long-range communication on small, inexpensive batteries that last for years. This technology supports large-scale IoT networks sprawling over vast industrial and commercial campuses. LPWANs can connect a variety of sensors which facilitates numerous applications from remote monitoring, smart metering and worker safety to building controls and facility management. Nevertheless, LPWANs can only send small blocks of data at a low rate, and therefore are better suited for use in cases that don’t require high bandwidth and are not time-sensitive.

Cellular technology is well established in the world. It offers reliable broadband communication supporting a variety of applications. On the downside, they impose very high operational costs and power requirements.

Zigbee is a short-range, low-power, wireless standard (IEEE 802.15.4), commonly deployed in mesh topology to extend coverage by relaying sensor data over multiple sensor nodes. Because of their physical short-range (< 100m), Zigbee and similar mesh protocols are best-suited for medium-range applications with an even distribution of nodes in close proximity.

Bluetooth and Bluetooth Low Energy (BLE) are wireless technologies used to transfer data over short distances. The technology is frequently used in small consider devices that connect to users' phones and tablets. For instance, the technology is used in many speaker systems. Bluetooth Low Energy uses less power than standard Bluetooth and is used in hardware such as fitness trackers, smart watches and other connected devices in order to wirelessly transmit data without heavily compromising the battery power in a user’s phone.

WiMax stands for Worldwide Interoperability for Microwave Access. This wireless technology allows data to be transferred at a rate of 30-40 megabits per second. The term refers specifically to interoperable implementations of the IEEE 802.16 wireless family. The technology was once used by several mobile carriers, notably Sprint, to deliver wireless data to its customers. Sprint, along with many of the other carriers who used the technology, has since switched over to using faster LTE 4G networks for data.

WiFi uses radio waves (RF) to allow two devices to communicate with one another. The technology is most commonly used to connect Internet routers to devices like computers, tablets and phones; however, it can be used to connect together any two hardware components. WiFi is a local wireless network that runs of the 802.11 standards set forth by the Institute of Electrical and Electronics Engineers (IEEE).

Adapted from:
"Chapter 8 Device Drivers" by Multiple Contributors, The Linux Documentation Project is in the Public Domain, CC0
"Thin client" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"USB" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

09-A.1: Supported Linux Devices is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

09-A.1.1: Supported Linux Devices (cont'd)

Audio and Video Devices

Audio connectors and video connectors are electrical or optical connectors for carrying audio and video signals. Audio interfaces and video interfaces define physical parameters and interpretation of signals. For digital audio and digital video, this can be thought of as defining the physical layer, data link layer, and most or all of the application layer. For analog audio and analog video these functions are all represented in a single signal specification like NTSC or the direct speaker-driving signal of analog audio. Physical characteristics of the electrical or optical equipment includes the types and numbers of wires required, voltages, frequencies, optical intensity, and the physical design of the connectors. Any data link layer details define how application data is encapsulated (for example for synchronization or error-correction). Application layer details define the actual audio or video format being transmitted, often incorporating a codecs not specific to the interface, such as PCM, MPEG-2, or the DTS Coherent Acoustics codec. In some cases, the application layer is left open; for example, HDMI contains an Ethernet channel for general data transmission.

Some types of connectors are used by multiple hardware interfaces; for example, RCA connectors are defined both by the composite video and component video interfaces, but DVI is the only interface that uses the DVI connector. This means that in some cases not all components with physically compatible connectors will actually work together.

Some of these connectors, and other types of connectors, are also used at radio frequency (RF) to connect a radio or television receiver to an antenna or to a cable system; RF connector applications are not further described here. Analog A/V connectors often use shielded cables to inhibit radio frequency interference (RFI) and noise.

For efficiency and simplicity, the same codec or signal convention is used by the storage medium. For example, VHS tapes can store a magnetic representation of an NTSC signal, and the specification for Blu-ray Discs incorporates PCM, MPEG-2, and DTS. Some playback devices can re-encode audio or video so that the format used for storage does not have to be the same as the format transmitted over the A/V interface (which is helpful if a projector or monitor cannot handle a newer codec).

Printers

Linux supports numerous printer interfaces. Over the life of Linux some interfaces that used to be popular for connecting printers have disappeared.

Parallel
A parallel printer interface works by sending an 8-bit information stream to the printer. The parallel printer cable has a DB-25 connector to connect to the computer and a 36-pin Centronics connector to the printer interface. Data integrity decreases if the cable gets too long, therefore the maximum length of the parallel cable is usually limited to 10 feet.

Serial
A serial printer interface sends the data to the printer one bit at a time. Serial printers are rarely used these days, however some dot matrix printers that are still found in certain applications use this interface.

Small Computer System Interface (SCSI)
There are not very many printers which use a SCSI interface. These printers are pretty much obsolete due to newer, faster printer interfaces. Most of the current printers being manufactured make use of the much higher speed of newer interfaces.

Universal Serial Bus (USB)
The USB port is the most common port available on PCs today. It is an excellent interface and is used on printers and desktop printers/scanners combos. USB is much faster than most of the other types of printer interfaces.

Network
Today most printers come with a built-in network adapter or can also have one fitted into them. Network printers are directly attached to the local network. The printer uses a standard network cable with a standard RJ-45 connector.

Wireless
Many printers nowadays provide the capability to use wireless connections that support 802.11, or Bluetooth. The main advantage of wireless connections is that there are no wires. The disadvantage is that there is a limit to the range of the network coverage.

Network Interface

A network interface controller (NIC, also known as a network interface card or network adapter) is a computer hardware component that connects a computer to a computer network. Most computers have at least one NIC, although it is entirely possible for Linux systems to have multiple network adapters; this is especially true of specialized Linux servers, such as firewalls.

Network controllers were originally implemented as expansion cards that plugged into a computer bus. The low cost and ubiquity of the Ethernet standard means that most new computers have a network interface controller built into the motherboard. Newer server motherboards may have multiple network interfaces built-in. The Ethernet capabilities are either integrated into the motherboard chipset or implemented via a low-cost dedicated Ethernet chip. A separate network card is typically no longer required unless additional independent network connections are needed or some non-Ethernet type of network is used. A general trend in computer hardware is towards integrating the various components of systems on a chip, and this is also applied to network interface cards.

GPIO

Stands for "General Purpose Input/Output." Some GPIOs have a primary function whereas other GPIOs are a convenient "accessory" to some other primary function.

GPIOs are used in a diverse variety of applications, limited only by the electrical and timing specifications of the GPIO interface and the ability of software to interact with GPIOs in a sufficiently timely manner.

GPIOs typically employ standard logic levels and cannot supply significant current to output loads. When followed by an appropriate high-current output buffer (or mechanical or solid-state relay), a GPIO may be used to control high-power devices such as lights, solenoids, heaters, and motors (e.g., fans and blowers). Similarly, an input buffer, relay or opto-isolator is often used to translate an otherwise incompatible signal (e.g., high voltage) to the logic levels required by a GPIO.

Adapted from:
"Audio and video interfaces and connectors" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0"
Network interface controller" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"General-purpose input/output" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

09-A.1.1: Supported Linux Devices (cont'd) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

09-A.1.2: Supported Linux Devices (continued)

Hardware Interfaces

SATA

Serial ATA (SATA, abbreviated from Serial AT Attachment) is a computer bus interface that connects host bus adapters to mass storage devices such as hard disk drives, optical drives, and solid-state drives.

SATA host adapters and devices communicate via a high-speed serial cable over two pairs of conductors. In contrast, parallel ATA (the redesignation for the legacy ATA specifications) uses a 16-bit wide data bus with many additional support and control signals, all operating at a much lower frequency. To ensure backward compatibility with legacy ATA software and applications, SATA uses the same basic ATA and ATAPI command sets as legacy ATA devices. SATA has replaced parallel ATA in consumer desktop and laptop computers.

SATA continues to advance and has been combined, over time, with other interfaces to assist in higher throughput capacities.

SCSI

Small Computer System Interface (SCSI) is a set of standards for physically connecting and transferring data between computers and peripheral devices.

SCSI is derived from "SASI", the "Shugart Associates System Interface", developed circa 1978 and publicly disclosed in 1981. A SASI controller provided a bridge between a hard disk drive's low-level interface and a host computer, which needed to read blocks of data.

SCSI is available in a variety of interfaces. The first was parallel SCSI (also called SCSI Parallel Interface or SPI), which uses a parallel bus design. Since 2005, SPI was gradually replaced by Serial Attached SCSI (SAS), which uses a serial design but retains other aspects of the technology. Many other interfaces which do not rely on complete SCSI standards still implement the SCSI command protocol; others drop physical implementation entirely while retaining the SCSI architectural model. iSCSI, for example, uses TCP/IP as a transport mechanism, which is most often transported over Gigabit Ethernet or faster network links.

HBA

In computer hardware, a host controller, host adapter, or host bus adapter (HBA) connects a computer, which acts as the host system, to other network and storage devices. The terms are primarily used to refer to devices for connecting SCSI, Fibre Channel and SATA devices. Devices for connecting to IDE, Ethernet, FireWire, USB and other systems may also be called host adapters.

Host adapters can be integrated in the motherboard or be on a separate expansion card.

PCI

Peripheral Component Interconnect (PCI) is a local computer bus for attaching hardware devices in a computer and is part of the PCI Local Bus standard. The PCI bus supports the functions found on a processor bus but in a standardized format that is independent of any particular processor's native bus.

The PCI specification has more recently been superseded by the PCI Express (PCIe).

PCI Express (Peripheral Component Interconnect Express), officially abbreviated as PCIe or PCI-e is a high-speed serial computer expansion bus standard, designed to replace the older PCI, PCI-X and AGP bus standards. It is the common motherboard interface for personal computers' graphics cards, hard drives, SSDs, WiFi and Ethernet hardware connections. PCIe has numerous improvements over the older standards, including higher maximum system bus throughput, lower I/O pin count and smaller physical footprint, better performance scaling for bus devices, a more detailed error detection and reporting mechanism (Advanced Error Reporting, AER), and native hot-swap functionality. More recent revisions of the PCIe standard provide hardware support for I/O virtualization.

Adapted from:
"Serial ATA" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"SCSI" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Host adapter" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Peripheral Component Interconnect" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"PCI Express" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

09-A.1.2: Supported Linux Devices (continued) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

09-B.2: Configuring Linux Devices

EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

Device Information Files

There are numerous files spread across the Linux filesystem that are useful for managing the various devices that are connected to or used by the system.

/**proc** is a special filesystem in Linux that presents information about processes and other system information in a hierarchical file-like structure, providing a more convenient and standardized method for dynamically accessing process data held in the kernel than traditional tracing methods or direct access to kernel memory. Typically, it is mapped to a mount point named */proc* at boot time. The proc file system acts as an interface to internal data structures in the kernel. It can be used to obtain information about the system and to change certain kernel parameters at runtime (sysctl).

/sys is a pseudo file system provided by the Linux kernel that exports information about various kernel subsystems, hardware devices, and associated device drivers from the kernel's device model to user space through virtual files. In addition to providing information about various devices and kernel subsystems, exported virtual files are also used for their configuration.

/dev is a device file or special file in an interface to a device driver that appears in a file system as if it were an ordinary file. These special files allow an application program to interact with a device by using its device driver via standard input/output system calls. Using standard system calls simplifies many programming tasks, and leads to consistent user-space I/O mechanisms regardless of device features and functions.

/**etc** contains all system related configuration files or its sub-directories. A "configuration file" is defined as a local file used to control the operation of a program; it must be static and cannot be an executable binary.

udev

Udev is the Linux subsystem that supplies your computer with device events. In plain English, that means it's the code that detects when you have things plugged into your computer, like a network card, external hard drives (including USB thumb drives), mouses, keyboards, joysticks and gamepads, DVD-ROM drives, and so on. That makes it a potentially useful utility, and it's well-enough exposed that a standard user can manually script it to do things like performing certain tasks when a certain hard drive is plugged in.

udev rules written by the administrator go in /etc/udev/rules.d/; their file name has to end with .rules. The udev rules shipped with various packages are found in /usr/lib/udev/rules.d/. The files in which the rules are defined are conventionally named with a number as prefix (e.g 50-udev-default.rules) and are processed in lexical order independently of the directory they are in. Files installed in /etc/udev/rules.d, however, override those with the same name installed in the system default path.

The `udevadm` Command

The **`udevadm`** command manages udev. There are various options, arguments and subcommands.

Syntax:

```
udevadm [ OPTIONS ] [subcommand] [arguments]
```

Some of the most frequently used subcommands are shown below.

Sub-command	Meaning
control	Modify the internal state of the running udev daemon<
info	Query the udev database for device information.
monitor	Listens to the kernel uevents and events sent out by a udev rule and prints the devpath of the event to the console. It can be used to analyze the event timing by comparing the timestamps of the kernel uevent and the udev event.
test	Simulate a udev event run for the given device, and print debug output.
trigger	Request device events from the kernel. Primarily used to replay events at system coldplug time.

Here is an example of a rule that creates a symlink when a webcam is connected. If this camera is currently connected and has been loaded with the device name /dev/video2, then we can run the udevadm command with some options and we can find out a lot about the webcam.

```
pbmac@pbmac-server $ udevadm info --attribute-walk --path=$(udevadm info --query=path

Udevadm info starts with the device specified by the devpath and then walks up the cha
It prints for every device found, all possible attributes in the udev rules key format
A rule to match, can be composed by the attributes of the device and the attributes fr

looking at device '/devices/pci0000:00/0000:00:04.1/usb3/3-2/3-2:1.0/video4linux/video
  KERNEL=="video2"
  SUBSYSTEM=="video4linux"
    ...
looking at parent device '/devices/pci0000:00/0000:00:04.1/usb3/3-2/3-2:1.0':
  KERNELS=="3-2:1.0"
  SUBSYSTEMS=="usb"
  ...
looking at parent device '/devices/pci0000:00/0000:00:04.1/usb3/3-2':
  KERNELS=="3-2"
  SUBSYSTEMS=="usb"
  ATTRS{idVendor}=="05a9"
  ATTRS{manufacturer}=="OmniVision Technologies, Inc."
  ATTRS{removable}=="unknown"
  ATTRS{idProduct}=="4519"
  ATTRS{bDeviceClass}=="00"
  ATTRS{product}=="USB Camera"
```

We can gather some of the information here: SUBSYSTEM, SUBSYSTEMS, and ATTRS (attributes),

Then we can create a file, /etc/udev/rules.d/83-webcam.rules, that creates a symbolic link when this device is plugged in. In this fashion we can ALWAYS refer to this device as video-cam, no matter if it gets mounted on /dev/video0, or /dev/video1, or wherever.

```
KERNEL=="video[0-9]*", SUBSYSTEM=="video4linux", SUBSYSTEMS=="usb", ATTRS{idVendor}=='
SYMLINK+="video-cam"
```

Adapted from:
"procfs" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"sysfs" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Device file" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"1.6. /etc" by Multiple Contributors, The Linux Documentation Project is in the Public Domain, CC0
"udev" by Multiple Contributors, Arch Linux is licensed under CC BY-SA 3.0

09-B.2: Configuring Linux Devices is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

09-B.2.1: Configuring Linux Devices - Printing

CUPS

CUPS (formerly an acronym for Common UNIX Printing System) is a modular printing system for Unix-like computer operating systems which allows a computer to act as a print server. A computer running CUPS is a host that can accept print jobs from client computers, process them, and send them to the appropriate printer.

CUPS consists of a print spooler and scheduler, a filter system that converts the print data to a format that the printer will understand, and a backend system that sends this data to the print device. CUPS uses the Internet Printing Protocol (IPP) as the basis for managing print jobs and queues.

There is a web based interface as shown in the following image:

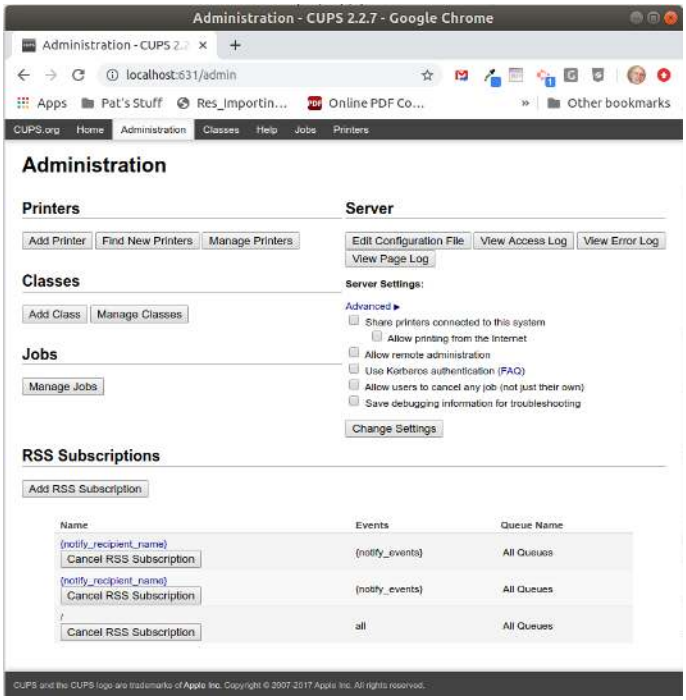


Figure **9.2.1.1**: CUPS Administrative Interface ("Cups Interface" by Patrick McClanahan is licensed under CC BY-SA 4.0)

This interface, as shown in the URL bar, is at localhost:631/admin. It is possible to configure printers from this interface.These changes are saved in /etc/cups/supsd.conf and /etc/cups/cups-files.conf.

In order to perform administrative tasks via the web interface, a user must either have the root account enabled on the server, or authenticate as a user in the lpadmin group. For security reasons, CUPS won't authenticate a user that doesn't have a password.

The use of CUPS greatly simplifies the installation and configuration of printers on a Linux system.

The `lpr` Command

The **`lpr`** command submits files for printing. Files named in a shell session route to the named printer or to the system default destination if no destination is specified. If no files are listed, lpr reads the print file from the standard **input**.

Syntax:

```
lpr [ OPTIONS ] [FILENAMES]
```

Command Options:

Options	Options Meaning
E	Forces encryption when connecting to the server.
-H *server*[:*port*]	Specifies an alternate server.
-P [*destination*[/*instance*]	Prints files to the named printer.
-U *username*	Specifies an alternate username.
-# *copies*	Sets the number of copies to print from **1** to **100**.
-h	Disables banner printing. This option is equivalent to "**-o job-sheets=none**".
-l	Specifies that the print file is already formatted for the destination and should be sent without filtering. This option is equivalent to "**-o raw**".
-m	Sends an email when a job is complete.
-o *option*[=*value*]	Sets a job option.
-p	Specifies that the print file should be formatted with a shaded header with the date, time, job name, and page number. This option is equivalent to "**-o prettyprint**" and is only useful when printing text files.
-q	Hold job for printing.
-r	Specifies that the named print files should be deleted after printing them.

09-B.2.1: Configuring Linux Devices - Printing is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

09-C.3: Monitoring Devices in Linux

EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

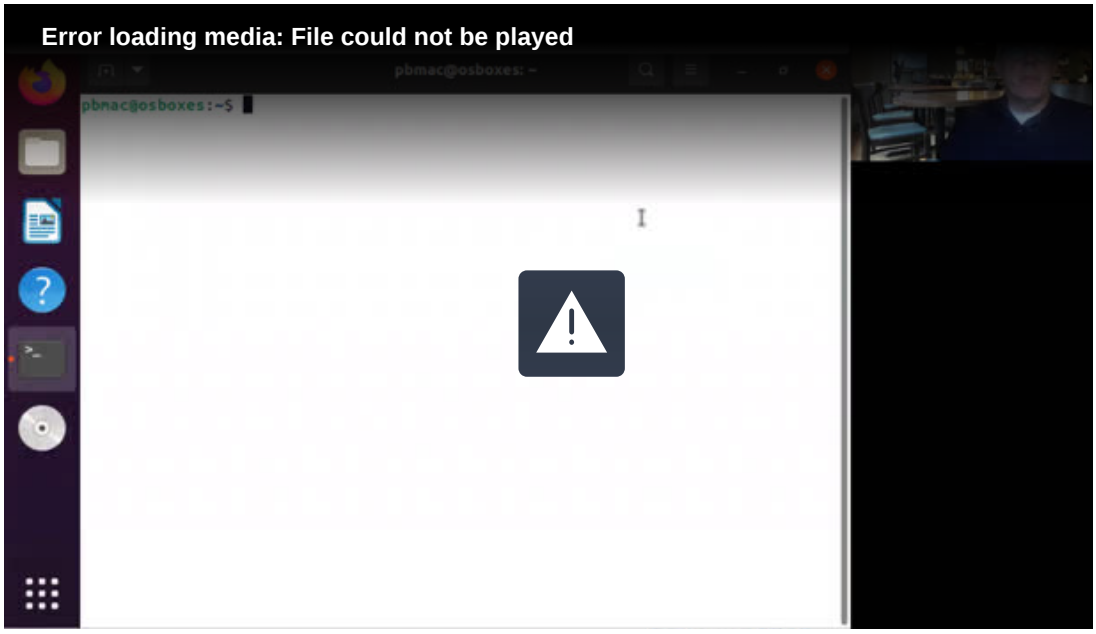
The `lsdev` Command

`lsdev` gathers information about your computer's installed hardware from the /proc/interrupts, /proc/ioports and /proc/dma files, giving you a quick overview of which hardware uses what I/O addresses and what IRQ and DMA channels.

The information used by the lsdev command is found in the the following files:

- /proc/interrupts — Each logical CPU core and its associated interrupt requests (IRQ) is listed in this file. A device will send an interrupt to the processor to inform the processor that the hardware needs to perform some task. The file lists how many signals were sent to each CPU core through one of the many IRQ addresses in the system. The name of the hardware device that is mapped to each IRQ address is also listed.
- /proc/ioports —This file provides a list of currently registered port regions used for input or output communication with a device.
- /proc/dma —This file contains a list of the registered ISA DMA channels in use.

This video covers the use of some of the 'ls' type of commands.



The `lsusb` Command

The **`lsusb`** command in Linux is used to display the information about USB buses and the devices connected to them. The properties displayed are speed, BUS, class, type details, etc. This information is gathered by scanning the /dev/bus/usb for information about the connected device, the device id, and vendor information.

Syntax:

```
lsusb [ OPTIONS ] - probably the most used option is -V which provides a very verbose output for
```

Command Options:

Options	Meaning
-v, --verbose	Tells *lsusb* to be verbose and display detailed information about the devices shown. This includes configuration descriptors for the device's current speed. Class descriptors will be shown, when available, for USB device classes including hub, audio, HID, communications, and chipcard
-s [[*bus*]:][*devnum*]	Show only devices in specified *bus* and/or *devnum*. Both ID's are given in decimal and may be omitted.
-d [*vendor*]:[*product*]	Show only devices with the specified vendor and product ID. Both ID's are given in hexadecimal
-D device	Do not scan the /dev/bus/usb directory, instead display only information about the device whose device file is given. The device file should be something like /dev/bus/usb/001/001
-t	Tells lsusb to dump the physical USB device hierarchy as a tree. This overrides the v option.
-V, --version	Print version information on standard output, then exit successfully.

Example:

```
pbmac@pbmac-server $ lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 004: ID 046d:c52b Logitech, Inc. Unifying Receiver
Bus 004 Device 003: ID 413c:2006 Dell Computer Corp.
Bus 004 Device 002: ID 413c:1004 Dell Computer Corp.
|------|              |--------| |------------------|
Bus number            Device ID   Vendor and product info
```

The `lspci` Command

The **`lspci`** command on Linux systems is a utility for displaying information about PCI buses in the system and devices connected to them. By default, it shows a brief list of devices. Use the options described below to request either a more verbose output or output intended for parsing by other programs.

Syntax:

```
lspci [ OPTIONS ]
```

Command Options:

Option	Meaning
-m	Dump PCI device data in a backward-compatible machine readable form. See below for details.
-mm	Dump PCI device data in a machine readable form for easy parsing by scripts. See the man page for details.
-t	Show a tree-like diagram containing all buses, bridges, devices and connections between them.

Example:

```
pbmac@pbmac-server $ lspci
00:00.0 Host bridge: Intel Corporation 4 Series Chipset DRAM Controller (rev 03)
00:01.0 PCI bridge: Intel Corporation 4 Series Chipset PCI Express Root Port (rev 03)
00:19.0 Ethernet controller: Intel Corporation 82567LM-3 Gigabit Network Connection (rev 02)
00:1d.7 USB controller: Intel Corporation 82801JD/DO (ICH10 Family) USB2 EHCI Controller #1 (rev
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev a2)
00:1f.2 RAID bus controller: Intel Corporation SATA Controller [RAID mode] (rev 02)
00:1f.3 SMBus: Intel Corporation 82801JD/DO (ICH10 Family) SMBus Controller (rev 02)
01:00.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] RV620 LE [Radeon HD 345
```

The lpq Command

The **lpq** command shows the current print queue status on the named printer. Jobs queued on the default destination will be shown if no printer or class is specified on the command-line. The +interval option allows you to continuously report the jobs in the queue until the queue is empty; the list of jobs is shown once every *interval* seconds.

Syntax:

```
lpq [ OPTIONS ]
```

Command Options:

Options	Meaning
-E	Forces encryption when connecting to the server.
-P destination[/instance]	Specifies an alternate printer or class name.
-U username	Specifies an alternate username.
-a	Reports jobs on all printers.
-h server[:port]	Specifies an alternate server.
-l	Requests a more verbose (long) reporting format.

Example:

```
pbmac@pbmac-server $ lpq
printer is ready and printing
Rank    Owner   Job     File(s)                         Total Size
active  patm    603     (stdin)
```

Adapted from:
"lspci" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"lsusb command in Linux with Examples" by basilmohamed, Geeks for Geeks is licensed under CC BY-SA 4.0

09-C.3: Monitoring Devices in Linux is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

09-D.4: Troubleshooting: Hardware Issues

EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

Troubleshooting Hardware Issues

Linux servers run mission-critical business applications in many different types of infrastructures including physical machines, virtualization, private cloud, public cloud, and hybrid cloud. It's important for Linux sysadmins to understand how to manage Linux hardware infrastructure, including software-defined functionalities related to networking, storage, Linux containers, and multiple tools on Linux servers.

It can take some time to troubleshoot and solve hardware-related issues on Linux. Even highly experienced sysadmins sometimes spend hours working to solve mysterious hardware and software discrepancies.

There are a lot of issues that can appear, seemingly out of nowhere. It is often due to incompatibilities or incorrect configurations of:

- WiFi drivers
- Printer drivers
- Video issues
- Audio problems

WiFi drivers (especially Broadcom chips)

Generally speaking, WiFi drivers—and Broadcom cards in particular—continue to be one of the most problematic technical issues facing Linux. There were hundreds of posts about this topic on LQ alone in 2016, and myriad more elsewhere. Dozens of Broadcom wireless cards are available, and detailed instructions for getting them to work with each distribution is far too involved for a single article, but the basic troubleshooting steps are the same:

- ascertain exactly which Broadcom card you have by using lspci to find out the PCI ID
- determine whether the distribution you use supports that card
- and if it does, identify the proper way to get the card working

For example, if you have a 14e4:4315 PCI ID and are using Ubuntu, then you know the BCM4312 card is supported by installing the firmware-b43-installer package. The other option you have is to research the WiFi card before your purchase to ensure it's fully supported by your distribution of choice out of the box.

Printer drivers (especially Canon and Lexmark)

Printers also continue to be problematic, with Canon and Lexmark repeatedly cited for being an issue. If you're purchasing a new printer, research compatibility before you buy. But if you are migrating from another operating system, that may not be an option. If you are doing research, the OpenPrinting database and the official support channel for your distribution are the two best places to start. Note that you should ensure all functionality of a device is fully compatible, especially if it's a multifunction product. One common complaint with Canon printers is that the drivers are often only available on non-English and sometimes obscure sites.

Memory

Memory issues can cause a variety of problems in a computer. Looking from a software perspective, if a process fails to free up allocated memory when it is no longer needed, this is called a memory leak. The available memory on the system can be used up. This can lead to poor performance and can cause system instability since other software is not able to access the memory it needs to execute properly. The use of memory monitoring tools like **`free`** as well as system process monitoring tools such as **`top`** can help identify the problem.

Some memory issues are caused by a fault in the physical RAM modules or the motherboard. Linux has ways of detecting these faults. If a system log file records a "Machine Check Exception" error message this normally indicates an issue with RAM. The

mcelog command can retrieve and print these error messages for easier analysis. If the messages contain error-correcting code (ECC) errors, one of the memory modules has probably failed.

Video

Video is a nuanced topic, as simple straightforward video works extremely well out of the box on Linux. Where the issues pop up are video accelerators/acceleration; the latest video cards and newest technologies, such as NVIDIA Optimus and ATI dynamic GPU switching; installation and stability of proprietary drivers; efficient power management; and reliable suspend and resume. If you're not a gamer, do not need high-end graphics for another reason, and are not on a laptop, then you probably don't have to worry about this. If you're looking for a new laptop, be sure to research compatibility before your purchase.

Audio

Once again, for simple setups, audio has been easy to configure and reliable under Linux for a while. As soon as you get into professional production, echo cancelation, audio routing, unified mixing, and other complex setups, however, it can go south pretty quickly. My suggestion is to use one of the dedicated audio-related distributions if you need high-end real-time audio.

The **lshw** Command

The **lshw** command stands for “list hardware.” It's a Linux tool which is used to generate the detailed information of the system’s hardware configuration from various files in the /proc directory. **lshw** can also report exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache memory configuration, bus speed, etc on DMI-capable x86 or IA-64 (Itanium family of 64 microprocessors) system and some PowerPC machine. This command needs root permission to show full information or else partial information will be displayed.

Syntax:

```
lshw [-format] [ OPTIONS ]
```

Command Options:

Options	Options Meaning
-html	Outputs the device tree as an HTML page.
-xml	Outputs the device tree as an XML tree.
-short	Outputs the device tree showing hardware paths, very much like the output of HP-UX's ioscan.
-businfo	Outputs the device list showing bus information, detailing SCSI, USB, IDE and PCI addresses.
-class class	Only show the given class of hardware. Class can be found using lshw -short or lshw -businfo.
-C class	Alias for -class class.

The example shown below is also showing the concept of device classes. In this instance the input calls, as shown by the -c option. The second USB device shows that it is a keyboard, but it is actually the USB wireless receiver for a mouse. The second **lshw** command shows the short listing of the disk class; -c and -class options are synonymous.

```
pbmac@pbmac-server $ sudo lshw -c input
  *-usb
       description: Keyboard
       product: Dell USB Keyboard Hub
       vendor: Dell
```

```
       physical id: 1
       bus info: usb@4:1.1
       version: 48.00
       capabilities: usb-1.10
       configuration: driver=usbhid maxpower=90mA speed=2Mbit/s
  *-usb:1
       description: Keyboard
       product: USB Receiver
       vendor: Logitech
       physical id: 2
       bus info: usb@4:2
       version: 12.11
       capabilities: usb-2.00
       configuration: driver=usbhid maxpower=98mA speed=12Mbit/s
pbmac@pbmac-server $ sudo lshw -short -class disk
H/W path           Device      Class       Description
=====================================================
/0/1/0.0.0         /dev/sda    disk        500GB ST3500630AS
/0/2/0.0.0         /dev/cdrom  disk        DVD+-RW TS-H653H
```

The **dmidecode** Command

The **dmidecode** command (Desktop Management Interface table decoder) records data from DMI table and produces human readable output. It is useful to retrieve the system's hardware related information such as processor, RAM (DIMMs), BIOS details, memory, serial numbers etc.

Syntax:

```
dmidecode [ OPTIONS ]
```

Command Options:

Options	Option Meaning
-d, --dev-mem FILE	Read memory from device FILE (default: /dev/mem)
-q, --quiet	Be less verbose. Unknown, inactive and OEM -specific entries are not displayed. Meta-data and handle references are hidden.
-s, --string KEYWORD	Only display the value of the DMI string identified by KEYWORD. KEYWORD must be a keyword from a known list (see the man page for the list).
-t, --type TYPE	Only display the entries of type TYPE. TYPE can be either a DMI type number, a comma-separated list of type numbers, or a keyword from the following list: bios, system, baseboard, chassis, processor, memory, cache, connector, slot.

```
pbmac@pbmac-server $ sudo dmidecode -t processor
# dmidecode 3.1
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
```

```
Handle 0x0400, DMI type 4, 40 bytes
Processor Information
    Socket Designation: CPU
    Type: Central Processor
    Family: Core 2 Duo
    Manufacturer: Intel
    ID: 7A 06 01 00 FF FB EB BF
    Signature: Type 0, Family 6, Model 23, Stepping 10
```

The ABRT Utility

ABRT is the Automatic Bug Reporting Tool. ABRT consists of a daemon, abrtd, which runs silently in the background most of the time. It springs into action when an application crashes, or a kernel oops is detected. The daemon then collects the relevant problem data such as a core file if there is one, the crashing application's command line parameters, and other data of forensic utility.

ABRT is capable of reporting problems to a remote issue tracker. Reporting can be configured to happen automatically whenever an issue is detected, or problem data can be stored locally, reviewed, reported, and deleted manually by a user. The reporting tools can send problem data to a Bugzilla database, a Red Hat Technical Support (RHTSupport) site, upload it using FTP/SCP, email it, or write it to a file.

The part of ABRT which handles already-existing problem data (as opposed to, for example, creation of new problem data) has been factored out into a separate project, libreport. The libreport library provides a generic mechanism for analyzing and reporting problems, and it is used by applications other than ABRT. However, ABRT and libreport operation and configuration is closely integrated. They are therefore discussed as one in this document.

The ABRT packages provide the following crucial components, among others:

- abrtd — The ABRT daemon which runs under root as a background service.
- abrt-applet — The program that receives messages from abrtd and informs you whenever a new problem occurs.
- abrt-gui — The GUI application that shows collected problem data and allows you to further process it.
- abrt-cli — The command line interface, which provides similar functionality to the GUI.
- abrt-ccpp — The ABRT service that provides the C/C++ problems analyzer.
- abrt-oops — The ABRT service that provides the kernel oopses analyzer.

Adapted from:
"Troubleshooting tips for the 5 most common Linux issues" by Jeremy Garcia, OpenSource.com is licensed under CC BY-SA 4.0
"dmidecode command in Linux with Examples" by Suraj1994, Geeks for Geeks is licensed under CC BY-SA 4.0
"Chapter 20. Automatic Bug Reporting Tool (ABRT)" by Multiple Contributors, Fedora Project & Community is licensed under CC BY-SA 3.0

09-D.4: Troubleshooting: Hardware Issues is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 10: Network Management

- 10-A.1: TCP/IP Fundamentals
  - 10-A.1.1: TCP/IP Fundamentals - Identities
  - 10-A.1.2: TCP/IP Fundamentals - Addressing
  - 10-A.1.3: TCP/IP Fundamentals - Ports and Segments
- 10-B.2: Server Roles in Linux
  - 10-B.2-3: Server Roles in Linux (continued)
  - 10-B.2.1: Server Roles in Linux - Continued
  - 10-B.2.4: Server Roles in Linux (continued)
- 10-C.3: Network Connection
- 10-D.4: Static VS Dynamic IP Addresses
- 10-D.5: DHCP Configuration
- 10-D.6: Name Resolution
- 10-E.7: Cloud Configuration
- 10-E.8: The Cloud and Virtualization
- 10-E.9: Cloud Bootstrapping and Storage
- 10-E.10: Configuring Networks and Virtualization Tools
- 10-F.11: How to Troubleshoot Networking Problems
  - 10-F.11.1: Network Troubleshooting Commands - netstat / ss
  - 10-F.11.2: Network Troubleshooting Commands - dig / nslookup / host
  - 10-F.11.3: Network Troubleshooting Commands - ip / route / nmap
  - 10-F.11.4: Network Troubleshooting Commands - wireshark
  - 10-F.11.5: Network Troubleshooting Commands - iftop / iperf / mtr
  - 10-F.11.6: Network Troubleshooting Commands - arp / whois

---

This page titled Module 10: Network Management is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

10-A.1: TCP/IP Fundamentals

Objectives of this Module

In this lesson, you will:

- Describe the fundamental concepts of TCP/IP.
- Determine the various roles that Linux servers can play.
- Learn how to connect to a network.
- Learn how to configure DNS and DHCP client services.
- Learn about configuring cloud and virtualization technologies.
- Troubleshoot networking and connection issues.

Linux uses networks A LOT. Therefore, we have to have some understanding of networks and how they work. Then we can build Linux specific networking skills.

What is a Network Protocol?

Networks use protocols as a set of rules for formatting and processing data. Network protocols provide a common language so computers can communicate with one another. Computers within any network may use totally different software and hardware, however, protocols enable them all to communicate with each other at any time.

When standardized protocols are used, it's like a common language that computers can use. It is just like two people from different parts of the world trying to communicate - they may not understand each other's native languages, but they can communicate by using a shared third language. If one computer uses the Internet Protocol (IP) and a second computer also uses IP, they will be able to communicate. But if one computer uses IP and the other uses some other protocol, they will be unable to communicate.

On the Internet, there are different protocols for different types of processes. Protocols are often discussed in terms of which OSI model layer they belong to.

The OSI Model

The Open Systems Interconnection model (OSI model) is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard communication protocols.

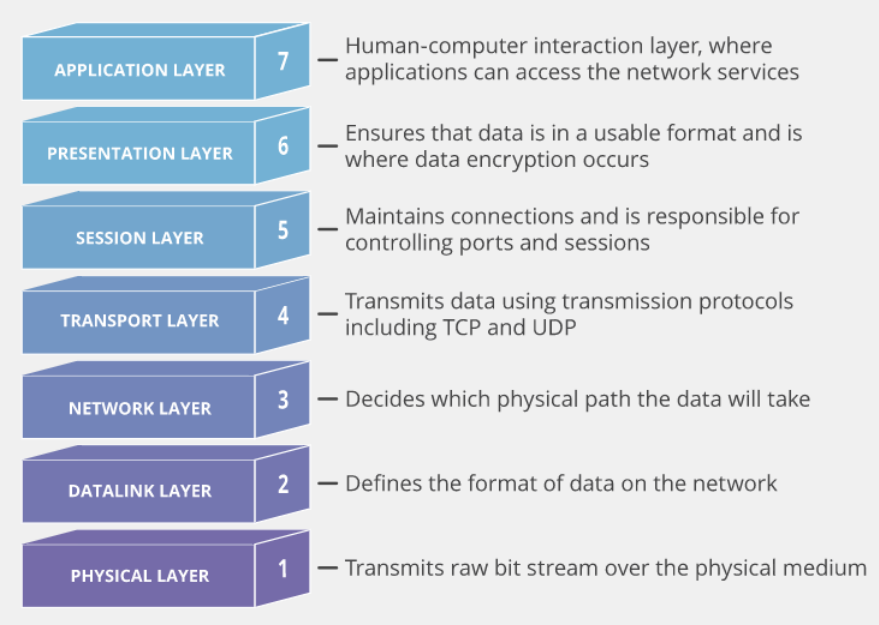


Figure **10.1.1**: The OSI Model ("The OSI Model " by Bill MacKenty is licensed under CC BY-NC-SA 4.0)

The model partitions the flow of data in a communication system into seven abstraction layers, from the physical implementation of transmitting bits across a communications medium to the highest-level representation of data of a distributed application. Each intermediate layer serves a class of functionality to the layer above it and is served by the layer below it. Classes of functionality are realized in software by standardized communication protocols.

The TCP/IP Model

The Internet protocol suite is the conceptual model and set of communications protocols used in the Internet and similar computer networks. It is commonly known as TCP/IP because the foundational protocols in the suite are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). During its development, versions of it were known as the Department of Defense (DoD) model because the development of the networking method was funded by the United States Department of Defense through DARPA. Its implementation is a protocol stack.

The Internet protocol suite provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed, and received. This functionality is organized into four abstraction layers, which classify all related protocols according to the scope of networking involved. From lowest to highest, the layers are the link layer, containing communication methods for data that remains within a single network segment (link); the internet layer, providing inter-networking between independent networks; the transport layer, handling host-to-host communication; and the application layer, providing process-to-process data exchange for applications.

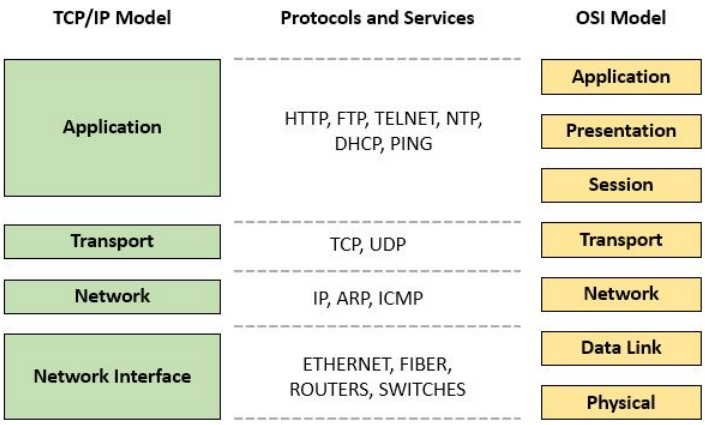


Figure \(\PageIndex{1}\): TCP/IP Model diagram. ("TCP/IP Model diagram" by Ardika6879, Wikimedia Commons is licensed under CC BY-SA 4.0)

Adapted from:

"OSI model" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"Internet protocol suite" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

This page titled 10-A.1: TCP/IP Fundamentals is shared under a not declared license and was authored, remixed, and/or curated by Patrick McClanahan.

10-A.1.1: TCP/IP Fundamentals - Identities

Network Nodes

If the network in question is the Internet or an intranet, many physical network nodes are host computers (also known as Internet nodes) identified by an IP address, and all hosts are physical network nodes. However, some data-link-layer devices such as switches, bridges and wireless access points do not have an IP host address (except sometimes for administrative purposes), and are not considered to be Internet nodes or hosts, but are considered physical network nodes and LAN nodes. In addition to an IP address nodes are known by their physical address, also known as the MAC address. Some nodes may also have an assigned hostname.

Node Identifier	Meaning
Hostname	In computer networking, a hostname (archaically nodename) is a label that is assigned to a device connected to a computer network and that is used to identify the device in various forms of electronic communication, such as the World Wide Web. Hostnames may be simple names consisting of a single word or phrase, or they may be structured.
MAC Address	A media access control address (MAC address) is a unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network segment. This use is common in most IEEE 802 networking technologies, including Ethernet, Wi-Fi, and Bluetooth. Within the Open Systems Interconnection (OSI) network model, MAC addresses are used in the medium access control protocol sublayer of the data link layer. As typically represented, MAC addresses are recognizable as six groups of two hexadecimal digits, separated by hyphens, colons, or without a separator.
IP Address	An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two main functions: host or network interface identification and location addressing. Internet Protocol version 4 (IPv4) defines an IP address as a 32-bit number. A new version of IP (IPv6), using 128 bits for the IP address, was standardized in 1998.

Network Devices

Several essential networking devices are used to connect a variety of electronic components to a network. These devices are called network devices, and they transfer data in a fast, secure and correct way over electronic networks.

Network Device	Description
Network media	Refers to the communication channels used to interconnect nodes on a computer network. Typical examples of network media include copper coaxial cables, copper twisted pair cables and optical fiber cables used in wired networks, and radio waves used in wireless data communications networks.
Bridge	A device that connects multiple network segments. It works on OSI layers 1 and 2.
Switch	A device that connects devices together on a computer network by using packet switching to receive, process and forward data to the destination device. Unlike less advanced network hubs, a network switch forwards data only to one or multiple devices that need to receive it, rather than broadcasting the same data out of each of its ports. It works on OSI layer 2.
Router	A networking device that forwards data packets between computer networks. Routers perform the "traffic directing" functions on the Internet. A data packet is typically forwarded from one router to another through the networks that constitute the internetwork until it reaches its destination node. It works on OSI layer 3.
Gateway	An interface providing a compatibility between networks by converting transmission speeds, protocols, codes, or security measures.

Frames and Packets

The terms are sometimes used interchangeably except for the layers of the OSI model they operate at. They are both logical grouped units of information with a destination address. A frame operates at the data link layer and often refers to the header and trailer that are used for synchronization and error control. A packet is most often associated with the network layer of the OSI model and includes header information and usually user data. A packet is distinguished by its frame type.

Adapted from:
"MAC address" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Hostname" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"IP address" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Networking hardware" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Network media" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-A.1.1: TCP/IP Fundamentals - Identities is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-A.1.2: TCP/IP Fundamentals - Addressing

IPv4 Addressing

Internet Protocol version 4 (IPv4) is the fourth version of the Internet Protocol (IP). It is one of the core protocols of standards-based internetworking methods in the Internet and other packet-switched networks. IPv4 was the first version deployed for production on SATNET in 1982 and on the ARPANET in January 1983. IPv4 uses a 32-bit address space which provides 4,294,967,296 unique addresses, but large blocks are reserved for special networking methods.

In the original design of IPv4, an IP address was divided into two parts: the network identifier was the most significant octet of the address, and the host identifier was the rest of the address. There is also a second number, known as the subnet mask. Subnetting is the process of designating some high-order bits from the host part as part of the network prefix and adjusting the subnet mask appropriately. This divides a network into smaller subnets. The following diagram modifies the above example by moving two bits from the host part to the network prefix to form four smaller subnets, each one quarter of the previous size. Understanding subnetting is is essential if you work in the world of networks.

IPv4 Classes

A classful network is a network addressing architecture used in the Internet from 1981 until the introduction of Classless Inter-Domain Routing in 1993. The method divides the IP address space for Internet Protocol version 4 (IPv4) into five address classes based on the leading four address bits. Classes A, B, and C provide unicast addresses for networks of three different network sizes. Class D is for multicast networking and the class E address range is reserved for future or experimental purposes.

Since its discontinuation, remnants of classful network concepts have remained in practice only in limited scope in the default configuration parameters of some network software and hardware components, most notably in the default configuration of subnet masks.

IPv4 Class	Beginning Address	Ending Address	# of Networks	Hosts per Network	Default Subnet Mask
Class A	0.0.0.0	127.0.0.0	128	167,772,126	255.0.0.0 or /8
Class B	128.0.0.0	191.255.255.255	16,384	65,635	255.255.0.0 or /16
Class C	192.0.0.0	223.255.255.255	2,097,152	253	255.255.255.0 or /24
Class D	224.0.0.0	239.255.255.255	N/A	N/A	N/A
Class E	240.0.0.0	255.255.255.255	N/A	N/A	N/A

Reserved Addresses

Within each of the first three classes is a reserved set of IPv4 addresses. These addresses are reserved for use internally within an organization.

- Class A Reserved: 10.0.0.0–10.255.255.255
- Class B Reserved: 172.16.0.0–172.31.255.255
- Class C Reserved: 192.168.0.0 - 192.168.255.255

Loopback Addresses

The class A network 127.0.0.0 is reserved for loopback. IP packets whose source addresses belong to this network should never appear outside a host. Packets received on a non-loopback interface with a loopback source or destination address must be dropped.

Link-local

There is a special address block 169.254.0.0/16 for link-local addressing. These addresses are only valid on the link (such as a local network segment or point-to-point connection) directly connected to a host that uses them. These addresses are not routable. Like private addresses, these addresses cannot be the source or destination of packets traversing the internet. These addresses are

primarily used for address autoconfiguration when a host cannot obtain an IP address from a DHCP server or other internal configuration methods.

IPv6

IPv6 is the latest version of the Internet Protocol, which identifies nodes connected to the Internet so they can be located. Every device that is connected to the Internet is identified through a unique IP address.

The previous version, known as IPv4, uses a 32-bit addressing scheme to support 4.3 billion devices. As the Internet has grown, personal computers, smartphones and now Internet of Things devices proves that a LOT more addresses are required.

Fortunately, the Internet Engineering Task Force (IETF) had the foresight that back in 1998 to create IPv6. This new addressing scheme uses 128-bit addressing to support approximately 340 trillion trillion (or 2 to the 128th power) devices.

The main advantages of IPv6 are:

- More efficient routing - IPv6 reduces the size of routing tables and makes routing more efficient and hierarchical.
- More efficient packet processing - IPv6's simplified packet header makes packet processing more efficient.
- Directed data flows - IPv6 supports multicast rather than broadcast.
- Simplified network configuration - Address auto-configuration (address assignment) is built in to IPv6.
- Support for new services - By eliminating Network Address Translation (NAT), true end-to-end connectivity at the IP layer is restored, enabling new and valuable services.
- Security - IPSec, which provides confidentiality, authentication and data integrity, is baked into in IPv6.

Adapted from:
"IPv4" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Classful network" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-A.1.2: TCP/IP Fundamentals - Addressing is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-A.1.3: TCP/IP Fundamentals - Ports and Segments

Network Ports

The physical ports on your computer allow it to communicate with peripheral devices such as your keyboard and mouse and to connect with internet devices via Ethernet cables.

Within computer networking, ports serve a similar purpose. When a computer system seeks to connect to another computer, the port serves as a communication endpoint. It is also possible for different services running on the same computer to expose various ports and communicate with one another using these ports. In simple terms, if a software application or service needs to communicate with others, it will expose a port. Ports are identified with positive 16-bit unsigned integers, ranging from 0 to 65535. Other services use this port number to communicate with the service or app. Port numbers are divided into three ranges: well-known ports, registered ports, and dynamic or private ports.

Well-known Port Numbers:

Port Number	Usage
20	File Transfer Protocol (FTP) Data Transfer
21	File Transfer Protocol (FTP) Command Control
22	Secure Shell (SSH)
23	Telnet - Remote login service, unencrypted text messages
25	Simple Mail Transfer Protocol (SMTP) E-mail Routing
53	Domain Name System (DNS) service
80	Hypertext Transfer Protocol (HTTP) used in World Wide Web
110	Post Office Protocol (POP3) used by e-mail clients to retrieve e-mail from a server
119	Network News Transfer Protocol (NNTP)
123	Network Time Protocol (NTP)
143	Internet Message Access Protocol (IMAP) Management of Digital Mail
161	Simple Network Management Protocol (SNMP)
194	Internet Relay Chat (IRC)
443	HTTP Secure (HTTPS) HTTP over TLS/SSL

Network Segments

Network segments are used to divide a network into multiple segments or subnets, each acting as its own small network. By doing this network administrators can control the flow of traffic between subnets, improve monitoring, boost performance, localize technical issues and enhance security. Below is an example of a network segment from Cisco's Packet Tracer. One segment is on the left, and another segment is on the right.

Each of the segments has its own network ID, which is part of the IP address that is assigned to the devices in each subnet.

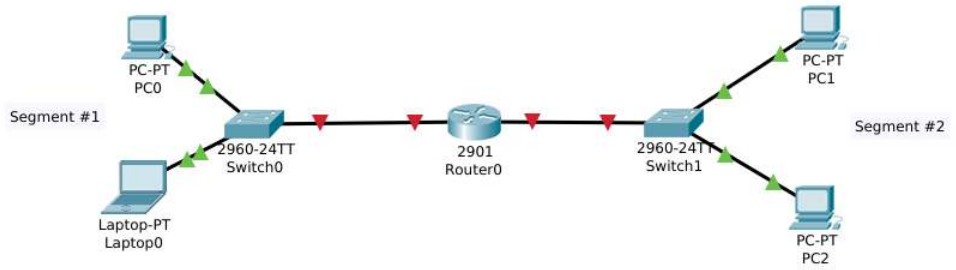


Figure **10.1.3.1**: Network Segments. ("Network Segments" by Patrick McClanahan is licensed under CC BY-SA 4.0)

Adapted from:

"14 common network ports you should know" by Kedar Vijay Kulkarni, OpenSource.com is licensed under CC BY-SA 4.0

10-A.1.3: TCP/IP Fundamentals - Ports and Segments is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-B.2: Server Roles in Linux

EXAM OBJECTIVES COVERED

2.5 Summarize and explain server roles.

When using Linux there are a wide variety of network services that can be configured and run as part of the services you are providing. Below we cover a wide variety of the most commonly used services. This is by no means an exhaustive list, or a detailed coverage of these individual services.

Authentication Services

Centralized authentication means that a single username and password, also referred to as credentials, can be used to access a number of different services because those services all look at the same database when performing authentication. This exchange with a centralized authentication source may or may not return a token that can be used in conjunction with a single sign-on (SSO) architecture.

Certificate of Authority

In cryptography, a certificate authority or certification authority (CA) is an entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others (relying parties) to rely upon signatures or on assertions made about the private key that corresponds to the certified public key. A CA acts as a trusted third party—trusted both by the subject (owner) of the certificate and by the party relying upon the certificate.

One particularly common use for certificate authorities is to sign certificates used in HTTPS, the secure browsing protocol for the World Wide Web. Another common use is in issuing identity cards by national governments for use in electronically signing documents.

Clustering

A computer cluster is a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. Unlike grid computers, computer clusters have each node set to perform the same task, controlled and scheduled by software.

The components of a cluster are usually connected to each other through fast local area networks, with each node (computer used as a server) running its own instance of an operating system. In most circumstances, all of the nodes use the same hardware and the same operating system, although in some setups (e.g., using Open Source Cluster Application Resources (OSCAR)), different operating systems can be used on each computer, or different hardware.

Clusters are usually deployed to improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Database

A database server is a server which uses a database application that provides database services to other computer programs or to computers, as defined by the client–server model. Database management systems (DBMSs) frequently provide database-server functionality, and some database management systems, such as MySQL, rely exclusively on the client–server model for database access.

Most database applications respond to a query language. Each database understands its query language and converts each submitted query to server-readable form and executes it to retrieve results.

Examples of proprietary database applications include Oracle, DB2, Informix, and Microsoft SQL Server. Examples of free software database applications include PostgreSQL; under the GNU General Public License examples include Ingres and MySQL. Every server uses its own query logic and structure. The SQL (Structured Query Language) query language is more or less the same on all relational database applications.

DHCP Services

The Dynamic Host Configuration Protocol (DHCP) is a network management protocol used on Internet Protocol (IP) networks, whereby a DHCP server dynamically assigns an IP address and other network configuration parameters to each device on the network, so they can communicate with other IP networks. A DHCP server enables computers to request IP addresses and networking parameters automatically from the Internet service provider (ISP), reducing the need for a network administrator or a user to manually assign IP addresses to all network devices. In the absence of a DHCP server, a computer or other device on the network needs to be manually assigned an IP address, or to assign itself an APIPA address, the latter of which will not enable it to communicate outside its local subnet.

DHCP can be implemented on networks ranging in size from home networks to large campus networks and regional ISP networks. A router or a residential gateway can be enabled to act as a DHCP server. Most residential network routers receive a globally unique IP address within the ISP network. Within a local network, a DHCP server assigns a local IP address to each device connected to the network.

Name/DNS Services

A DNS hosting service is a service that runs Domain Name System servers. Most, but not all, domain name registrars include DNS hosting service with registration. Free DNS hosting services also exist. Many third-party DNS hosting services provide Dynamic DNS.

DNS hosting service is optimal when the provider has multiple servers in various geographic locations that provide resilience and minimize latency for clients around the world. By operating DNS nodes closer to end users, DNS queries travel a much shorter distance, resulting in faster Web address resolution speed.

DNS can also be self-hosted by running on generic Internet hosting services.

Adapted from:
"Computer cluster" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Database server" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"DNS hosting service" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Certificate authority" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-B.2: Server Roles in Linux is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-B.2-3: Server Roles in Linux (continued)

Name Server/DNS Services

The Domain Name System (DNS) provides the database to be used in the translation from human-readable hostnames, such as www.opensource.com, to IP addresses, like 54.204.39.132, so that your internet-connected computers and other devices can access them. The primary function of the BIND (Berkeley Internet Name Domain) software is that of a domain name resolver that uses that database. There is other name resolver software, but BIND is currently the most widely used DNS software on the internet. The terms name server, DNS, and resolver will be used pretty much interchangeably throughout this article.

Without these name resolver services, surfing the web as freely and easily as we do would be nearly impossible. As humans, we tend to do better with names like Opensource.com, while computers do much better with numbers like 54.204.39.132. Therefore, we need a translation service to convert the names that are easy for us to the numbers that are easy for our computers.

NTP Services

The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks. In operation since before 1985, NTP is one of the oldest Internet protocols in current use. NTP was designed by David L. Mills of the University of Delaware.

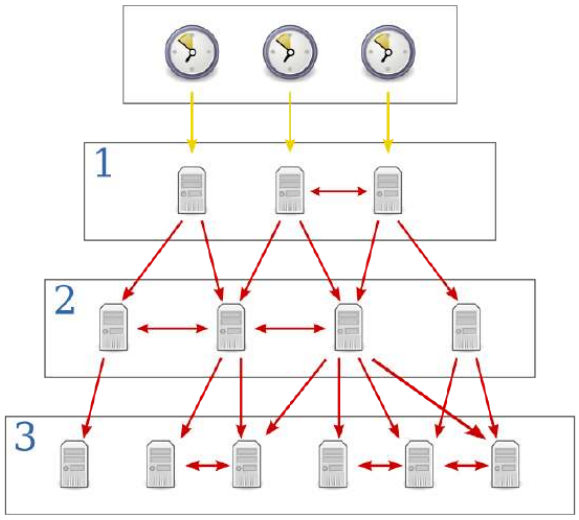


Figure **10.2 – 3.1**: NTP In Action. (""Network_Time_Protocol_servers_and_clients.svg" by Benjamin D. Esham, Wikiemedia Commons is in the Public Domain, CC0)

NTP is intended to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC). NTP can usually maintain time to within tens of milliseconds over the public Internet, and can achieve better than one millisecond accuracy in local area networks under ideal conditions. Asymmetric routes and network congestion can cause errors of 100 ms or more.

Networked devices are connected to another device to obtain NTP services. This continues until at some point in the network a device is connected to an actual atomic clock or other device that keeps the time.

Proxy Services

A proxy server is a server that behaves as an intermediary between a desktop computer and the internet. The proxy server allows a client machine to make an indirect connection to network servers and services. There are many reasons why you might want to include a proxy server on your network:

- Share internet connection on a network
- Speed up internet connections
- Hide the IP address of the client computer
- Enforce internet access control

- Scan content be sent by the local computer

SNMP Services

Simple Network Management Protocol (SNMP) is an Internet Standard protocol for collecting and organizing information about managed devices on IP networks and for modifying that information to change device behavior. Devices that typically support SNMP include cable modems, routers, switches, servers, workstations, printers, and more.

SNMP is widely used in network management for network monitoring. SNMP exposes management data in the form of variables on the managed systems organized in a management information base (MIB) which describe the system status and configuration. These variables can then be remotely queried (and, in some circumstances, manipulated) by managing applications.

Adapted from:
"Network Time Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Introduction to the Domain Name System (DNS)" by David Both, OpenSource.com is licensed under CC BY-SA 4.0
"Simple Network Management Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-B.2-3: Server Roles in Linux (continued) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-B.2.1: Server Roles in Linux - Continued

Email

The Simple Mail Transfer Protocol (SMTP) is a communication protocol for electronic mail transmission. As an Internet standard, SMTP was first defined in 1982 by RFC 821, and updated in 2008 by RFC 5321 to Extended SMTP additions, which is the protocol variety in widespread use today. Mail servers and other message transfer agents use SMTP to send and receive mail messages. SMTP servers commonly use the Transmission Control Protocol on port number 25.

File/Print Services

File servers may also be categorized by the method of access: Internet file servers are frequently accessed by File Transfer Protocol or by HTTP (but are different from web servers that often provide dynamic web content in addition to static files). Servers on a LAN are usually accessed by SMB/CIFS protocol (Windows and Unix-like) or NFS protocol (Unix-like systems).

Whereas, a print server is a type of server that connects printers to client computers over a network. It accepts print jobs from the computers and sends the jobs to the appropriate printers, queuing the jobs locally to accommodate the fact that work may arrive more quickly than the printer can actually handle. Ancillary functions include the ability to inspect the queue of jobs to be processed, the ability to reorder or delete waiting print jobs, or the ability to do various kinds of accounting (such as counting pages, which may involve reading data generated by the printer(s)). Print servers may be used to enforce administration policies, such as color printing quotas, user/department authentication, or watermarking printed documents.

Load Balancing Services

In computing, load balancing refers to the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient. However, in our usage we are referring more to network load balancing (commonly referred to as dual-WAN routing or multihoming) as the ability to balance traffic across two or more WAN links without using complex routing protocols like BGP.

This capability balances its network sessions like Web, email, etc. over multiple connections in order to spread out the amount of bandwidth used by each LAN user, thus increasing the total amount of bandwidth available. For example, a user has a single WAN connection to the Internet operating at 1.5Mbit/s. They wish to add a second broadband (cable, DSL, wireless, etc.) connection operating at 2.5Mbit/s. This would provide them with a total of 4Mbit/s of bandwidth when balancing sessions.

Logging Services

Most Linux systems already centralize logs using a syslog daemon. Linux has a special directory for storing logs called /var/log. This directory contains logs from the OS itself, services, and various applications running on the system. Syslog is actually a standard for creating and transmitting logs. The use of "syslog" can refer to either:

- The Linux syslog service, which listens for events using a network that system applications write to. Syslog can write its messages to a local file, usually in /var/log, or forward those messages to a remote log server. There are different syslog implementations including rsyslogd and syslog-ng.
- The syslog protocol, which is defined in RFC 5424, is a transport protocol that specifies how to transmit logs over a network. The standard also defines a data format of how messages are structured. The default network port is 514 for plaintext messages and port 6514 for encrypted messages.
- A syslog message, which is any log formatted in the syslog message format. A syslog message consists of a standardized header and message containing the log’s contents.

Monitoring Services

Linux, and the Linux community, provide a significant number of tools to monitor specific aspects of the Linux system.

Some of the tools include:

- The Linux **`top`** command. There are many open source variants of **`top`**.
- Network monitoring tools - ethertool, tcpdump, MRTG, traceroute to name just a few.

- Linux server monitory tools - Linux Dash, Nagios, vmstat.
- Log monitoring tools - Multitail, Logwatch.
- Network Manager tools - ifconfig, GNOME Network Manager, Wicd.
- Linux performance monitoring tools - Sysstat, Webmin, whowatch.

Adapted from:

"Simple Mail Transfer Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"File server" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Print server" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Network Load Balancing" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Load balancing (computing)" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-B.2.1: Server Roles in Linux - Continued is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-B.2.4: Server Roles in Linux (continued)

SSH Services

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

SSH provides a secure channel over an unsecured network by using a client–server architecture, connecting an SSH client application with an SSH server. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. SSH is generally used to access Unix-like operating systems, but it can also be used on Microsoft Windows. Windows 10 uses OpenSSH as its default SSH client and SSH server.

There are several clients that allow users to connect to an SSH server. Linux systems come with a command line, as well as a GUI based clients. For Windows the most popular is Putty, which does not come with Windows but is easily downloaded.

Virtualization/Container Services

Virtualization

Virtualization is a technique of how to separate a service from the underlying physical delivery of that service. It is the process of creating a virtual version of something like computer hardware. It was initially developed during the mainframe era. It involves using specialized software to create a virtual or software-created version of a computing resource rather than the actual version of the same resource. With the help of virtualization, multiple operating systems and applications can run on the same machine and its hardware at the same time, increasing the utilization and flexibility of hardware.

Virtual Machines

In computing, a virtual machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.

A virtual machine is a file, called an image, that mimics an actual computer. In other words, it creates a computer within a computer. It runs within an app like any other program, giving the end user the same experience on a virtual machine as they would have on the guest operating system itself. The host machine is protected from the guest system, meaning that the software inside a virtual machine can’t access the host computer itself, other than some of the hardware devices if so configured.

VPN Services

A virtual private network (VPN) extends a private network across a public network and enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network. Applications running across a VPN may therefore benefit from the functionality, security, and management of the private network. Encryption is a common, although not an inherent, part of a VPN connection.

VPN technology was developed to provide access to corporate applications and resources to remote or mobile users, and to branch offices. For security, the private network connection may be established using an encrypted layered tunneling protocol, and users may be required to pass various authentication methods to gain access to the VPN. In other applications, Internet users may secure their connections with a VPN to circumvent geo-blocking and censorship or to connect to proxy servers to protect personal identity and location to stay anonymous on the Internet. Some websites, however, block access to known IP addresses used by VPNs to prevent the circumvention of their geo-restrictions, and many VPN providers have been developing strategies to get around these blockades.

A VPN is created by establishing a virtual point-to-point connection through the use of dedicated circuits or with tunneling protocols over existing networks. A VPN available from the public Internet can provide some of the benefits of a wide area network (WAN). From a user perspective, the resources available within the private network can be accessed remotely.

Web Services

The term Web service (WS) refers to either a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web, or it is a server running on a computer device. The server listens for requests at a particular port over a network, serves web documents (HTML, JSON, XML, images), and creates web applications services, which serve in solving specific domain problems over the Web (WWW, Internet, HTTP)

There is a lot that falls under the heading of "web services." For the most part it communicates via one of the two protocols: 1) HTTP, which uses TCP port 80; or 2) HTTPS, secure HTTP, which runs on TCP port 443.

Adapted from:
"Secure Shell Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Virtualization In Cloud Computing and Types" by Namrata Bisht, Geeks for Geeks is licensed under CC BY-SA 4.0
"Dynamic Host Configuration Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-B.2.4: Server Roles in Linux (continued) is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-C.3: Network Connection

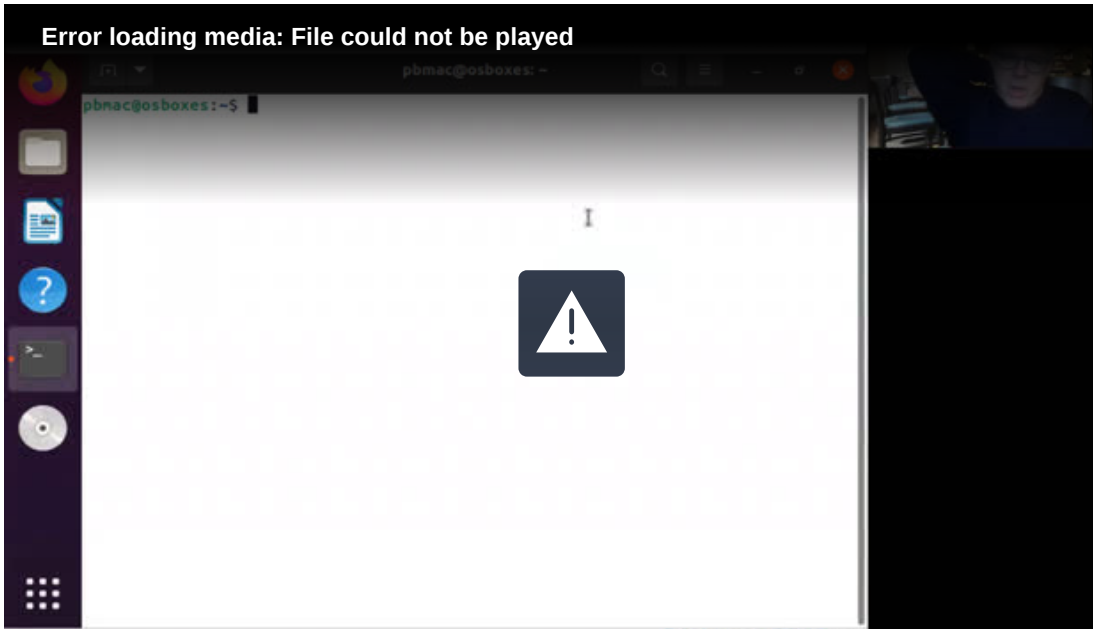
EXAM OBJECTIVES COVERED

1.3 Given a scenario, configure and verify network connection parameters.
4.1 Given a scenario, analyze system properties and remediate accordingly.

Configuring the System Hostname

The **`hostnamectl`** command provides a proper API used to control Linux system hostname and change its related settings. The command also helps to change the hostname without actually locating and editing the /etc/hostname file on a given system.

Using the hostnamectl command is the topoic of this video, just a quick overview and some examples of usage.



Syntax:

```
hostnamectl [ OPTIONS ] COMMAND ....
```

Command Options:

Options	Meaning
--no-ask-password	Do not query the user for authentication for privileged operations.
--static, --transient, --pretty	If status is invoked (or no explicit command is given) and one of these switches is specified, hostnamectl will print out just this selected hostname.
-H, --host=	Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":", and then a container name, separated by "/", which connects directly to a specific container on the specified host.
-M, --machine=	Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character.
-h, --help	Print a short help text and exit.
--version	Print a short version string and exit.

Where COMMAND can be any of the following:

- status: Used to check current hostname settings.
- set-hostname NAME: Used to set system hostname.
- set-icon-name NAME: Used to set icon name for host.
- set-chassis NAME: Set chassis type for host.

Types of hostname:

- Static: Assigned by system admin and used to initialize the kernel hostname during boot time.
- Dynamic or Transient: Assigned by DNS server or DHCP server during run time.
- Pretty: A high-level hostname assigned by system admin or end-user.

The example below shows the current status of the server. We then change the Static name to PbmacUbuntu, which can be seen in the second status command. However, notice that the second status query shows a difference between the static hostname and the transient hostname.

```
pbmac@pbmac-server $ hostnamectl status
   Static hostname: pbmac-server
         Icon name: computer-desktop
           Chassis: desktop
        Machine ID: 75d57d2bcd2831ebf58bcc1a000008d4
           Boot ID: 688e9be968c34bffa95173e1d3e1effe
  Operating System: Ubuntu 18.04.4 LTS
            Kernel: Linux 4.15.0-91-generic
      Architecture: x86-64
pbmac@pbmac-server $ sudo hostnamectl set-hostname PbmacUbuntu --static
pbmac@pbmac-server $ hostnamectl status
   Static hostname: PbmacUbuntu
Transient hostname: pbmac-server
```

The static host name is the traditional hostname which can be chosen by the user, and is stored in the `/etc/hostname` file. The “transient” hostname is a dynamic host name maintained by the kernel. It is initialized to the static host name by default, whose value defaults to “localhost.” It can be changed by DHCP or DHCP at runtime.

IP Configuration

Any computer that needs to connect to a network must have a couple of pieces of information: 1) an IP address; 2) a hostname; and 3) a MAC address. We discussed the hostname in the previous section. The MAC address is a hardware address that is assigned to your network interface by the manufacturer. That leaves the IP address to discuss.

IP stands for Internet Protocol. An Internet Protocol address, or IP address is a unique identifier given to every machine in a network. An IP address serves two primary functions. It is used as an interface identification for a network of machines and it also serves to provide a location of that machine, much like a physical address for a home or business. Because an IP address is a unique identifier, it allows computers to send and receive information to and from specific computers in a given network.

Each device on an IP network requires three different pieces of information in order to correctly communicate with other devices on the network: an IP address, a subnet mask, and a broadcast address. You will usually see each of these numbers written as four "octets" (e.g. 198.41.12.151, 255.255.255.0, and 198.41.12.255). Each device also needs a MAC address.

Linux Network Manager

NetworkManager is a program for providing detection and configuration for systems to automatically connect to networks. NetworkManager's functionality can be useful for both wireless and wired networks. For wireless networks, NetworkManager prefers known wireless networks and has the ability to switch to the most reliable network. NetworkManager-aware applications can switch from online and offline mode. NetworkManager also prefers wired connections over wireless ones, and has support for modem connections and certain types of VPN. NetworkManager was originally developed by Red Hat and now is hosted by the GNOME project.

The `nmcli` Command

The **`nmcli`** command is a tool which is used for controlling NetworkManager. **`nmcli`** command can also be used to display network device status, create, edit, activate/deactivate, and delete network connections.

It is used in scripts, and instead of manually managing the network connections it utilizes NetworkMaager via **`nmcli`**. Servers use the **`nmcli`** command on headless machines since it can be used to control system-wide connections via NetworkManager without a GUI.

Syntax:

```
nmcli [ OPTIONS ] OBJECT { COMMAND | help }
```

Where the **OBJECT** can be any one of the following:

- g[eneral] - NetworkManager's general status and operations
- n[etworking] - overall networking control
- r[adio] NetworkManager - radio switches
- c[onnection] - NetworkManager's connections
- d[evice] - devices managed by NetworkManager
- a[gent] - NetworkManager secret agent or polkit agent
- m[onitor] - monitor NetworkManager changes

Options:

Options	Option Meaning
-t[erse]	terse output
-p[retty]	pretty output
-m[ode]	tabular|multiline output mode
-c[olors] auto|yes|no	whether to use colors in output
-f[ields] <field, 1field2 ...>|all|common	specify fields to output

The **`nmcli`** command can show simple status of the system, or it can show all of the current connections.

```
pbmac@pbmac-server $ nmcli device status
DEVICE   TYPE      STATE      CONNECTION
enp0s25  ethernet  connected  Wired connection 1
lo       loopback  unmanaged  --
```

The nmtui Command

nmtui is an alternative to **nmcli**. **nmtui** is short for Network Manager Text User Interface, and is another handy tool that allows system administrators to configure network interfaces in Linux using a graphical display. The **nmtui** command can be run directly from the terminal or even in putty.

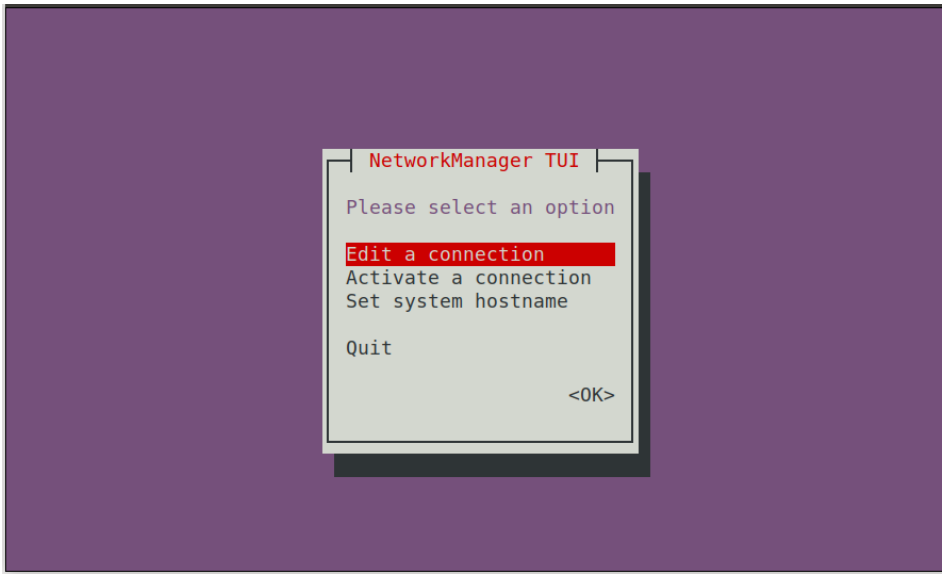
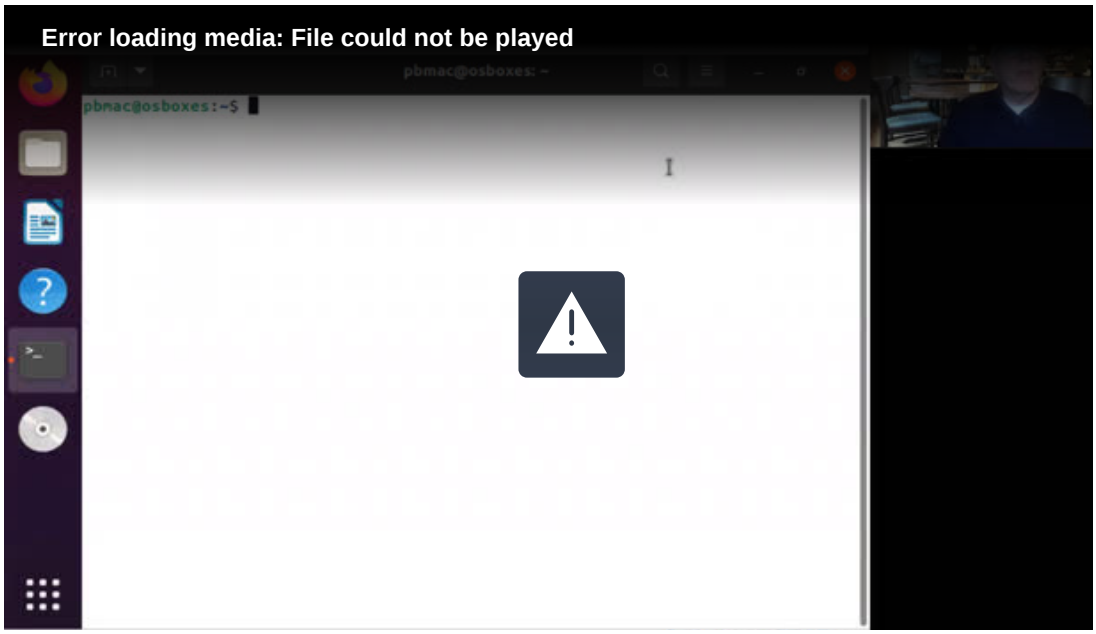


Figure 10.3.1: Network Manager TUI. ("Network Manager" by Patrick McClanahan is licensed under CC BY-SA 4.0)

With this interface you use the arrows keys on the keyboard to move through the list, or use the Tab key to move to OK and then hit the Enter key. It takes a bit of getting used to - but if you do NOT have a GUI installed, you can still use tools like this.

The video compares the ifconfig command and the newer ip command.



The `ifconfig` Command

The **`ifconfig`** command is used to configure the kernel-resident network interfaces. It is used at the boot time to set up the interfaces as necessary. After that, it is usually used when needed during debugging or when you need system tuning. Also, this command is used to assign the IP address and netmask to an interface or to enable or disable a given interface.

Syntax:

```
ifconfig [ OPTIONS ] [INTERFACE]
```

The options are quite numerous, so it is suggested that you use the man page to read up on the available options.

Newer versions of some Linux distributions don't have **`ifconfig`** command pre-installed, since it is being replavced by the ip command. So, if there is an error "ifconfig: command not found" use the system's package install utility to install the package.

The `ip` Command

The **`ip`** command in Linux is present in the net-tools which is used for performing several network administration tasks. IP stands for Internet Protocol. This command is used to show or manipulate routing, devices, and tunnels. It is similar to, and replaces, the old **`ifconfig`** command. It is much more powerful with more functions and facilities attached to it. The **`ip`** command is used to perform several tasks like assigning an address to a network interface or configuring network interface parameters.

It can perform several other tasks like configuring and modifying the default and static routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, assigning, deleting and setting up IP addresses and routes.

Syntax:

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
```

Again, the options are numerous, so please check out the man page for this command.

Where the OBJECT can be any one of the following:l[ink] – used to display and modify network interfaces.

- addr[ess] – used to display and modify protocol addresses (IP, IPv6).
- r[oute] – used to display and alter the routing table.
- n[eigh] – used to display and manipulate neighbor objects (ARP table)

The following examples show several uses of the **`ip`** command. The first examples shows the interface and its status. Then both the IPv4 and IPV6 addresses are displayed along with the address duration (addresses assigned by DHCP servers can expire after a predetermined amount of time).

```
pbmac@pbmac-server $ ip address show enp0s25
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
    link/ether bc:30:5b:ca:95:8a brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.6/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s25
       valid_lft 68970sec preferred_lft 68970sec
    inet6 fe80::2e86:9a11:9c02:4a5/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
pbmac@pbmac-server $ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default ql
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group
    link/ether bc:30:5b:ca:95:8a brd ff:ff:ff:ff:ff:ff
```

The second example simply shows the two interfaces, the loopback interface and the ethernet interface, and the status.

Adapted from:
"8.4. Using Hostnamectl" by Unknown, Fedora Documentation Project is licensed under CC BY-SA 3.0
"hostnamectl command in Linux with Examples" by Suraj1994, Geeks for Geeks is licensed under CC BY-SA 4.0
"NetworkManager" by Multiple Contributors, Arch Linux is licensed under CC BY-SA 3.0
"nmcli command in Linux with Examples" by Suraj1994, Geeks for Geeks is licensed under CC BY-SA 4.0
"ifconfig command in Linux with Examples" by rossoskull, Geeks for Geeks is licensed under CC BY-SA 4.0
"ip command in Linux with examples" by raghavmangal22, Geeks for Geeks is licensed under CC BY-SA 4.0

10-C.3: Network Connection is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-D.4: Static VS Dynamic IP Addresses

EXAM OBJECTIVES COVERED

1.3 Given a scenario, configure and verify network connection parameters.

Static vs Dynamic IP Addresses

IP stands for Internet Protocol. IP address may be a distinctive numerical symbol allotted to every device on a network to spot each affiliation unambiguously.

The distinction between a Static and Dynamic IP address has to do with how the address gets assigned and how long that address is valid. Both static IP Address and Dynamic IP Address are used to identify a computer on a network. Static IP addresses are assigned by the network administrator, who gets them from their Internet Service Provider. Static addresses remain fixed to a specific device until someone changes that device's IP address.

Dynamic IP addresses are provided by a DHCP server. Dynamic addresses expire after a preset amount of time and must be renewed. Most often a network's administrator will use one of the private network address spaces to assign addresses within the organization's network. We will discuss DHCP in more detail on the next page.

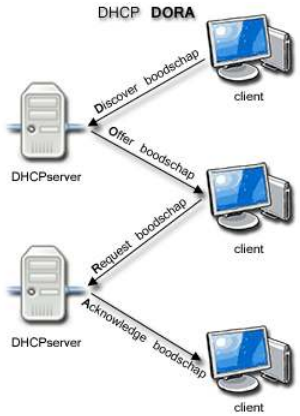


Figure **10.4.1**: DHCP handshake. ("DHCPDORA.png" by Endaargaanweweer, Wikimedia Commons is in the Public Domain, CC0)

Differences between Static and Dynamic IP addressing:

STATIC IP ADDRESS	DYNAMIC IP ADDRESS
Provided by ISP (Internet Service Provider).	Provided by DHCP (Dynamic Host Configuration Protocol).
Static IP address does not change any time, which means if a static IP address is provided then it can't be changed or modified.	Can change any time.
Less secure.	There is a lower amount of risk than static IP address's risk.
Difficult to designate.	Easy to designate.
The device designed by static IP address can be traced.	The device designed by dynamic IP address can't be traced.
More stable than dynamic IP address.	Less stable than static IP address.
The cost to maintain the static IP address is higher.	The maintaining cost of dynamic IP address is less.
Used where computational data is less confidential.	Used where data is more confidential and needs more security.

Adapted from:
"Difference between Static and Dynamic IP address" by MKS075, Geeks for Geeks is licensed under CC BY-SA 4.0

10-D.4: Static VS Dynamic IP Addresses is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-D.5: DHCP Configuration

DHCP Configuration

DHCP is an abbreviation for Dynamic Host Configuration Protocol. It is an application layer protocol used by hosts for obtaining network setup information. DHCP across a network is controlled by a DHCP server that dynamically distributes network configuration parameters such as IP addresses, subnet mask and gateway address.

To clarify what DHCP is about:

- Dynamic – Means it works automatically
- Host – Any computer that is connected to the network
- Configuration – To configure a host means to provide network information(ip address,subnet mask,Gateway address) to a host
- Protocol – Set of rules

Disadvantage of Manually Configuring the Host

Configuring a host when it is connected to the network can be done either manually i.e., by the network administrator or by the DHCP server. In case of home networks, manual configuration is quite easy, whereas in the large networks, the Network Administrator might face many problems. Also, the manual configuration is prone to mistakes. For example, a Network Administrator might assign an IP address which was already assigned. Thus, causing difficulty for both administrator as well as neighbors on the network.

Configuring a Host Using DHCP

To configure a host, we require the following things:

- **Leased IP address** – IP address which is "leased" to a particular network device for a specific time frame which can be a few hours, few days or even few weeks.
- **Subnet Mask** – This is determined by which subnet the host is on.
- **Gateway address** – The Gateway is the Internet Service Provider that connects user to the internet. The Gateway address lets the host know where to send network packets that are being sent to an outside network.

There are several DHCP entities that administrators need to be familiar with:

- **DHCP server:** It automatically provides network information (IP address, subnet mask, gateway address) - this data is "leased" to the requesting host. Once the duration is expired, that network information can be assigned to another machine. It also keeps track of which IP addresses are assigned and who they are assigned to.
- **DHCP client:** Any network device which requests an IP address lease is considered as DHCP client.
- **DHCP Relay Agent:** In case we have only one DHCP server for multiple networks, this Agent forwards the DHCP request to the DHCP server. Using a DHCP Relay Agent allows the use of a single DHCP server to configure multiple LANs.

The DHCP Lease and Renewal Process

DHCP servers listen on port number 67 and the client listens on port 68. It is a Client/Server protocol which uses UDP services. IP address is assigned from a pool of addresses known to the server. In DHCP, the client and the server usually exchange four DHCP messages in order to make a connection, also called DORA process, but there are eight DHCP messages in the process.

These messages are given as below:

DHCP discover message – This is a first message in the communication process between server and client. This message is generated by the client in order to discover if there is any DHCP server present in a network. This message is broadcasted to all devices present in a network to find the DHCP server. This message is 342 or 576 bytes long.

DHCP offer message – The server will respond to a host with this message specifying the unleased IP address and the other configuration information. This message is broadcasted by the server. The message is 342 bytes. If there are more than one DHCP servers present in the network the client will accept the first DHCP OFFER message it receives. There is a server ID specified in the DHCP OFFER packet in order for the client to identify which server it is dealing with.

DHCP request message – When a client receives an offer message, it responds by broadcasting a DHCP request message. The client then sends ARP request in order to find out if there is any other host in the network with same IP address. If there is no reply then the client assumes there is no other client with same IP address in the network and the message is broadcasted to server showing the acceptance of the IP address.

DHCP acknowledgement message – In response to the request message, the server will track the client ID and the assigned IP address and the lease time.

DHCP negative acknowledgement message – Whenever a DHCP server receives a request for an IP address that is invalid according to the scopes that is configured with, it sends DHCP Nak message to client. For example, if the server has no IP unused addresses in its specified address pool, then this message is sent to client.

DHCP decline – If the DHCP client determines the offered configuration parameters are different or invalid, it sends the DHCP server a decline message. If the client receives a reply to the ARP by a host the client sends DHCP decline message to the server showing the offered IP address is already in use.

DHCP release – A DHCP client sends a DHCP release packet to the server telling the server to release IP address and cancel any remaining lease time.

DHCP inform – If a client address has obtained IP address manually then the client uses a DHCP inform to obtain other local configuration parameters, such as domain name. In reply to the dhcp inform message, DHCP server generates DHCP ack message with local configuration suitable for the client without allocating a new IP address. This DHCP ack message is unicast to the client.

The /etc/dhcp/dhclient.conf File

The dhclient.conf file contains configuration information for dhclient, the Internet Systems Consortium DHCP Client.

The dhclient.conf file is a free-form ASCII text file. It is parsed by the recursive-descent parser built into dhclient. The file may contain extra tabs and newlines for formatting purposes. Keywords in the file are case-insensitive. Comments may be placed anywhere within the file (except within quotes). Comments begin with the # character and end at the end of the line.

The dhclient.conf file can be used to configure the behavior of the client in a wide variety of ways: protocol timing, information requested from the server, information required of the server, defaults to use if the server does not provide certain information, values with which to override information provided by the server, or values to prepend or append to information provided by the server. The configuration file can also be pre-initialized with addresses to use on networks that don't have DHCP servers.

Adapted from:
"Dynamic Host Configuration Protocol (DHCP)" by saurabhsharma56, Geeks for Geeks is licensed under CC BY-SA 4.0
"How DHCP server dynamically assigns IP address to a host?" by Brahmani Sai., Geeks for Geeks is licensed under CC BY-SA 4.0

10-D.5: DHCP Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-D.6: Name Resolution

Name Resolution

Name resolution is a process of converting an IP address to a user friendly computer name. Originally networks used host files to resolve names to IP addresses. These files were a text file that the computer consulted if a name lookup was necessary. All the computers on the network and their IP address mappings had to be entered manually into these files. The file was then automatically copied to all the other machines on the network. When a resource was required, by the user typing its name, the machine referred to the host file to find the IP address.

As networks got larger and larger the text based hosts file grew too large and required too much maintenance. Then came the advent of dynamically assigned IP addresses. IP to name resolution took on a transitory nature with the possibility of IP addresses changing on a weekly basis. The days of the host files were numbered as the primary tool of name resolution.

The `/etc/hosts` File

The computer file **`hosts`** is a file that maps hostnames to IP addresses. It is a plain text file. Originally a file named HOSTS.TXT was manually maintained and made available via file sharing. It contains the hostnames and address of hosts as contributed for inclusion by member organizations. The Domain Name System, first described in 1983 and implemented in 1984, automated the publication process and provided instantaneous and dynamic hostname resolution in the rapidly growing network. In modern operating systems, the hosts file remains an alternative name resolution mechanism, configurable often as part of facilities such as the Name Service Switch as either the primary method or as a fallback method.

The format of the file is quite simple: an IP address, followed by a hostname. In the following example the first entry is the local loopback entry - in IPv4 and then IPv6 format; then the entry for pbmac-server, which has no domain portion of the hostname; and then the entry for www.google.com - which includes the google.com domain. In this case it is assumed the pbmac-server is on the local domain.

```
pbmac@pbmac-server $ cat /etc/hosts
127.0.0.1  localhost loopback
::1        localhost
192.168.0.1    pbmac-server
172.217.6.78   www.googlecom
```

The `/etc/resolv.conf` File

The **`/etc/resolv.conf`** is a resolver configuration file for Linux. It is used to configure the domain name servers for a particular host. The file **`/etc/resolv.conf`** file contains information that is read by the resolver routines the first time they are invoked by a process. The file is designed to be human readable and contains a list of keywords with values that provide various types of resolver information.

The file is simply a keyword, followed by an IP address. The IP address points to the DNS server that handles ALL of the name resolutions that are not provided in the **`/etc/hosts`** file.

```
pbmac@pbmac-server $ cat /etc/resolv.conf
nameserver 10.0.80.11
nameserver 10.0.80.12
```

The /etc/nsswitch.conf File

A system administrator usually configures the operating system's name services using the file **`/etc/nsswitch.conf`**. This file lists databases (such as passwd, shadow and group) and one or more sources for obtaining that information. Examples for sources are files for local files, ldap for the Lightweight Directory Access Protocol, nis for the Network Information Service, nisplus for NIS+, and wins for Windows Internet Name Service.

The nsswitch.conf file has line entries for each service consisting of a database name in the first field, terminated by a colon, and a list of possible source databases mechanisms in the second field. A typical file might look like:

```
pbmac@pbmac-server $ cat /etc/nsswitch.conf
passwd:     files ldap
shadow:     files
group:      files ldap

hosts:      files dns
```

The order of the services listed determines in which order NSS will attempt to use those services to resolve queries on the specified database. In the above example the name resolution uses the **`/etc/hosts`** file first, and if that does not provide a resolution, then the system consults the DNS server at the IP address shown in the **`/etc/resolv.conf`** file.

Adapted From:
"hosts (file)" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"/etc/resolv.conf" by Multiple contributors, Cyberciti.biz is licensed under CC BY-NC-SA 3.0
Name Service Switch" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-D.6: Name Resolution is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-E.7: Cloud Configuration

EXAM OBJECTIVES COVERED

1.5 Compare and contrast cloud and virtualization concepts and technologies.

Cloud Computing

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server.

Clouds may be limited to a single organization (enterprise clouds), or be available to many organizations (public cloud).

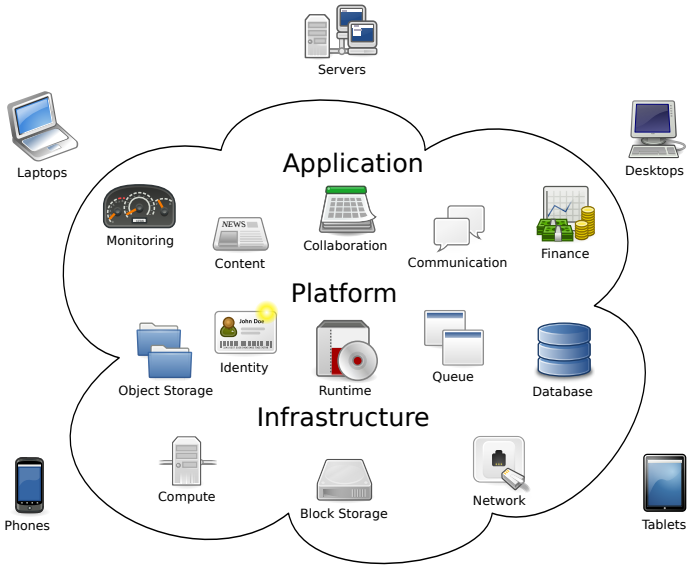


Figure 10.7.1: Cloud computing metaphor: the group of networked elements providing services need not be individually addressed or managed by users; instead, the entire provider-managed suite of hardware and software can be thought of as an amorphous cloud.. ("Cloud_computing.svg" by Sam Johnston, Wikimedia Commonss is licensed under CC BY-SA 3.0)

Cloud computing is a relatively new technology, and there is a great deal of new terminology that comes with new technology. The National Institute of Standards and Technology has attempted to address some of this for cloud computing. There are five essential characteristics of cloud computing:

Characteristic	Description
On-demand self-service	A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
Broad network access	Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
Resource pooling	The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.
Rapid elasticity	Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
Measured service	Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Service Models of Cloud Computing

Along with these characteristics there are several service models that we find in cloud computing:

Model	Description
Software as a Service (SaaS)	The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.
Platform as a Service (PaaS)	The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.
Infrastructure as a Service (IaaS)	The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls)
Mobile "backend" as a service (MBaaS)	In the mobile "backend" as a service (m) model, also known as backend as a service (BaaS), web app and mobile app developers are provided with a way to link their applications to cloud storage and cloud computing services with application programming interfaces (APIs) exposed to their applications and custom software development kits (SDKs). Services include user management, push notifications, integration with social networking service and more. This is a relatively recent model in cloud computing, with most BaaS startups dating from 2011 or later, but trends indicate that these services are gaining significant mainstream traction with enterprise consumers.
Function as a service (FaaS)	Function as a service (FaaS) is a service-hosted remote procedure call that leverages serverless computing to enable the deployment of individual functions in the cloud that run in response to events. FaaS is included under the broader term serverless computing, but the terms may also be used interchangeably.
Serverless computing	Serverless computing is a cloud computing code execution model in which the cloud provider fully manages starting and stopping virtual machines as necessary to serve requests, and requests are billed by an abstract measure of the resources required to satisfy the request, rather than per virtual machine, per hour. Despite the name, it does not actually involve running code without servers. Serverless computing is so named because the business or person

Model	Description
	that owns the system does not have to purchase, rent or provision servers or virtual machines for the back-end code to run on.

Models of Cloud Deployment

Model	Description
Private cloud	A cloud infrastructure operated solely for a single organization, whether managed internally or by a third party, and hosted either internally or externally. Undertaking a private cloud project requires significant engagement to virtualize the business environment, and requires the organization to reevaluate decisions about existing resources. It can improve business, but every step in the project raises security issues that must be addressed to prevent serious vulnerabilities.
Public cloud	Cloud services are considered "public" when they are delivered over the public Internet, and they may be offered as a paid subscription, or free of charge. Architecturally, there are few differences between public- and private-cloud services, but security concerns increase substantially when services (applications, storage, and other resources) are shared by multiple customers. Most public-cloud providers offer direct-connection services that allow customers to securely link their legacy data centers to their cloud-resident applications.
Hybrid cloud	Hybrid cloud is a composition of a public cloud and a private environment, such as a private cloud or on-premises resources, that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect collocation, managed and/or dedicated services with cloud resources. Gartner defines a hybrid cloud service as a cloud computing service that is composed of some combination of private, public and community cloud services, from different service providers. A hybrid cloud service crosses isolation and provider boundaries so that it can't be simply put in one category of private, public, or community cloud service.

There are additional deployment models such as Community cloud, Distributed cloud, Multicloud, High Performance Computing cloud, and Big Data cloud. However, the ones listed in the table above are the most common deployment models.

Providers of Cloud Services

There are quite a large number of cloud service providers. Some of the larger, more well known companies include: Microsoft Azure, Amazon Web Services (AWS), Google Cloud, Alibaba Cloud, IBM Cloud, Oracle, Salesforce, SAP, Rackspace Cloud, and VMWare. This is only a very small sampling. For additional service providers consult places like Wikipedia's Cloud computing providers category page.

Adapted from:
"Cloud computing" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-E.7: Cloud Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-E.8: The Cloud and Virtualization

Virtualization

In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.

Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Microsoft Windows may
host a virtual machine that looks like a computer with the Ubuntu Linux operating system; Ubuntu-based software can be run on the virtual machine.

In hardware virtualization, the host machine is the machine that is used by the virtualization and the guest machine is the virtual machine. The words host and guest are used to distinguish the software that runs on the physical machine from the software that runs on the virtual machine. The software or firmware that creates a virtual machine on the host hardware is called a hypervisor or virtual machine monitor.

Benefits of Virtualization

1. More flexible and efficient allocation of resources.
2. Enhances development productivity.
3. Lowers the cost of IT infrastructure.
4. Remote access and rapid scalability.
5. High availability and disaster recovery.
6. Pay per use of the IT infrastructure on demand.
7. Enables running multiple operating system.

Hypervisor

A hypervisor (or virtual machine monitor, VMM, virtualizer) is computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources: for example, Linux, Windows, and macOS instances can all run on a single physical x86 machine. This contrasts with operating-system-level virtualization, where all instances (usually called containers) must share a single kernel, though the guest operating systems can differ in user space, such as different Linux distributions with the same kernel.

The term hypervisor is a variant of supervisor, a traditional term for the kernel of an operating system: the hypervisor is the supervisor of the supervisors, with hyper- used as a stronger variant of super-. The term dates to circa 1970; in the earlier CP/CMS (1967) system, the term Control Program was used instead.

There are 2 types of hypervisors, depending on where they sit in relation to the underling hardware.

Type-1, native or bare-metal hypervisors

These hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. For this reason, they are sometimes called bare metal hypervisors. The first hypervisors, which IBM developed in the 1960's, were native hypervisors. These included the test software SIMMON and the CP/CMS operating system (the predecessor of IBM's z/VM). Modern equivalents include AntsleOS, Microsoft Hyper-V and Xbox One system software, Nutanix AHV, XCP-ng, Oracle VM Server for SPARC, Oracle VM Server for x86, POWER Hypervisor and VMware ESXi (formerly ESX) and Xen.

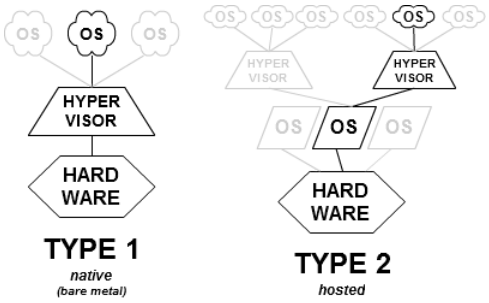


Figure **10.8.1**: Type-1 and type-2 hypervisors ("Hyperviseur.png" by Scsami, Wikimedia Commonss is licensed under CC BY-SA 3.0)

Type-2 or hosted hypervisors

These hypervisors run on a conventional operating system (OS) just as other computer programs do. A guest operating system runs as a process on the host. Type-2 hypervisors abstract guest operating systems from the host operating system. Parallels Desktop for Mac, QEMU, VirtualBox, VMware Player and VMware Workstation are examples of type-2 hypervisors.

Virtual Machine Templates

A virtual machine template is a master copy of a specific virtual machine configuration. A template is usually set up with the guest OS, whatever applications are desired, and a specific VM hardware configuration. Virtual machine templates are used when you need to deploy several VMs and ensure that they are all consistent with one another.

Oftentimes a virtual machine can be converted to a template, for example if you have a master image of a frequently deployed server OS, such as Ubuntu Linux Server. This virtual machine can be customized by adding specific tools to the OS, configuring the disk space and memory required to form a standard build, and then all future Ubuntu Server installations can be deployed from this virtual machine template.

It is common to find various templates for situations such as setting up a software development environment for a group; all of the virtual machines would have the same IDE tools installed, and all of the desktop tools necessary for the users.

Adapted from:
"Virtualization" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Hypervisor" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-E.8: The Cloud and Virtualization is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-E.9: Cloud Bootstrapping and Storage

Booting with VMs

We have discussed the booting of Linux. However, booting in the virtual world is a bit different. In the world of virtual machines there are several different ways to control the boot. Most Linux VMs use either cloud-init; Anaconda or Kickstart.

When an instance configured for **Cloud-init** boots up and the service (actually, four services in systemd implementations to handle dependencies during the boot process) starts, it checks its configuration for a data source to determine what type of cloud it is running in. Each major cloud provider has a data source configuration that tells the instance where and how to retrieve configuration information. The instance then uses the data source information to retrieve configuration information provided by the cloud provider, such as networking information and instance-identification information, and configuration data provided by the customer, such as authorized keys to be copied, user accounts to be created, and many other possible tasks.

Anaconda is the installation program used by Fedora, Red Hat Enterprise Linux and some other distributions. Anaconda is a fairly sophisticated installer. It supports installation from local and remote sources.

During installation, a target computer's hardware is identified and configured, and the appropriate file systems for the system's architecture are created. Finally, Anaconda allows the user to install the operating system software on the target computer. Anaconda can also upgrade existing installations of earlier versions of the same distribution. After the installation is complete, you can reboot into your installed system and continue doing customization using initial setup.

Kickstart installations offer a means to automate the installation process, either partially or fully. Kickstart files contain answers to all questions normally asked by the installation program, such as what time zone you want the system to use, how the drives should be partitioned, or which packages should be installed. Providing a prepared Kickstart file when the installation begins therefore allows you to perform the installation automatically, without need for any intervention from the user. This is especially useful when deploying CentOS on a large number of systems at once.

Kickstart files can be kept on a single server system and read by individual computers during the installation. This installation method can support the use of a single Kickstart file to install CentOS on multiple machines, making it ideal for network and system administrators.

All Kickstart scripts and the log files of their execution are stored in the /tmp directory to assist with debugging installation failures.

Virtual System Storage

When working with virtual systems a virtual disk is simply a file or set of files that appears as a physical disk drive to a guest operating system. These files can be on the host machine or on a remote computer connected via a network. When a virtual machine is configured with a virtual disk, you can install a new operating system onto the virtual disk without repartitioning a physical disk or rebooting the host.

In most virtual machines the actual files used by the virtual disk start out small and grow to their maximum size as needed. The main advantage of this approach is the smaller file size. Smaller files require less storage space and are easier to move in the virtual machines to a new location.

Thin and Thick Provisioning

In computing, thin provisioning involves using virtualization technology to give the appearance of having more physical resources than are actually available. If a system always has enough resource to simultaneously support all the virtualized resources, then it is not thin provisioned. The term thin provisioning is applied to disk layer in this article, but could refer to an allocation scheme for any resource. For example, real memory in a computer is typically thin-provisioned to running tasks with some form of address translation technology doing the virtualization. Each task acts as if it has real memory allocated. The sum of the allocated virtual memory assigned to tasks typically exceeds the total of real memory.

The efficiency of thin or thick/fat provisioning is a function of the use case, not of the technology. Thick provisioning is typically more efficient when the amount of resource used very closely approximates to the amount of resource allocated. Thin provisioning

offers more efficiency where the amount of resource used is much smaller than allocated, so that the benefit of providing only the resource needed exceeds the cost of the virtualization technology used.

Just-in-time allocation differs from thin provisioning. Most file systems back files just-in-time but are not thin provisioned. Over-allocation also differs from thin provisioning; resources can be over-allocated/oversubscribed without using virtualization technology; for example overselling seats on a flight without allocating actual seats at time of sale, avoiding having each consumer having a claim on a specific seat number.

Persistent Volumes

Managing storage is a distinct problem from managing compute instances. The PersistentVolume subsystem provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed.

A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plugins like Volumes but have a lifecycle independent of any individual Pod that uses the PV. This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.

Blob and Block Storage

A Binary Large OBject (BLOB) is a collection of binary data stored as a single entity in a database management system. Blobs are typically images, audio or other multimedia objects, though sometimes binary executable code is stored as a blob. Database support for blobs is not universal. Often multimedia files are stored in this manner because they are quite large.

The data type and definition was introduced to describe data not originally defined in traditional computer database systems, particularly because it was too large to store practically at the time the field of database systems was first being defined in the 1970's and 1980's. The data type became practical when disk space became cheap. The term is used in NoSQL databases, especially in Key-value store databases such as Redis.

In computing data storage, a block, sometimes called a physical record, is a sequence of bytes or bits usually containing some whole number of records having a maximum length; a block size. Data thus structured are said to be blocked. The process of putting data into blocks is called blocking, while deblocking is the process of extracting data from blocks. Blocked data is normally stored in a data buffer, and read or written a whole block at a time. Blocking reduces the overhead and speeds up the handling of the data-stream.

Adapted from:
"Anaconda" by Multiple contributors, The Fedora Project is licensed under CC BY-SA 3.0
"How Cloud-init can be used for your Raspberry Pi homelab" by Chris Collins, OpenSource.com is licensed under CC BY-SA 4.0
"Kickstart Installations" by Multiple Contributors, Centos Installation Guide is licensed under CC BY-SA 3.0
"Thin provisioning" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Persistent Volumes" by Unknown Contributors, Kubernetes is licensed under CC BY 4.0
"Block (data storage)" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Binary large object" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-E.9: Cloud Bootstrapping and Storage is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-E.10: Configuring Networks and Virtualization Tools

Networking Virtual Machines

Networking in virtual machines uses the concept of a virtual network switch. A virtual network switch is constructed in software and runs on the host machine. The VMs connect to the network through the virtual switch. Depending on the configuration of the virtual switch, a VM can use an existing virtual network managed by the hypervisor, or can connect the host system's physical network interface.

Configuration	Description
Not attached	In this mode the hypervisor reports to the guest OS that a network card is present, but that there is no connection. This is as if no Ethernet cable was plugged into the card.
Network Address Translation (NAT)	If all that is needed is to browse the Web, download files, and view email inside the guest, then this default mode should be sufficient.
Bridged networking	This is for more advanced networking needs, such as network simulations and running servers in a guest. When enabled the hypervisor connects to one of the installed network cards and exchanges network packets directly.
Internal networking	This can be used to create a different kind of software-based network which is visible to selected virtual machines, but not to applications running on the host or to the outside world.

The virsh Shell

The **`virsh`** program is command line interface for managing virtual guest domains. The program can be used to create, pause, and shutdown domains. It can also be used to list current domains. Libvirt is a C toolkit that **`virsh`** is built on, and is used to interact with the virtualization capabilities of recent versions of Linux (and other OSes). It is free software available under the GNU Lesser General Public License. Virtualization of the Linux Operating System means the ability to run multiple instances of Operating Systems concurrently on a single hardware system where the basic resources are driven by a Linux instance.

Issuing the **`virsh`** command places the user in a shell with its own set of commands, some of which are shown below.

Sub-Command	Meaning
help	Prints basic help information.
list	Lists all guests.
create	Creates a guest from an XML configuration file and starts the new guest.
start	Starts an inactive guest.
reboot	Reboots a guest.
restore	Restores a previously saved guest stored in a file.
resume	Resumes a paused guest.
save	Saves the present state of a guest to a file.
shutdown	Gracefully shuts down a guest.
suspend	Pauses a guest.

Libvirt

Libvirt is a collection of software that provides a convenient way to manage virtual machines and other virtualization functionality, such as storage and network interface management. These software pieces include an API library, a daemon (libvirtd), and a command line utility (**`virsh`**).

An primary goal of libvirt is to provide a single way to manage multiple different virtualization providers/hypervisors. For example, the command '**`virsh list --all`**' can be used to list the existing virtual machines for any supported hypervisor (KVM, Xen, VMWare ESX, etc.) No need to learn the hypervisor specific tools!

Virtual Machine Managers

Virtual Machine Manager (VMM) is a software program to create Guest virtual machines on the host machine. It's also known as Hypervisor or Virtual Machine Monitor (VMM). So, basically, the Virtual Machine Manager provides a Graphical user interface. That allows the users to create, delete, manage, allocate different hardware resources from the host machine to guest operating systems also known as virtual OS.

Virtual Machine Manager (VMM) helps the administrators to provide a unified management interface. And free them from using long commands to create or manage Virtual machines on the server. You will need to do some searching to find which VMMs have the capability to run on your Linux distribution

---

10-E.10: Configuring Networks and Virtualization Tools is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-F.11: How to Troubleshoot Networking Problems

EXAM OBJECTIVES COVERED

1.3 Given a scenario, configure and verify network connection parameters.
4.1 Given a scenario, analyze system properties and remediate accordingly.

Where to Start?

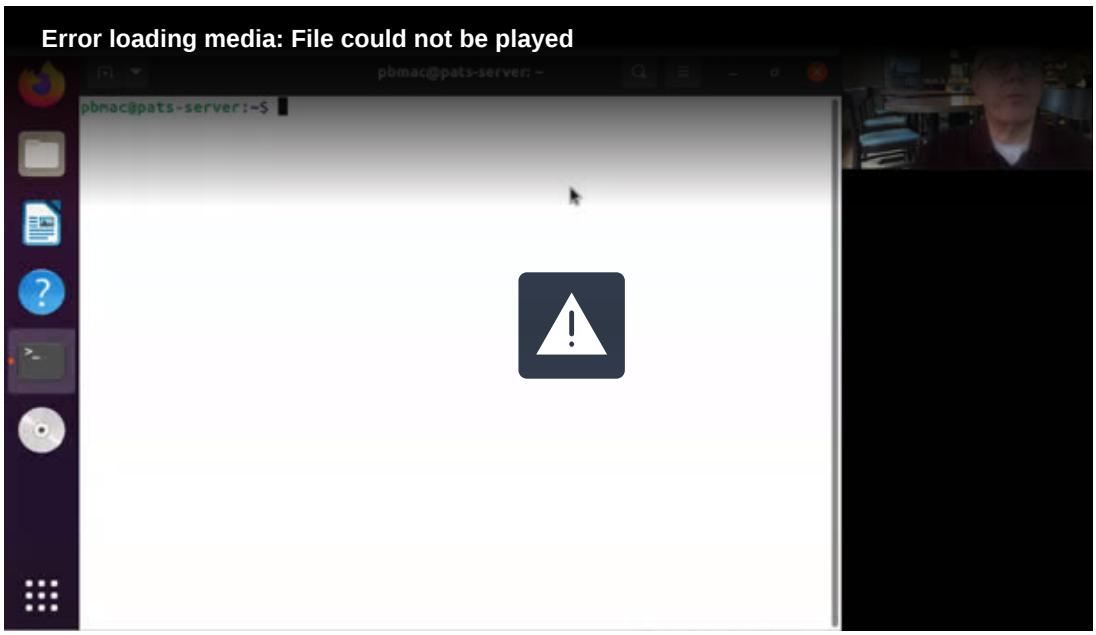
To begin with remember, keep it simple. That is a good place to start. Some simple things to check out:

- Is the device powered on - no, really - make sure.
- Is the cable plugged into the network interface - most NICs show a green light if it is seeing a network signal.
- Is the other end of the cable plugged into an appropriate place - either a wall plate, or a switch/router.
- Is the NIC in an "up" state?

If these things all check out then move on to some of the tools that have been talked about:

- Attempt to ping a host - first try one on the local network, then one out on the Internet.
- Attempt an IP based ping - if this works but the hostname doesn't ping the name resolution is not set properly.
- Test multiple systems on the network segment - do they all fail the same?

An overview of some of the network troublshooting tools that are available.



The `ping` Command

The **`ping`** command is used to check the network connectivity between host and server/host. This command takes IP address and sends a data packet to the specified address with the message “PING” and gets a response from the server/host. This time is recorded and is called latency. Fast ping low latency means faster connection. Ping is generally measured in milliseconds. Every modern operating system has this ping pre-installed.

Syntax

```
ping [ OPTIONS ] [ IP address ]
```

Command Options:

Options	Meaning
-4	Use IPv4 only.
-6	Use IPv6 only.
-a	Audible ping.
-A	Adaptive ping. Interpacket interval adapts to round-trip time, so that effectively not more than one (or more, if preload is set) unanswered probe is present in the network.
-b	Allow pinging a broadcast address.
-B	Do not allow ping to change source address of probes. The address is bound to one selected when ping starts.
-c count	Stop after sending count ECHO_REQUEST packets. With deadline option, ping waits for count ECHO_REPLY packets, until the timeout expires.
-D	Print timestamp (unix time + microseconds as in gettimeofday) before each line.
-f	Flood ping. For every ECHO_REQUEST sent a period "." is printed, while for every ECHO_REPLY received a backspace is printed.
-h	Show help.
-i interval	Wait interval seconds between sending each packet. Real number allowed with dot as a decimal separator (regardless locale setup)
-I interface	interface is either an address, or an interface name

In the following example we see four replies from the host www.google.com - notice the icmp_seq number: this is the sequence number of the reply, then at the end is the round trip time for the ping request and the reply to get back to the sender.

```
pbmac@pbmac-server $ ping www.google.com
PING www.google.com (172.217.6.68) 56(84) bytes of data.
64 bytes from sfo07s17-in-f4.1e100.net (172.217.6.68): icmp_seq=1 ttl=114 time=17.6 ms
64 bytes from sfo07s17-in-f4.1e100.net (172.217.6.68): icmp_seq=2 ttl=114 time=23.2 ms
64 bytes from sfo07s17-in-f4.1e100.net (172.217.6.68): icmp_seq=3 ttl=114 time=21.7 ms
64 bytes from sfo07s17-in-f4.1e100.net (172.217.6.68): icmp_seq=4 ttl=114 time=20.1 ms
--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 17.684/20.484/23.285/2.804 ms
```

The `traceroute` / `tracepath` Commands

The **`traceroute`** command prints the route that a packet takes to reach the specified host. This command is useful when you want to know about the route and about all the hops that a packet takes. The image below depicts how **`traceroute`** command is used to reach the Google(172.217.26.206) host from the local machine, and it also prints details about all the hops that it visits in between. The **`tracepath`** command is used to trace paths to destination discovering MTU along this path. It uses UDP port or some random port. It is similar to **`traceroute`**, but it does not require superuser privileges and has no fancy options.

Syntax:

```
traceroute / tracepath [ OPTIONS ] [ IP address ]
```

Command Options:

Options	Meaning
-4, -6	Explicitly force IPv4 or IPv6 tracerouting. By default, the program will try to resolve the name given, and choose the appropriate protocol automatically. If resolving a host name returns both IPv4 and IPv6 addresses, traceroute will use IPv4.
-I, --icmp	Use ICMP ECHO for probes
-T, --tcp	Use TCP SYN for probes
-d, --debug	Enable socket level debugging (when the Linux kernel supports it)
-f first_ttl, --first=first_ttl	Specifies with what TTL to start. Defaults to 1.
-g gateway, --gateway=gateway	Tells traceroute to add an IP source routing option to the outgoing packet that tells the network to route the packet through the specified gateway
-i interface, --interface=interface	Specifies the interface through which traceroute should send packets. By default, the interface is selected according to the routing table.
-m max_ttl, --max-hops=max_ttl	Specifies the maximum number of hops (max time-to-live value) traceroute will probe. The default is 30.

Here is an example of traceroute output. It shows the hostname/IP address and then three round trip time (RTT) for the packet to reach that point and return to your computer. This is listed in milliseconds. There are three columns because the traceroute sends three separate signal packets. This is to display consistency, or a lack thereof, in the route that is being traced.

```
pbmac@pbmac-server $ traceroute www.deltacollege.edu
traceroute to www.deltacollege.edu (52.36.131.229), 30 hops max, 60 byte packets
 1  _gateway (192.168.1.1)  3.411 ms  3.405 ms  3.395 ms
 2  96.120.86.49 (96.120.86.49)  12.338 ms  12.339 ms  12.330 ms
 3  ae-252-1216-rur02.park.ca.ccal.comcast.net (96.110.146.133)  12.974 ms  12.965 ms  12.953 ms
 4  ae-30-ar01.sacramento.ca.ccal.comcast.net (162.151.40.153)  13.836 ms  13.827 ms  13.816 ms
```

Adapted from:
"PING Command in Linux with examples" by him0000, Geeks for Geeks is licensed under CC BY-SA 4.0
"traceroute command in Linux with Examples" by codstar, Geeks for Geeks is licensed under CC BY-SA 4.0

10-F.11: How to Troubleshoot Networking Problems is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-F.11.1: Network Troubleshooting Commands - netstat / ss

The `netstat` Command

The **`netstat`** command is used to check the status of ports, whether they are open, closed, or waiting and masquerade connections. Network Statistic (netstat) command displays connection information, routing table information, etc.

Syntax:

```
netstat [ OPTIONS ]
```

Command Options:

Options	Option Meaning
--verbose , -v	Tells the user what is going on by being verbose. Especially prints some useful information about unconfigured address families.
-c, --continuous	This will cause **`netstat`** to print the selected information every second continuously.
--interfaces=iface , -I=iface , -i	Displays a table of all network interfaces, or the specified iface.
--listening, -l	Shows only listening sockets. (These are omitted by default.)
--route, -r	Displays the kernel routing tables.
--statistics, -s	Displays summary statistics for each protocol.

The output of **`netstat`** obviously depends on the options given on the command line. The following example shows the default:

```
pbmac@pbmac-server $ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 pbmac-server.loca:44042 205.180.87.178:https    ESTABLISHED
tcp        0      0 pbmac-server.loca:57930 netlab.cisconetla:https ESTABLISHED
tcp        0      0 pbmac-server.loca:47834 sfo03s01-in-f206.:https ESTABLISHED
tcp        0      0 pbmac-server.loca:58120 64.58.232.179:https     ESTABLISHED
tcp        0      0 pbmac-server.loca:59788 sfo07s16-in-f66.1:https ESTABLISHED
```

`Netstat` provides statistics for the following:

- Proto – The name of the protocol (TCP or UDP).
- Recv-Q/Send-Q - The number of bytes waiting to be received or sent over this connection.

- Local address – The IP address of the local computer and the port number being used. The name of the local computer that corresponds to the IP address and the name of the port is shown unless the -n parameter is specified. An asterisk (*) is shown for the host if the server is listening on all interfaces. If the port is not yet established, the port number is shown as an asterisk.
- Foreign address – The IP address and port number of the remote computer to which the socket is connected. The names that corresponds to the IP address and the port are shown unless the -n parameter is specified. If the port is not yet established, the port number is shown as an asterisk (*).
- State – Indicates the state of a TCP connection. The possible states are as follows: CLOSE_WAIT, CLOSED, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, LAST_ACK, LISTEN, SYN_RECEIVED, SYN_SEND, and TIME_WAIT. For more information about the states of a TCP connection, see RFC 793.

The `ss` Command

The **`ss`** command is a new tool used to dump socket statistics; it displays information a bit like **`netstat`** - although **`ss`** is simpler and faster. The **`ss`** command can also display additional TCP and more state information than most other tools.

The **`ss`** command displays stats for things like PACKET, TCP, UDP, DCCP, RAW, and Unix domain sockets. This utility displays very detailed information about how a Linux machine is communicating with other machines, networks, and services; and information about network connections, networking protocol statistics, and Linux socket connections. This information makes it much easier to troubleshoot various networking issues.

Syntax:

```
ss [ OPTIONS ]
```

Command Options:

Options	**Option Meaning**
-a, --all	Display all sockets.
-l, --listening	Display listening sockets.
-o, --options	Show timer information.
-e, --extended	Show detailed socket information.
-m, --memory	Show socket memory usage.
-p, --processes	Show process using socket.
-i, --info	Show internal TCP information.

The output is similar to **`netstat`**.

```
pbmac@pbmac-server $ ss -4 state listening
Netid   Recv-Q    Send-Q        Local Address:Port         Peer Address:Port
tcp     0         128           127.0.0.53%lo:domain            0.0.0.0:*
tcp     0         128                 0.0.0.0:ssh               0.0.0.0:*
tcp     0         5                 127.0.0.1:ipp               0.0.0.0:*
tcp     0         100               127.0.0.1:smtp              0.0.0.0:*
tcp     0         5                   0.0.0.0:902               0.0.0.0:*
```

```
tcp      0          128                127.0.0.1:46697            0.0.0.0:*
tcp      0          128                127.0.0.1:6379             0.0.0.0:*
```

Adapted from:
"netstat" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-F.11.1: Network Troubleshooting Commands - netstat / ss is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-F.11.2: Network Troubleshooting Commands - dig / nslookup / host

The **dig** Command

The **dig** command stands for Domain Information Groper. It is a network administration command-line tool for querying the Domain Name System (DNS). The **dig** command is useful for network troubleshooting and for educational purposes. It can operate based on command line option and flag arguments, or in batch mode by reading requests from an operating system file. When a specific name server is not specified in the command invocation, it uses the operating system's default resolver, usually configured in the file resolv.conf. Without any arguments it queries the DNS root zone.

Syntax:

```
dig [ OPTIONS ]
```

dig is very easy to use, and works well without any options. The list of options is quite long, feel free to consult the man page for this command.

The following example is a simple lookup of the IP address for the server specified on the command line - deltacollege.edu. The lines that begin with a semi-colon provide comments about the output.

```
pbmac@pbmac-server $ dig deltacollege.edu
; <<>> DiG 9.11.3-1ubuntu1.11-Ubuntu <<>> deltacollege.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64205
;; flags: qr rd ra; QUERY: 1, Answer: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;deltacollege.edu.        IN    A

;; Answer SECTION:
deltacollege.edu.    3599    IN    A    52.36.131.229
```

The **nslookup** Command

nslookup is a network administration command-line tool available in many computer operating systems for querying the Domain Name System (DNS) to obtain domain name or IP address mapping, or other DNS records. The name "**nslookup**" means "name server lookup."

The command does not use the operating system's local Domain Name System resolver library to perform its queries, and thus may behave differently from **dig**, which it does. Additionally, vendor-provided versions may include output of other sources of name information, such as host files and Network Information Service. Some behaviors of **nslookup** may be modified by the contents of resolv.conf.

Syntax:

```
nslookup [ OPTION ] [name | -] [server]
```

It is possible to use **nslookup** in a command line mode, or an interactive mode. In the first example the command prints output and returns to the system prompt. In the second example issuing **nslookup** alone takes the user to a command prompt where command specific input can be entered.

```
pbmac@pbmac-server $ nslookup deltacollege.edu
Server:         8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
Name:    deltacollege.edu
Address: 52.36.131.229

pbmac@pbmac-server $ nslookup
> deltacollege.edu
Server:         8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
Name:    deltacollege.edu
Address: 52.36.131.229
>
```

The **host** Command

The **host** command in Linux system is used for DNS (Domain Name System) lookup. It is used to find the IP address of a particular domain name, or if you want to find out the domain name of a particular IP address the **host** command becomes handy. You can also find more specific details of a domain by specifying the corresponding option along with the domain name.

Syntax:

```
host [ OPTIONS ] hostname
```

Command Options

Options	Meaning
-4	This option specifies that only IPv4 should be used for query transport. See also the -6 option.
-6	This option specifies that only IPv6 should be used for query transport. See also the -4 option.
-a	The -a ("all") option is normally equivalent to -v -t ANY. It also affects the behavior of the -l list zone option.
-A	The -A ("almost all") option is equivalent to -a, except that RRSIG, NSEC, and NSEC3 records are omitted from the output.
-c class	This option specifies the query class, which can be used to lookup HS (Hesiod) or CH (Chaosnet) class resource records. The default class is IN (Internet).
-C	This option indicates that named should check consistency, meaning that host queries the SOA records for zone name from all the listed authoritative name servers for that zone. The list of name servers is defined by the NS records that are found for the zone.
-d	This option prints debugging traces, and is equivalent to the -v verbose option.
-l	This option tells named` to list the zone, meaning the ``host command performs a zone transfer of zone name and prints out the NS, PTR, and address records (A/AAAA).
-N ndots	This option specifies the number of dots (ndots) that have to be in name for it to be considered absolute. The default value is that defined using the ndots statement in /etc/resolv.conf, or 1 if no ndots statement is present. Names with fewer dots are interpreted as relative names, and are searched for in the domains listed in the search or domain directive in /etc/resolv.conf.
-p port	This option specifies the port to query on the server. The default is 53.
-r	This option specifies a non-recursive query; setting this option clears the RD (recursion desired) bit in the query. This means that the name server receiving the query does not attempt to resolve name. The -r option enables host to mimic the behavior of a name server by making non-recursive queries, and expecting to receive answers to those queries that can be referrals to other name servers.
-R number	This option specifies the number of retries for UDP queries. If number is negative or zero, the number of retries is silently set to 1. The default value is 1, or the value of the attempts option in /etc/resolv.conf, if set.
-s	This option tells named not to send the query to the next nameserver if any server responds with a SERVFAIL response, which is the reverse of normal stub resolver behavior.
-t type	This option specifies the query type. The type argument can be any recognized query type: CNAME, NS, SOA, TXT, DNSKEY, AXFR, etc.

Without a hostname the command will print out the command options.

Using the -v option generates a very verbose output - gives a lot of information.

```
pbmac@pbmac-server $ host -v deltacollege.edu
Trying "deltacollege.edu"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53400
;; flags: qr rd ra; QUERY: 1, Answer: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;deltacollege.edu.        IN    A

;; Answer SECTION:
deltacollege.edu.    3599    IN    A    52.36.131.229
Received 50 bytes from 8.8.8.8#53 in 164 ms
Trying "deltacollege.edu"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4462
;; flags: qr rd ra; QUERY: 1, Answer: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;deltacollege.edu.        IN    AAAA
;; AUTHORITY SECTION:
deltacollege.edu.    1799    IN    SOA    sdc-ns2.deltacampus.net. hostmaster.deltacol

Received 104 bytes from 8.8.8.8#53 in 187 ms
Trying "deltacollege.edu"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41683
;; flags: qr rd ra; QUERY: 1, Answer: 4, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;deltacollege.edu.        IN    MX

;; Answer SECTION:
deltacollege.edu.    3599    IN    MX    20 mailin.deltacollege.edu.
deltacollege.edu.    299    IN    MX    15 mailin1.deltacollege.edu.
deltacollege.edu.    3599    IN    MX    5 deltacollege-edu.mail.protection.outlook.co
deltacollege.edu.    3599    IN    MX    15 mailin3.deltacollege.edu.
Received 165 bytes from 8.8.8.8#53 in 133 ms
```

Adapted from:
"dig (command)" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"host command in Linux with examples" by sethusubramanian, Geeks for Geeks is licensed under CC BY-SA 4.0

10-F.11.2: Network Troubleshooting Commands - dig / nslookup / host is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-F.11.3: Network Troubleshooting Commands - ip / route / nmap

The `ip` Command

The **`ip`** command comes from the net-tools which is used for performing several network administration tasks. IP stands for Internet Protocol. This command is used to show or manipulate routing, devices, and tunnels. It is similar to **`ifconfig`** but it is much more powerful with more functions and facilities attached to it. **`ifconfig`** is one of the deprecated Linux commands and has not been maintained for many years. The **`ip`** command is used to perform several tasks, like assigning an address to a network interface or configuring network interface parameters.

It can perform several other tasks like configuring and modifying the default and static routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, or assigning, deleting and setting up IP addresses and routes.

Syntax:

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
```

Command Options

Options	Meaning
-V, -Version	Print the version of the ip utility and exit.
-h, -human, -human-readable	output statistics with human readable values followed by suffix.
-b, -batch <FILENAME>	Read commands from provided file or standard input and invoke them. First failure will cause termination of ip.
-s, -stats, -statistics	Output more information. If the option appears twice or more, the amount of information increases. As a rule, the information is statistics or some time values.
-d, -details	Output more detailed information.
-f, -family <FAMILY>	Specifies the protocol family to use.
-4	shortcut for -family inet.
-6	shortcut for -family inet6.
-B	shortcut for -family bridge.
-M	shortcut for -family mpls.
-0	shortcut for -family link.
-r, -resolve	use the system's name resolver to print DNS names instead of host addresses.
-N, -Numeric	Print the number of protocol, scope, dsfield, etc directly instead of converting it to human readable name.
-t, -timestamp	display current time when using monitor option.

With the **`ip`** command you can:

- Assign an IP address to a specific interface
- Check the IP address on any interface
- Remove an IP address from an interface

- Enable an IP address on an interface
- Disable a network interface
- Check the routing table
- Add a static route
- Remove a static route
- Add a persistent static route

The `route` Command

The **`route`** command enables users to work with the IP/kernel routing table. It is mainly used to set up static routes to specific hosts or networks via an interface. It can also show or update the IP/kernel routing table. A router, which normally does the routing in a network, is a device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet using information in its routing table or a routing policy to direct the packet to the next network on its journey.

Syntax:

```
route [ OPTIONS ]
```

Command Options:

Options	Meaning
-A family	use the specified address family (eg `inet'). Use route --help for a full list. You can use -6 as an alias for --inet6 and -4 as an alias for -A inet
-F	operate on the kernel's FIB (Forwarding Information Base) routing table. This is the default.
-C	operate on the kernel's routing cache.
-v	select verbose operation.
-n	show numerical addresses instead of trying to determine symbolic host names. This is useful if you are trying to determine why the route to your nameserver has vanished.
-e	use netstat(8)-format for displaying the routing table. -ee will generate a very long line with all parameters from the routing table.
del	delete a route.
add	add a new route.
target	the destination network or host. You can provide an addresses or symbolic network or host name. Optionally you can use /prefixlen notation instead of using the netmask option.
-net	the target is a network.
-host	the target is a host.
netmask NM	when adding a network route, the netmask to be used.
gw GW	route packets via a gateway. NOTE: The specified gateway must be reachable first. This usually means that you have to set up a static route to the gateway beforehand. If you specify the address of one of your local interfaces, it will be used to decide about the interface to which the packets should be routed to. This is a BSDism compatibility hack.
metric M	set the metric field in the routing table (used by routing daemons) to M. If this option is not specified the metric for inet6 (IPv6) address family defaults to '1', for inet (IPv4) it defaults to '0'. You should always specify an explicit metric value to not rely on those defaults - they also differ from iproute2.
mss M	sets MTU (Maximum Transmission Unit) of the route to M bytes. Note that the current implementation of the route command does not allow the option to set the Maximum Segment Size (MSS).
window W	set the TCP window size for connections over this route to W bytes. This is typically only used on AX.25 networks and with drivers unable to handle back to back frames.

The following example shows the routing table - the gateway device, the local link and the network this host sits on:

```
pbmac@pbmac-server $ route
Kernel IP routing table
```

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp0s25
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp0s25
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 enp0s25

To add a default gateway an IP address must be used
pbmac@pbmac-server $  route add default gw 169.254.0.0
```

The **nmap** Command

Nmap (Network Mapper) is a free and open-source network scanner used to discover hosts and services on a computer network by sending packets and analyzing the responses.

Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. **Nmap** can adapt to network conditions including latency and congestion during a scan.
Nmap features include:

- Host discovery – Identifying hosts on a network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular port open.
- Port scanning – Enumerating the open ports on target hosts.
- Version detection – Interrogating network services on remote devices to determine application name and version number.
- OS detection – Determining the operating system and hardware characteristics of network devices.
- Scriptable interaction with the target – using Nmap Scripting Engine (NSE) and Lua programming language.
- **Nmap** can provide further information on targets, including reverse DNS names, device types, and MAC addresses.

Typical uses of **Nmap**:

- Auditing the security of a device or firewall by identifying the network connections which can be made to, or through it.
- Identifying open ports on a target host in preparation for auditing.
- Network inventory, network mapping, maintenance and asset management.
- Auditing the security of a network by identifying new servers.
- Generating traffic to hosts on a network, response analysis and response time measurement.
- Finding and exploiting vulnerabilities in a network.
- DNS queries and subdomain search.

Syntax:

```
nmap [ OPTIONS ] [ TARGET ]
```

The options summary is presented when the command is run with no options and no target.

The target can be a hostname or an IP address.

The **nmap** utility offers various methods to scan a system. In the example below, a scan is performed using hostname www.deltacollege.edu, to find out all open ports and services on the system. As usual, if the -v option is used the output is much more verbose, not necessarily providing any additional information, but providing it in a very clear manner.

```
pbmac@pbmac-server $ nmap www.deltacollege.edu

Starting Nmap 7.60 ( https://nmap.org ) at 2020-10-20 13:33 PDT
Nmap scan report for www.deltacollege.edu (52.36.131.229)
Host is up (0.040s latency).
rDNS record for 52.36.131.229: ec2-52-36-131-229.us-west-2.compute.amazonaws.com
```

```
Not shown: 997 filtered ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 5.00 seconds
```

Adapted from:
"ip command in Linux with examples" by raghavmangal22, Geeks for Geeks is licensed under CC BY-SA 4.0
"Nmap" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-F.11.3: Network Troubleshooting Commands - ip / route / nmap is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-F.11.4: Network Troubleshooting Commands - wireshark

Wireshark

Wireshark is a free and open-source packet analyzer used for network troubleshooting, analysis, software and communications protocol development, and education. It is very similar to **tcpdump**, but has a graphical front-end, plus some integrated sorting and filtering options.

Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface, including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering

Since **Wireshark** is a data capturing program that "understands" the structure (encapsulation) of different networking protocols, it can parse and display the fields along with their meanings as specified by different networking protocols. **Wireshark** uses pcap to capture packets, so it can only capture packets on the types of networks that pcap supports.

- Data can be captured "from the wire" from a live network connection or read from a file of already-captured packets.
- Live data can be read from different types of networks, including Ethernet, IEEE 802.11, PPP, and loopback.
- Captured network data can be browsed via a GUI, or via the terminal (command line) version of the utility, TShark.
- Captured files can be programmatically edited or converted via command-line switches to the "editcap" program.
- Data display can be refined using a display filter.

The tcpdump Command

The **tcpdump** command line is a utility that allows you to capture and analyze network traffic going through your system. It is often used to help troubleshoot network issues, as well as a security tool.

A powerful and versatile tool that includes many options and filters, **tcpdump** can be used in a variety of cases. Since it's a command line tool, it is ideal to run in remote servers or devices for which a GUI is not available, to collect data that can be analyzed later. It can also be launched in the background or as a scheduled job using tools like cron.

Syntax:

```
tcpdump [ OPTIONS ]
```

There are numerous options, and a full listing can be found on the **tcpdump** man page. A few are listed below.

Command Options:

Options	Meaning
-c count	Exit after receiving the specified number of packets.
-i interface	Listen on the specified interface. If unspecified, **tcpdump** searches the system interface list for the lowest numbered, configured up interface (excluding loopback).
-n	Don't convert host addresses to names. This can be used to avoid DNS lookups.
-v	When parsing and printing, produce (slightly more) verbose output.

Here is some typical output:

```
pbmac@pbmac-server $ sudo tcpdump -i any -c 5
[sudo] password for pbmac:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
14:26:13.216611 IP 192.168.1.8 > 224.0.0.251: igmp v2 report 224.0.0.251
14:26:13.225365 IP pbmac-server.local.34227 > dns.google.domain: 41755+ PTR? 251.0.0.2
14:26:13.250131 IP dns.google.domain > pbmac-server.local.34227: 41755 NXDomain 0/1/0
14:26:13.361473 IP6 PbmacUbuntu.mdns > ff02::fb.mdns: 0 PTR (QM)? 251.0.0.224.in-addr
14:26:13.361526 IP pbmac-server.local.mdns > 224.0.0.251.mdns: 0 PTR (QM)? 251.0.0.224
5 packets captured
412 packets received by filter
402 packets dropped by kernel
```

The `netcat` Command

`netcat` (often abbreviated to nc) is a computer networking utility for reading from and writing to network connections using TCP or UDP. The command is designed to be a dependable back-end that can be used directly or easily driven by other programs and scripts. At the same time, it is a feature-rich network debugging and investigation tool, since it can produce almost any kind of connection its user could need and has a number of built-in capabilities.

Syntax:

```
netcat [ OPTIONS ]
```

Command Options:

Options	Meaning
-e filename	specify filename to exec after connect (use with caution). See the -c option for enhanced functionality.
-g gateway	source-routing hop point[s], up to 8
-G num	source-routing pointer: 4, 8, 12, ...
-h	display help
-i secs	delay interval for lines sent, ports scanned
-l	listen mode, for inbound connects
-n	numeric-only IP addresses, no DNS
-o file	hex dump of traffic
-p port	local port number (port numbers can be individual or ranges: lo-hi [inclusive])
-q seconds	after EOF on stdin, wait the specified number of seconds and then quit. If seconds is negative, wait forever.
-b	allow UDP broadcasts
-r	randomize local and remote ports
-s addr	local source address

Some of **`netcat's`** features include:

- Outbound or inbound connections, TCP or UDP, to or from any ports
- Full DNS forward/reverse checking, with appropriate warnings
- Ability to use any local source port
- Ability to use any locally configured network source address
- Built-in port-scanning capabilities, with randomization
- Built-in loose source-routing capability
- Can read command line arguments from standard input
- Slow-send mode, one line every N seconds
- Hex dump of transmitted and received data
- Optional ability to let another program service establish connections
- Optional telnet-options responder

Some examples:

```
Opening a raw connection to port 25
pbmac@pbmac-server $ nc mail.server.net 25

Performing an HTTP request
pbmac@pbmac-server $ printf "GET /index.html HTTP/1.0\r\nHost: info.cern.ch\r\n\r\n"
The full response (including HTTP headers) will be dumped to standard output.

To transfer files between servers:
Destination server - starts the nc process listening to port 7999 and accept the input
pbmac@pbmac-server $ nc -l -p 7999 > testInput.txt
Then of the source server - netcat sends the fileoutputTest.txt to the specified IP ad
pbmac@pbmac-server $ nc 192.168.1.99 7555 < outputTest.txt

To use netcat to scan all ports up to 750 by issuing this command:
pbmac@pbmac-server $ netcat -z -v domain.com 1-750
Along with the -z option, the -v option to tell netcat to provide more verbose informa
```

Adapted from:
"Wireshark" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"An introduction to using tcpdump at the Linux command line" by Ricardo Gerardi, OpenSource.com is licensed under CC BY-SA 4.0
"netcat" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-F.11.4: Network Troubleshooting Commands - wireshark is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-F.11.5: Network Troubleshooting Commands - iftop / iperf / mtr

The `iftop` Command

`iftop` monitors network traffic and displays a table of current bandwidth usage. An interface may be specified or, if not, it will listen on the first interface it finds which looks like an external interface. **`iftop`** must be run with sufficient permissions to monitor all network traffic; on most systems this means that it must be run as a root user.

By default, **`iftop`** will look up hostnames associated with addresses and count all IP packets that pass through the filter. Hostname look-up can add substantial traffic, in and of itself, and may result in an inaccurate display of network traffic.

Syntax:

```
iftop [ OPTIONS ]
```

Command Options:

Options	Meaning
-h	Print a summary of usage.
-n	Don't do hostname lookups.
-N	Do not resolve port number to service names
-p	Run in promiscuous mode, so that traffic which does not pass directly through the specified interface is also counted.
-P	Turn on port display.
-l	Display and count datagrams addressed to or from link-local IPv6 addresses. The default is not to display that address category.
-b	Don't display bar graphs of traffic.
-B	Display bandwidth rates in bytes/sec rather than bits/sec.
-i interface	Listen to packets on interface.

In the example, if we look on the top address:

```
                19.1Mb       38.1Mb          57.2Mb              76.3Mb   95.4Mb
|_______________|____________|_______________|___________________|_____________
pbmacrserver.local     => tglrden.search.spotxchang     0b      2.01Kb      515b
                       <=                               0b      1.39Kb      355b
pbmacrserver.local     => ec2713756754713.usrwestrl     0b      1.88Kb      491b
                       <=                               0b       310b        90b
224.8.8.251            => Chromecast.local              0b         0b         0b
                       <=                               0b      1.85Kb      481b
pbmacrserver.local     => 199.244.58.41                 0b      1.81Kb    1.81Kb
                       <=                               0b       342b       342b
pbmacrserver.local     => pgrin7f189.1e188.net       2.78Kb      552b       140b
                       <=                            1.58Kb      386b        98b
pbmacrserver.local     => 5f08358871n7f35.1e188.net     0b       385b        87b
                       <=                               0b       534b       144b
```

```
224.8.8.251             => 192.168.1.21                  0b        0b        0b
                        <=                               0b      611b      280b
pbmacrserver.local      => Chromecast.local              0b      341b      341b
                        <=                               0b      250b      250b
pbmacrserver.local      => dns.google                    0b      111b       50b
                        <=                               0b      195b      lllb

------------------------------------------------------------------------------
TX:          cum:  443KB  peak: 25.6Kb rates:         3.25Kb    6.77Kb.    4.24Kb
RX'               1.84MB        28.4Kb.              1.82Kb    6.13Kb     4.32Kb
TOTAL:            2.27MB        35.5Kb               5.87Kb    12.9Kb     8.56Kb
```

- pbmac-server.local in the preceding **2 seconds** is sent 0KB of data to the host tfl-den.search.spotxchang
- In the preceding **10 seconds** pbmac-server.local is sent 2.01Kb data
- In the preceding **40 seconds** pbmac-server.local515b data

At the same time :

- tfl-den.search.spotxchang in the preceding **2 seconds** is sent 0Kb data to pbmac-server.local
- In the preceding **10 seconds** it sent 1.39Kb data
- In the preceding **40 seconds** it sent 355b data

At the bottom line, **`iftop`** show us the results:

- Cumulative TX and RX kilobytes transmitted
- Peak TX and RX data over the last 40 seconds
- Total transfer rates averaged over 2 seconds, 10 seconds and 40 seconds

The `iperf` Command

`Iperf` is a widely used tool for network performance measurement and tuning. It is significant as a cross-platform tool that can produce standardized performance measurements for any network. **`Iperf`** has client and server functionality, and can create data streams to measure the throughput between the two ends in one or both directions. Typical **`iperf`** output contains a time-stamped report of the amount of data transferred and the throughput measured.

The data streams can be either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP):

- UDP: When used for testing UDP capacity, **`iperf`** allows the user to specify the datagram size and provides results for the datagram throughput and the packet loss.
- TCP: When used for testing TCP capacity, **`iperf`** measures the throughput of the payload. **`Iperf`** uses 1024 × 1024 for mebibytes and 1000 × 1000 for megabytes.

Syntax:

```
iperf { -c | -s } [ OPTIONS ]
```

Command Options:

Options	Meaning
-f, --format	[kmKM] format to report: Kbits, Mbits, KBytes, MBytes
-h, --help	print a help synopsis
-i, --interva	l n pause n seconds between periodic bandwidth reports
-l, --len n[KM]	set length read/write buffer to n (default 8 KB)
-m, --print_mss	print TCP maximum segment size (MTU - TCP/IP header)
-o, --output <filename>	output the report or error message to this specified file
-p, --port n	set server port to listen on/connect to to n (default 5001)
-u, --udp	use UDP rather than TCP
-w, --window n[KM]	TCP window size (socket buffer size)
-B, --bind <host>	bind to <host>, an interface or multicast address
-C, --compatibility	for use with older versions does not sent extra msgs
-M, --mss n	set TCP maximum segment size (MTU - 40 bytes)
-N, --nodelay	set TCP no delay, disabling Nagle's Algorithm
-v, --version	print version information and quit
-V, --IPv6Version	Set the domain to IPv6

The -c or -s option must be used with other OPTIONS.

The following example shows the test of the bandwidth measurements between the local server and the remote server:

```
pbmac@pbmac-server $ iperf -c www.deltacollege.edu -u 100
iperf: ignoring extra argument -- 100
------------------------------------------------------------
Client connecting to www.deltacollege.edu, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.6 port 49584 connected with 52.36.131.229 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.
```

The `mtr` Command

MTR relies on Internet Control Message Protocol (ICMP) Time Exceeded packets coming back from routers, or ICMP Echo Reply packets when the packets have hit their destination host. MTR also has a User Datagram Protocol (UDP) mode that sends UDP packets, with the time to live (TTL) field in the IP header increasing by one for each probe sent, toward the destination host. When the UDP mode is used, MTR relies on ICMP port unreachable packets when the destination is reached.

The tool is often used for network troubleshooting. By showing a list of routers traversed, and the average round-trip time as well as packet loss to each router, it allows users to identify links between two given routers responsible for certain fractions of the

overall latency or packet loss through the network. This can help identify network overuse problems.

Syntax:

```
mtr [ OPTIONS ] HOSTNAME
```

Command Options:

Options	Meaning
-h --help	Print the summary of command line argument options.
-v --version	Print the installed version of mtr.
-r --report	This option puts mtr into report mode. When in this mode, mtr will run for the number of cycles specified by the -c option, and then print statistics and exit. This mode is useful for generating statistics about network quality. Note that each running instance of mtr generates a significant amount of network traffic. Using mtr to measure the quality of your network may result in decreased network performance.
-w --report-wide	This option puts mtr into wide report mode. When in this mode, mtr will not cut hostnames in the report.
-c COUNT --report-cycles COUNT	Use this option to set the number of pings sent to determine both the machines on the network and the reliability of those machines. Each cycle lasts one second.
-s BYTES --psize BYTES PACKETSIZE	These options or a trailing PACKETSIZE on the commandline sets the packet size used for probing. It is in bytes inclusive IP and ICMP headers If set to a negative number, every iteration will use a different, random packetsize upto that number.
-n --no-dns	Use this option to force mtr to display numeric IP numbers and not try to resolve the host names.

The columns in the output are relatively self explanatory: 1) Host - the remote system; 2) Packets Loss % - percent of packets that got no ack; 3) Packets Snt - number of packets sent; 4) Last - round trip time of the last packet; 5) Avg - the average time; 6) Best - the shortest round trip time; 7) Wrst - the slowest time; and 8) StDev - the standard deviation in the ping times.

```
                              My traceroute [v8.92]

PbmacUbuntu (192.168.1.6)
Keys:  Help   Display mode     Restart statistics  Order of fields   quit
Packets Pings
Host                                Loss%   Snt   Last  Avg  Best   Wrst StDev
1. _gateway                          9.9%   663    8.7  7.1   3.2   16.1   1.6
2. 96.128.86.49                      0.0%   663   15.6 15.3  11.1   36.9   2.2
3. aer25271216rrur82.park.ca.ccal.c 8.8%   663   16.1 16.8  11.5   57.1   4.1
4. aer38rar81.sacramento.ca.ccal.co 8.8%   663   15.4 18.7  14.6   77.4   6.2
5. ber364317cs83.sunnyvale.c .ibone 8.8%   663   28.8 21.8  18.1   72.4   2.9
6. ber1312rcr12.sunnyvale.ca ibone. 29.3%  663   18.4 21.8  16.5   41.1   3.2
7. ber384rcr81.99reatoaks.ca.ibone. 8.8%   662   28.4 21.6  17.3   36.5   2.4
8. ber24117pe11.99reatoaks.ca.ibone 8.8%   662   22.1 21.2  15.4   34.4   2.1
```

```
 9. 58.288.233.46                 8.8%   662   19.8 28.6  15.1   53.9   2.5
 18. 216.239.48.18                8.8%   662   21.8 23.2  17.9  113.7   6.4
 11. 289.85.248.35                8.8%   662   19.8 28.5  17.2   38.7   2.8
 12. sfo87sl7rin7f14.1e188.net    8.8%   662   28.2 28.5  14.7   47.1   2.3
```

Adapted from:

"iftop" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"Iperf" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"MTR (software)" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-F.11.5: Network Troubleshooting Commands - iftop / iperf / mtr is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

10-F.11.6: Network Troubleshooting Commands - arp / whois

The `arp` Command

Most of the computer programs/applications use logical address (IP address) to send/receive messages; however the actual communication happens over the physical address (MAC address) i.e., from layer 2 of OSI model. So our mission is to get the destination MAC address which helps in communicating with other devices. This is where ARP comes into the picture. Its functionality is to translate IP address to physical address.

The acronym ARP stands for Address Resolution Protocol. which is one of the most important protocols of the Network layer in the OSI model.

The Address Resolution Protocol is a request-response protocol whose messages are encapsulated by a link layer protocol. It is communicated within the boundaries of a single network, never routed across inter-networking nodes. This property places ARP into the link layer of the Internet protocol suite.

Syntax:

```
arp [ OPTIONS ]
```

There are important terms associated with ARP:

1. **ARP Cache:** After resolving MAC address, the ARP sends it to the source where it stores in a table for future reference. The subsequent communications can use the MAC address from the table.
2. **ARP Cache Timeout:** Indicates the time for which the MAC address in the ARP cache can reside.
3. **ARP request:** This is nothing but broadcasting a packet over the network to validate whether we came across destination MAC address or not. The packet includes:
    1. The physical address of the sender.
    2. The IP address of the sender.
    3. The physical address of the receiver is FF:FF:FF:FF:FF:FF or 1’s.
    4. The IP address of the receiver
4. **ARP response/reply:** It is the MAC address response that the source receives from the destination which aids in further communication of the data.

An example: Two computers in an office (Computer 1 and Computer 2) are connected to each other in a local area network by Ethernet cables and network switches, with no intervening gateways or routers. Computer 1 has a packet to send to Computer 2. Through DNS, it determines that Computer 2 has the IP address 192.168.0.55.

To send the message, Computer 1 requires Computer 2's MAC address. First, Computer 1 looks in its cached ARP table to look up 192.168.0.55 to see if any records of Computer 2's MAC address, which is 00:eb:24:b2:05:ac, exist. If the MAC address is found, Computer 1 sends an Ethernet frame using the destination MAC address 00:eb:24:b2:05:ac. The frame contains Computer 2's IP packet onto the link. If the cache did not produce a result for 192.168.0.55, then Computer 1 has to send a broadcast ARP request message (destination FF:FF:FF:FF:FF:FF MAC address), which is accepted by all computers on the local network, requesting an answer for "Who has IP address 192.168.0.55?"

Computer 2 responds with an ARP response message containing its MAC and IP addresses. As part of dealing with the request, Computer 2 may insert an entry for Computer 1 into its ARP table for future use.

Computer 1 receives Computer 2's IP AND MAC address, puts them in the cache table for future reference and then can now send the packet it wanted to send in the first place.

The `whois` Command

whois (pronounced as the phrase "who is") is a query and response protocol that is widely used for querying databases that store the registered users or assignees of an Internet resource, such as a domain name, an IP address block or an autonomous system. It is

also used for a wider range of other information. The protocol stores and delivers database content in a human-readable format. The current iteration of the **whois** protocol was drafted by the Internet Society, and is documented in RFC 3912.

Syntax:

```
whois [ OPTIONS ] DOMAINNAME
```

The output from **whois** can be rather voluminous, depending on the domain being looked up. It may be that the user has to sort through some of the output to find the useful information. There are numerous web-based **whois** engines that do a good job of providing useful feedback.

```
pbmac@pbmac-server $ whois google.com
   Domain Name: GOOGLE.COM
   Registry Domain ID: 2138514_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.markmonitor.com
   Registrar URL: http://www.markmonitor.com
   Updated Date: 2019-09-09T15:39:04Z
   Creation Date: 1997-09-15T04:00:00Z
   Registry Expiry Date: 2028-09-14T04:00:00Z
   Registrar: MarkMonitor Inc.
   Registrar IANA ID: 292
   Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
   Registrar Abuse Contact Phone: +1.2083895740
```

Adapted from:
"Address Resolution Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"How Address Resolution Protocol (ARP) works?" by Vivek Reddy, Geeks for Geeks is licensed under CC BY-SA 4.0
"WHOIS" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

10-F.11.6: Network Troubleshooting Commands - arp / whois is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 11: Package Management

11-A.1: Who are the Package Managers

11-B.2: rpm / yum How To

11-C.3: dpkg / apt How To

11-D.4: Repository Configuration

11-E.5: Other Software Sources

11-F.6: Software Builds from Source

11-G.7: Software Dependency Troubleshooting

This page titled Module 11: Package Management is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

11-A.1: Who are the Package Managers

EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

Objectives of this Module

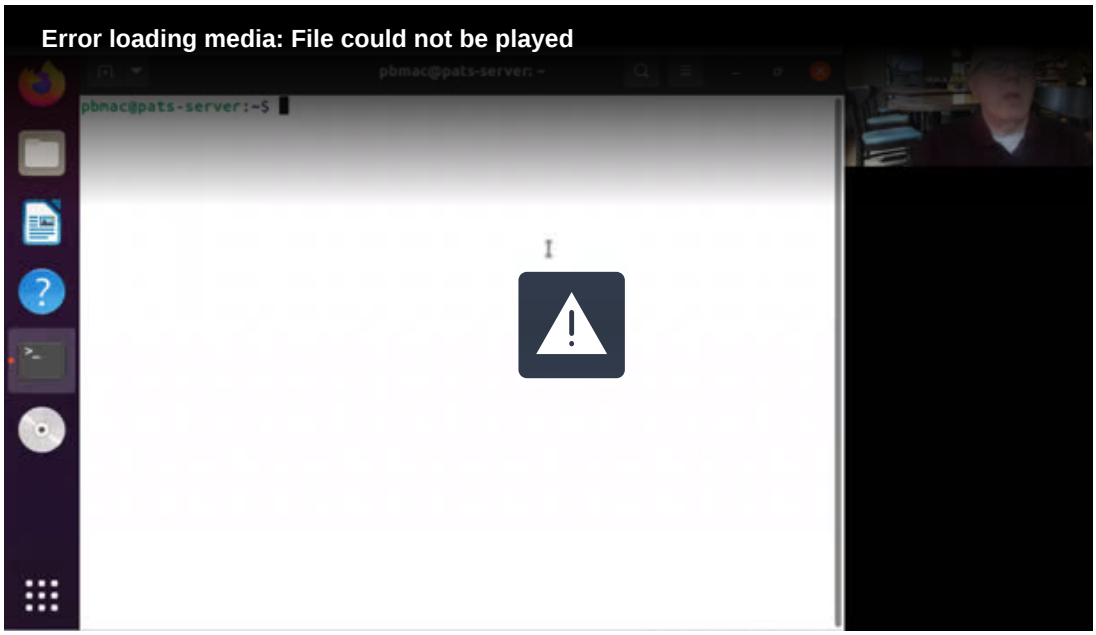
In this lesson, you will:

- Learn to identify the most common package managers in Linux, including RPM and dpkg.
- Practice managing RPM packages with the YUM front-end.
- Practice managing Debian packages with the APT front-end.
- Learn how to acquire software through methods other than the usual Linux package managers.
- Learn how to compile software packages from source code.

Package Manager

Every computerized device uses some form of software to perform its intended tasks. In the early days of software, products were stringently tested for bugs and other defects. For the last decade or so, software has been released via the internet with the intent that any bugs would be fixed by applying new versions of the software. In some cases, each individual application has its own updater. In others, it is left up to the user to figure out how to obtain and upgrade software.

This video give a brief overview of package management in Linux. This is by no means an exhaustive tutorial, as various Linux distributions use different package managers.



How was software on Linux installed before package managers?

Historically, software was provided either via FTP or mailing lists (eventually this distribution would grow to include basic websites). Only a few small files contained the instructions to create a binary (normally in a tarfile). You would untar the files, read the readme, and as long as you had GCC or some other form of C compiler, you would then typically run a ./configure script with some list of attributes, such as pathing to library files, location to create new binaries, etc. In addition, the configure process would check your system for application dependencies. If any major requirements were missing, the configure script would exit and you could not proceed with the installation until all the dependencies were met. If the configure script completed successfully, a Makefile would be created.

Once a Makefile existed, you would then proceed to run the make command (this command is provided by whichever compiler you were using). The make command has a number of options called make flags, which help optimize the resulting binaries for your system. In the earlier days of computing, this was very important because hardware struggled to keep up with modern software demands. Today, compilation options can be much more generic as most hardware is more than adequate for modern software.

Finally, after the make process had been completed, you would need to run make install (or sudo make install) in order to actually install the software. As you can imagine, doing this for every single piece of software was time-consuming and tedious—not to mention the fact that updating software was a complicated and potentially very involved process.

What is a Package?

Packages were invented to combat this complexity. Packages collect multiple data files together into a single archive file for easier portability and storage, or simply compress files to reduce storage space. The binaries included in a package are pre-compiled with default compiler options chosen by the developer. If you so desire, you can compile the source files yourself and choose different compiler options.. Packages also contain metadata, such as the software's name, a description of its purpose, a version number, and a list of dependencies necessary for the software to run properly.

Several flavors of Linux have created their own package formats. Some of the most commonly used package formats include:

- .deb: This package format is used by Debian, Ubuntu, Linux Mint, and several other derivatives. It was the first package type to be created.
- .rpm: This package format was originally called Red Hat Package Manager. It is used by Red Hat, Fedora, SUSE, and several other smaller distributions.
- .tar.xz: While it is just a compressed tarball, this is the format that Arch Linux uses.

While packages themselves don't manage dependencies directly, they represented a huge step forward in Linux software management.

What is a Software Repository?

A few years ago, before the proliferation of smartphones, the idea of a software repository was difficult for many users to grasp if they were not involved in the Linux ecosystem. To this day, most Windows users still seem to be hardwired to open a web browser to search for and install new software. However, those with smartphones have gotten used to the idea of a software "store." The way smartphone users obtain software and the way package managers work are not dissimilar. While there have been several attempts at making an attractive UI for software repositories, the vast majority of Linux users still use the command line to install packages. Software repositories are a centralized listing of all of the available software for any repository the system has been configured to use.

Installing Software from Source Code in Linux

One of the greatest strengths of your Linux distribution is its package manager and the associated software repository. With them, you have all the necessary tools and resources to download and install new software on your computer in a completely automated manner.

But despite all their efforts, the package maintainers cannot handle each and every use case. Nor can they package all the software available out there. So there are still situations where you will have to compile and install new software by yourself. One of the most common reasons to compile software is when there is a need to run a very specific version, or modify the source code by the use of some fancy compilation option.

Understanding the Build System of the Program

We usually talk about “compiling the sources,” but the compilation is only one of the phases required to produce a working software from its source. A build system is a set of tool and practices used to automate and articulate those different tasks in order to entirely build the software just by issuing few commands.

If the concept is simple, the reality is somewhat more complicated, because different projects or programming languages may have different requirements. There is an almost endless list of reasons to choose or create another build system. All that to say there are many different solutions used out there.

Red Hat Linux - using RPM	Debian Linux - iusing dpkg
• RedHat Enterprise Linux • Fedora • CentOS • Oracle Enterprise Linux • Scientific • CERN • SUSE • OpenSUSE • Mageia • PCLinuxOS • Berry • Elastix • ClearOS • FrameOS • Fermi • Turbolinux	• Debian • Ubuntu • Kali • Lubuntu • Xubuntu • Kubuntu • Linux Mint • Knoppix • Deepin • peppermint • bodhi linux

RPM-based Package Managers

Updating RPM-based systems, particularly those based on Red Hat technologies, has a very interesting and detailed history. In fact, the current versions of yum (for enterprise distributions) and DNF (for community) combine several open source projects to provide their current functionality.

Initially, Red Hat used a package manager called RPM (Red Hat Package Manager), which is still in use today. However, its primary use is to install RPMs, which you have locally, not to search software repositories. The package manager named up2date was created to inform users of updates to packages and enable them to search remote repositories and easily install dependencies. While it served its purpose, some community members felt that up2date had some significant shortcomings.

The current incantation of yum came from several different community efforts. Yellowdog Updater (YUP) was developed in 1999-2001 by folks at Terra Soft Solutions as a back-end engine for a graphical installer of Yellow Dog Linux. Duke University liked the idea of YUP and decided to improve upon it. They created Yellowdog Updater, Modified (yum) which was eventually adapted to help manage the university's Red Hat Linux systems. Yum grew in popularity, and by 2005 it was estimated to be used by more than half of the Linux market. Today, almost every distribution of Linux that uses RPMs uses yum for package management (with a few notable exceptions).

Debian-based Package Managers

One of the oldest Linux distributions currently maintained, Debian's system is very similar to RPM-based systems. They use .deb packages, which can be managed by a tool called dpkg. dpkg is very similar to RPM in that it was designed to manage packages that are available locally. It does no dependency resolution (although it does dependency checking), and has no reliable way to interact with remote repositories. In order to improve the user experience and ease of use, the Debian project commissioned a project called Deity. This codename was eventually abandoned and changed to Advanced Package Tool (APT).

Released as test builds in 1998 (before making an appearance in Debian 2.1 in 1999), many users consider APT one of the defining features of Debian-based systems. It makes use of repositories in a similar fashion to RPM-based systems, but instead of individual .repo files that yum uses, APT has historically used /etc/apt/sources.list to manage repositories. More recently, it also ingests files from /etc/apt/sources.d/. Following the examples in the RPM-based package managers, to accomplish the same thing on Debian-based distributions you have a few options. You can edit/create the files manually in the aforementioned locations from the terminal, or in some cases, you can use a UI front end (such as Software & Updates provided by Ubuntu et al.).

Dandified Yum (DNF)

DNF is a newer iteration on yum. Introduced in Fedora 18, it has not yet been adopted in the enterprise distributions, and as such is predominantly used in Fedora (and derivatives). Its usage is almost exactly the same as that of yum, but it was built to address poor performance, undocumented APIs, slow/broken dependency resolution, and occasional high memory usage. DNF is meant as a drop-in replacement for yum, and therefore I won't repeat the commands—wherever you would use `yum`, simply substitute `dnf`.

Zypper

Zypper is another package manager meant to help manage RPMs. This package manager is most commonly associated with SUSE (and openSUSE) but has also seen adoption by MeeGo, Sailfish OS, and Tizen. It was originally introduced in 2006 and has been iterated upon ever since. There is not a whole lot to say other than Zypper is used as the back end for the system administration tool YaST and some users find it to be faster than yum.

Zypper's usage is very similar to that of yum. To search for, update, install or remove a package, simply use the following:

```
zypper search kate
zypper update
zypper install kate
zypper remove kate
```

Some major differences come into play in how repositories are added to the system with zypper. Unlike the package managers discussed above, zypper adds repositories using the package manager itself.

Adapted from:
"The evolution of package managers" by Steve Ovens, OpenSource.com is licensed under CC BY-SA 4.0

This page titled 11-A.1: Who are the Package Managers is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

11-B.2: rpm / yum How To

EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

Using `rpm` & `yum`

The `rpm` Command

The **`rpm`** command is a free and open-source package management system. Though it was created for use in Red Hat Linux, **`rpm`** is now used in many Linux distributions such as Fedora, CentOS, OpenSUSE, OpenMandriva, and Oracle Linux.

Syntax:

```
rpm [ OPTIONS ] [ PACKAGE NAME ]
```

As always with Linux commands, **`rpm`** has an abundance of options that allow the user to run commands in a wide variety of output and formats.

Options:

Options	Option Meaning
-U, --upgrade	Upgrade/install
-I, --install	Install
-e, --erase	Remove
-q, --query	Query
-V, --verify	Verify

Using the **`rpm`** command one can install install, upgrade, query and remove software packages.

RPM: Install a Package

The most common command to install a package is:

```
rpm -Uhv package_file.rpm
```

This command upgrades a package with extra output. If the package has not been installed, this command installs the package.

RPM: Upgrading Software

Almost all installation steps use the **`rpm`** command with the –U option, short for upgrade. The basic syntax is:

```
pbmac@pbmac-server $ rpm  -U  package_name
```

For example:

```
pbmac@pbmac-server $ rpm –U jikes-1.16-1.i386.rpm
```

You can also use the --upgrade long option in place of -U .

Unless something goes wrong, you won’t see any response except for the shell prompt ready for your next command.

RPM: Check the Package Is Installed

Use the **`rpm -q`** command to query whether a package has been installed. To verify, you need to use the name of the installed package, not the name of the RPM file. You can also use a partial package name, such as jikes in this case. For example:

```
pbmac@pbmac-server $ rpm –q  jikes
jikes-1.16-1
```

The response shows that the package named jikes-1.16-1 has been installed. This package name corresponds to the RPM file used in the preceding installation example.

The **`rpm -q`** command just asks the RPM database if the package has been installed. There may be other issues with the package that this command won't show. For now, though, the **`rpm –q`** command verifies that the package has been installed.

There are a couple of options that are often found with the -q:

- -qa - Query all packages
- -qi - Display a lot of package information including description
- -qc - List configuration files

So, the command in the example below

```
pbmac@pbmac-server $ rpm –qc jikes
```

will show the configuration files for the jikes package.

The **`yum`** Command

`yum`: Checking For Updates

You can use the **`yum check-update`** command to see which installed packages on your system have updates available:

```
pbmac@pbmac-server $  yum check-update
Loaded plugins: langpacks, presto, refresh-packagekit

PackageKit.x86_64                    0.6.14-2.fc15                fedora
PackageKit-command-not-found.x86_64  0.6.14-2.fc15                fedora
PackageKit-device-rebind.x86_64      0.6.14-2.fc15                fedora
```

`yum`: Updating a Single Package

To update a single package, enter **`yum update package_name`** , for example:

```
pbmac@pbmac-server $  yum update gdb
Loaded plugins: langpacks, presto, refresh-packagekit
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package gdb.x86_64 0:7.2.90.20110411-34.fc15 will be updated
---> Package gdb.x86_64 0:7.2.90.20110429-36.fc15 will be an update
--> Finished Dependency Resolution

Dependencies Resolved
```

```
================================================================================
 Package     Arch         Version                          Repository      Size
================================================================================
Updating:
 gdb         x86_64       7.2.90.20110429-36.fc15          fedora         1.9 M

Transaction Summary
================================================================================
Upgrade       1 Package(s)

Total download size: 1.9 M
Is this ok [y/N]:
```

yum: Updating All Packages and Their Dependencies

To update all packages and their dependencies, simply enter the **`yum update`** command (without any arguments):

```
pbmac@pbmac-server $  yum update
```

yum: Listing Packages

`yum list` and related commands provide information about packages, package groups, and repositories. Variations of the **`yum list`** command include:

yum Command	Meaning
yum list all	Lists all installed and available packages.
yum list installed	Lists all packages installed on your system. The rightmost column in the output lists the repository from which the package was retrieved.
yum list available	Lists all available packages in all enabled repositories.
yum grouplist	Lists all package groups.
yum repolist	Lists the repository ID, name, and number of packages it provides for each enabled repository.

yum: Display Package Information

The **`yum info package_name [ more_names ]`** command displays information about one or more packages (glob expressions are valid here as well). For example, to display information about the abrt package, type:

```
pbmac@pbmac-server $  yum info abrt
```

yum: Install Packages

You can install a package and all of its non-installed dependencies by entering:

```
 pbmac@pbmac-server $ yum install  package_name
```

You can install multiple packages simultaneously by appending their names as arguments:

```
pbmac@pbmac-server $ yum install  package_name  [ more_names ]
```

`yum`: Remove Packages

The **`yum remove package_name`** uninstalls (removes in RPM and Yum terminology) the package, as well as any packages that depend on it. As when you install multiple packages, you can remove several at once by adding more package names to the command. For example, to remove totem, rhythmbox, and sound-juicer, type the following at a shell prompt:

```
pbmac@pbmac-server $  yum remove totem rhythmbox sound-juicer
```

Adapted from:
"The evolution of package managers" by Steve Ovens, OpenSource.com is licensed under CC BY-SA 4.0
"Cap. 2. RPM Overview" by Ben Cotton is licensed under CC BY-SA 3.0
"Chapter 4. Yum" by Multiple Contributors, Fedora 15 Deployment Guide is licensed under CC BY-SA 3.0

This page titled 11-B.2: rpm / yum How To is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

11-C.3: dpkg / apt How To

EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

Using **dpkg** / **apt**

The **dkpg** Command

dpkg is a tool to install, build, remove and manage Debian packages. The primary and more user-friendly front-end for **dpkg** is aptitude (also known as **apt**). **dpkg** itself is controlled entirely via command line parameters, which consist of exactly one action and zero or more options. The action-parameter tells **dpkg** what to do and options control the behavior of the action in some way.

Syntax:

```
dpkg [ OPTIONS ] [ PACKAGE_NAME ]
```

A few of the more common options are:

Options	Option Meaning
-i, –install	Install a package using the **dpkg** command. The command will extract all control files for the specified package, remove any previously installed older instance of the package, and install the new package on our system.
-r, –remove	Remove an installed package from our system. It removes every file belonging to the specific package except the configuration files. This can be seen as the uninstallation option.
-P, –purge	An alternative way to remove an installed package from our system. It completely removes every fie belonging to the specific package, including the configuration files. This can be seen as the ‘complete uninstallation’ option.
–update-avail	The information of the **dpkg** command about available packages in its repositories. If new packages are available, they are synced from the official repositories.
–merge-avail	Merge the information of the **dpkg** command about available packages in its repositories with previously available information. It is usually run right after the previous command.
–help	Display the help page for the **dpkg** command and exit.

The **apt** Command

Debian Linux uses **dpkg** packaging system. A packaging system is a way to provide programs and applications for installation. This way, the user or the system administrator does not have to build a program from the source code. APT (Advanced Package Tool) is the command line tool to interact with the packaging system in Debian-based Linux distributions.

There is already **dpkg** command to manage it. But **apt** is a more friendly way to handle packaging. You can use it to find and install new packages, upgrade packages, and remove the packages, etc.

Syntax:

```
apt [ OPTIONS ] [ sub-command ] [ package ]
```

`apt`: Update Package Database

`apt` actually works on a database of available packages. If the database is not updated, the system won't know if there are any newer packages available. This is why updating the repository should be the first thing to do in any Linux system after a fresh install.

Updating the package database requires superuser privileges so you'll need to use sudo.

```
pbmac@pbmac-server $ sudo apt update
Hit:1 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:2 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Ign:4 http://dl.google.com/linux/webdesigner/deb stable InRelease
Hit:5 https://deb.nodesource.com/node_10.x bionic InRelease
Hit:6 http://dl.google.com/linux/webdesigner/deb stable Release
```

You'll see three types of lines here, Hit, Get, and Ign. Basically these are:

- Hit: there is no change in package version from the previous version.
- Ign: the package is being ignored. Either the package is too recent that it doesn't even bother to check, or there was an error in retrieving the file but the error was trivial and thus it is being ignored. Don't worry, this is not an error.
- Get: There is a new version available. It will download the information about the version (not the package itself). You can see that there is download information (size in kb) with the 'get' line in the screenshot above.

`apt`: Upgrade Installed Packages

Once you have updated the package database, you can now upgrade the installed packages. The most convenient way is to upgrade all the packages that have available updates. You can simply use the command below:

```
pbmac@pbmac-server $ sudo apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade...Done
The following packaged will be upgraded:
```

This will show you how many and which packages are going to be upgraded and ask if you want to continue.

`apt`: Install New Packages

If you already know the name of the package, you can install it using the command below:

```
pbmac@pbmac-server $ sudo apt install <package_name>
```

Just replace the <package_name> with the desired package. Suppose you want to install mplayer, you can simply use the command below:

```
pbmac@pbmac-server $ sudo apt install mplayer
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  esound-common libaacs0 libaudiofile1 libbluray1 libesd0 libgif4 libvdpau1
```

```
Suggested packages:
  libbluray-bdj pulseaudio-esound-compat nvidia-vdpau-driver
  nvidia-vdpau-driver-ia32 vdpau-driver
The following NEW packages will be installed:
  esound-common libaacs0 libaudiofile1 libbluray1 libesd0 libgif4 libvdpau1
  mplayer2
0 upgraded, 8 newly installed, 0 to remove and 1 not upgraded.
Need to get 1,626 kB of archives.
After this operation, 4,365 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

The output shows extra packages that will be installed, a few suggested packages, and any new packages. The output tells you how much additional disk space will be used and then ask if you want to continue.

You are not bound to install just one package at a time. You can install several packages at a time by providing the package names all together:

```
pbmac@pbmac-server $ sudo apt install <package_1> <package_2> <package_3>
```

`apt`: Remove Installed Packages

Removing packages is as easy as installing them. Just use the command below:

```
pbmac@pbmac-server $ sudo apt remove <package_name>
```

Another way of uninstalling packages is to use **`purge`**. The command is used in the following manner:

```
 sudo apt purge <package_name>
```

What is the difference between apt remove and apt purge?

- **`apt remove`** just removes the binaries of a package. It leaves residue configuration files.

- **`apt purge`** removes everything related to a package including the configuration files.

If you used **`apt remove`** to a get rid of a particular software and then install it again, your software will have the same configuration files. Of course, you will be asked to override the existing configuration files when you install it again.

Purge is useful when you have messed up with the configuration of a program. You want to completely erase its traces from the system and perhaps start afresh. And yes, you can use **`apt purge`** on an already removed package.

The apt-get and apt-cache

There are also two other commands, **`apt-get`** and **`apt-cache`**, that are almost the same as the **`apt`** command. Some of the Debian distributions have opted to use **`apt`** as opposed to **`apt-get`** and **`apt-cache`** command. They all work, and are still viable commands. If you want to know more about the differences simply do a web search of the difference between these commands.

Adapted from:
"Using apt Commands in Linux [Complete Guide]" by Abhishek Prakash, Its FOSS is licensed under CC BY-SA 4.0

This page titled 11-C.3: dpkg / apt How To is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

11-D.4: Repository Configuration

EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

Repositories in Linux

On Linux systems most software is packaged in .deb or .rpm, files which contain the programs and libraries you need. These packages are stored in repositories, or repos. Repositories are servers which contain sets of packages. These packages can be downloaded using package manager utilities like **`yum`** or **`apt`** that we have discussed.

There are generally three different levels of repositories:

1. Local repositories - where each system has their own repository. It is best if these are used for packages that the system's user wants to use for specific purposes. If a local repository is used more for software that is used across the organization then it is very easy to create version conflicts because each system has their own repository.
2. Centralized repository - to resolve the problem mentioned above, there are central repositories where systems across the organization can download their packages from. This makes it much easier for system administrators to control what versions users are running.
3. Vendor repository - the software vendor controls these repositories, and which version are made available.

`yum`: Repository Configuration

The configuration file for yum and related utilities is located at /etc/yum.conf. This file contains one mandatory [main] section, which allows you to set Yum options that have global effect, and may also contain one or more [repository] sections, which allow you to set repository-specific options. However, best practice is to define individual repositories in new or existing .repo files in the /etc/yum.repos.d/directory. The values you define in the [main] section of the /etc/yum.conf file may override values set in individual [repository] sections.

```
pbmac@pbmac-server $  mkdir -p /home/pbmac/test-package/repo
pbmac@pbmac-server $  chown -R root.root /home/pbmac/test-package/repo
pbmac@pbmac-server $  createrepo /home/pbmac/test-package/repo
pbmac@pbmac-server $  chmod -R o-w+r /home/pbmac/test-package/repo
```

Create a repository configuration file, e.g. /etc/yum.repos.d/testP.repo containing below configuration.

```
[testP]
name=Pat's Test Repo
baseurl=file:///home/pbmac/test-package/repo
enabled=1
gpgcheck=0
```

Now, use **`yum repolist,`** and you will see the newly created repository in the list of accessible repositories.

```
pbmac@pbmac-server $  yum repolist
Loaded plugins: product-id, refresh-packagekit, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscrip
rhel_repo                          | 2.9 kB     00:00 ...
rhel_repoprimary_db                | 367 kB     00:00 ...
repo id                            repo name                          status
RHEL_6.4_Disc                      RHEL_6.4_x86_64_Disc               3,648
testP                              Pat's Test Repo                      150
```

apt: Repository Configuration

Ubuntu uses **`apt`** for package management. **`apt`** stores a list of repositories or software channels in the file

```
/etc/apt/sources.list
```

and in any file with the suffix .list under the directory:

```
/etc/apt/sources.list.d/
```

See **man sources.list** for more about this storage mechanism.

By editing these files from the command line, we can add, remove, or temporarily disable software repositories.

Typically, the beginning of the file /etc/apt/sources.list looks like this:

```
# sources.list
#deb cdrom:[Ubuntu 13.10 _Saucy Salamander_ - Release i386 (20131016.1)]/ saucy main r

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://ch.archive.ubuntu.com/ubuntu/ saucy main restricted
deb-src http://ch.archive.ubuntu.com/ubuntu/ saucy main restricted
#...
```

Explanation of the Repository Format

- All the lines beginning with one or two hashes (#) are comments, for information only.
- The lines without hashes are apt repository lines. Here's what they say:

    - **deb**: These repositories contain binaries or pre-compiled packages. These repositories are required for most users.
    - **deb-src**: These repositories contain the source code of the packages. Useful for developers.
    - **http://archive.ubuntu.com/ubuntu**: The URI (Uniform Resource Identifier), in this case a location on the internet.
    - **saucy** is the release name or version of your distribution.
    - **main** & **restricted** are the section names or components. There can be several section names, separated by spaces.

Adapted from:
"Repositories/CommandLine" by Mikkel Kirkgaard Nielsen, CommunityHelpWiki is licensed under CC BY-SA 3.0

This page titled 11-D.4: Repository Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

11-E.5: Other Software Sources

EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

Favorite App Sites

As Linux grows in popularity, the number of open source apps increases as well. While it was not so long ago that Linux was still a bit of an oddity in the computing world, it has now grown to such significant numbers that not only is the open source world porting apps to Linux, but the commercial app developers are also making versions of their software available to Linux users.

Some of the more popular apps can be found by a simple web search:

- gimp - offers almost every type of tool you will ever require to manipulate an image, scale it, crop it, or simply add a layer to it.
- Audacity - the most popular tool for basic audio editing tasks.
- OBS - a pretty popular and robust screen recorder app.
- VLC - a pretty simple media player that's open-source and free as well.
- Virtualbox - an amazing free and open-source virtualization solution for those who want to try different distros (or experiment with stuff) without affecting your host system.

There is a lot of software available, and sometimes you have to search for it.

The `wget` and `curl` Commands

`wget` is the non-interactive network downloader, which is used to download files from the server even when the user has not logged on to the system. It can work in the background without hindering the current process. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies. **`wget`** is non-interactive, meaning that it can work in the background, while the user is not logged on. This allows you to start a retrieval and disconnect from the system, letting **`wget`** finish the work. **`wget`** can follow links in HTML and XHTML pages and create local versions of remote web sites, fully recreating the directory structure of the original site. This is sometimes referred to as recursive downloading. While doing that, **`wget`** respects the Robot Exclusion Standard (/robots.txt). **`wget`** can be instructed to convert the links in downloaded HTML files to the local files for offline viewing.

`curl` is a command line tool to transfer data to or from a server, using any of the supported protocols (HTTP, FTP, IMAP, POP3, SCP, SFTP, SMTP, TFTP, TELNET, LDAP or FILE). **`curl`** is powered by Libcurl. This tool is preferred for automation, since it is designed to work without user interaction. **`curl`** can transfer multiple files at once.

These two tools both accomplish the same tasks, although there are differences:

- **`wget`** is recursive, meaning it can follow web page links and download subdirectories.
- **`wget`** is a command line only tool. **`curl`** is actually implemented as a cross-platform library with a stable API.
- **`curl`** supports more protocols than **`wget.`**

The `tar` Command

`tar` stands for 'tape archive' and is used to create/extract archive files. The **`tar`** command is one of the important commands which provides archiving functionality in Linux. We can use Linux **`tar`** command to create compressed or uncompressed archive files as well as maintain and modify them. An archive file is a file that is composed of one or more files along with metadata. Archive files are used to collect multiple data files together into a single file for easier portability and storage, or simply to compress files to use less storage space.

Syntax:

```
tar [ OPTIONS ] [ ARCHIVE-FILE-NAME ] [ file or directory to be archived ]
```

The options for **`tar`** are:

Options	Option Meaning
-c	Creates archive
-x	Extracts the archive
-f	Creates archive with given filename
-t	Displays or lists files in archived file
-u	Archives and adds to an existing archive file
-v	Displays verbose information
-A	Concatenates the archive files
-z	Zip, tells tar command that creates tar file using gzip
-j	Filter archive tar file using tbzip
-W	Verify an archive file
-r	Update or add file or directory in already existed .tar file

We can create an archive file using the c option (when using **tar** options we are NOT required to use a '-' in front of the option). So in the following command the options are c=create, v=verbose, and f=file to create which is exes.tar, and the files to include in that archive file are xya.cpp and xyz.cpp.

```
pbmac@pbmac-server $ tar cvf exes.tar xya.cpp xyz.cpp
xya.cpp
xyz.cpp

Then to list the contents of the file, use the t option
pbmac@pbmac-server $ tar tvf exes.tar
-rw-rw-r-- pbmac/pbmac     105 2019-09-19 20:58 xya.cpp
-rw-rw-r-- pbmac/pbmac     628 2020-09-24 20:21 xyz.cpp

To extract the files from the archive the extract optoin 'x' is used
pbmac@pbmac-server $ tar xvf exes.tar
xya.cpp
xyz.cpp
```

The **gzip** Command

The **gzip** command compresses files. Each individual file is compressed and added to the resulting file that is named the same as the original file with a .gz extension. The compressed file consists of a GNU zip header and compressed data.

If given a file as an argument, **gzip** compresses the file, adds a ".gz" suffix, and deletes the original file. With no arguments, **gzip** compresses the standard input and writes the compressed file to standard output. This is similar to the zip utility, but the two commands use different compression algorithms.

Notice, in the following example, the xyz.cpp file no longer exists; it was compressed, and the uncompressed file is now gone.

```
pbmac@pbmac-server $ gzip xya.cpp
pbmac@pbmac-server $ ls xya*
```

```
xya.cpp.gz
```

Also, it is possible to use a **gzip** option with the **tar** command. In the following example the z option is included and the output file has a .gz extension added onto it. We can see that the two files are different using the **file** command.

```
pbmac@pbmac-server $ tar czvf exes.tar.gz xya.cpp xyz.cpp
xya.cpp
xyz.cpp
pbmac@pbmac-server $ ls exes*
exes.tar  exes.tar.gz
pbmac@pbmac-server $ file exes.tar*
exes.tar:    POSIX tar archive (GNU)
exes.tar.gz: gzip compressed data, last modified: Mon Oct 26 21:10:53 2020, from Unix
```

Adapted from:
"tar command in Linux with examples" by Akansh Gupta, Geeks for Geeks is licensed under CC BY-SA 4.0
"Gzip Command in Linux" by Shubrodeep Banerje, Geeks for Geeks is licensed under CC BY-SA 4.0

This page titled 11-E.5: Other Software Sources is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

11-F.6: Software Builds from Source

EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

What is Building from Source

Building software from source means installing a program either by using a package manager and downloading a source package, or by using some other method to download to source code. Then, you compile the source code and install the binaries to your computer.

Many projects provide their source code via hosting services such as GitHub, GitLab, or Bitbucket. Larger projects at times host their source code on a project specific website. The code will usually be available as compressed archive file of some sort.

Compilers

One of the key components when building from source is the compiler. Every computer programming language has its own compiler (or interpreter), that converts the programming statements into machine readable executable language, also referred to as binary files.

In Linux, the most common compiler is the GNU compiler collection, which includes front end for several compilers including C and C++.

Libraries

In addition to the compiler, building programs from source requires that certain files, called libraries, are installed on the system. Libraries are collections of pre-compiled code that can be linked into other programs. The most common use of libraries is to provide system functions, such as the square root function sqrt found in the math library, or the C++ cout function.

Libraries are typically stored in special archive files with the extension '.a,' referred to as static libraries. Compiled programs need to be able to access shared libraries when they run so as to link to the library code. The standard system libraries are usually found in the directories '/usr/lib' and '/lib.'

The `ldd` Command

The **`ldd`** command prints the shared libraries required by each program or shared library specified on the command line.

```
pbmac@pbmac-server $ ldd zoo
    linux-vdso.so.1 (0x00007ffd6fb7e000)
    libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f3f565d6000)
    libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f3f563be000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3f55fcd000)
    libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f3f55c2f000)
    /lib64/ld-linux-x86-64.so.2 (0x00007f3f5695f000)
```

This shows the libraries like linux-vdso, libc.so and ld-linux-x86-64.so are used by the zoo binary. There is also the full path of the given libraries where they are generally residing in /lib or /lib64 directories.

The Compiling Process

A special set of tools help automate the building process. On Linux systems, this often comes in the form of a command line program called make. Source code written in different languages need specific compilers and commands to change them into binaries. The make program automates this process. Over the years, a pretty specific process has been developed for most open source software. This process is outlined below, and shows the commands to build the example application.

```
1. take the file you have downloaded, and extract the files. In this case its a gzippe
pbmac@pbmac-server $  tar -xzvf testP.tar.gz
Unpacking ...

2. Usually the files are archived wihtin a directory - so we have to change into the d
pbmac@pbmac-server $  cd testP

3. Usually there is a file named INSTALL which tells you the process to build this pac
pbmac@pbmac-server $  more INSTALL

4. Run the configure script, which determines the system configuration
pbmac@pbmac-server $  ./configure

5. Now run the 'make' command to actually compile all the source files, and create the
pbmac@pbmac-server $  make

6. Finally, install the newly created package
pbmac@pbmac-server $  make install
```

The make Command and makefiles

There is a command - the **`make`** command - that reads a configuration file called the makefile, which contains rules for building the application. If we were going to put together a makefile for an applicatoin named testP, it might look like the following:

```
testP: testP.o other.o
        @echo "Checking.."
        g++ -lm testP.o -o testP

testP.o: testP.cpp
        @echo "Creating object.."
        g++ -c testP.cpp

other.o: other.cpp
        g++c other.cpp
```

On the first line - testP is known as the 'target' - this is what we are creating. It is dependent on two files: testP.o and other.o. The .o files are object files that have been partially compiled. Since there are dependencies, the make command will look at each of them, and check out the other rules. For testP.o it will check and see if testP.cpp has changed since testP.o was last created....if so then the two lines following the testP.o target get executed, which echoes "Creating object..." and compiles the testp.cpp file. The same is true of the other.o rule. Once everything is up to date, the testP executable file is created.

Make files are really awesome; it's almost like learning another programming language when you get into the really comprehensive files.

Adapted from:
"What is a Makefile and how does it work?" by Sachin Patil, OpenSource.com is licensed under CC BY-SA 4.0

This page titled 11-F.6: Software Builds from Source is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

11-G.7: Software Dependency Troubleshooting

EXAM OBJECTIVES COVERED

4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

Troubleshooting Dependency Issues

Even with all the tools that have been discussed, there come times when a package does not compile, or may not load, or perhaps doesn't run properly because of some dependency issues. These problems can be as simple as downloading some small package, or it may require a lot of digging through documentation, log files and web searches. Sometimes the best thing to do is to remove the package and start all over again.

If the package is being built from scratch, verify that the proper compiler version and supporting libraries are installed. If there are issues with a particular version of a compiler or library it often will mention that in a README, or INSTALL text file, which you should read prior to attempting any build.

Sometimes access to a remote server gets dropped in the middle of a download, and even though the download appears to complete, there may be issues with the file that was downloaded. Checking the checksum files once the download is complete can save a lot of time.

`yum` and `apt` Dependencies

When using a package manager it might be best to configure it to automatically download all dependencies, so that you don't run into surprises when you attempt to run the software. Otherwise the person installing the software needs to be diligent to know which version of which software needs to be installed. This creates a lot of extra work, and increases the opportunity for dependency issues to wreak havoc.

Dependency Issue Guidelines

Software Dependency and Repository Issues

Use the tools provided by the package manager to troubleshoot dependencies and repositories:

- **`rpm -V [ PACKAGE_NAME ]`** will verify that all of the required pieces of the package are installed.
- **`rpm -qR [ PACKAGE_NAME ]`** or **`yum deplist [ PACKAGE_NAME ]`** can be used to discover the dependencies.
- **`apt show [ PACKAGE_NAME ]`** will provide a LOT of information...maybe a bit too much, but it does show the dependencies.
- There is **`apt-cache depends [ PACKAGE_NAME ]`** which provides more succinct output of the packages' dependencies.
- **`apt-rdepends -r [ PACKAGE_NAME ]`** is a command that provides dependency information AND reverse dependency information - what other packages rely on this one.
- **`dpkg -I path_to_deb_file`** will determine the dependency for a .deb file.

Patch and Update Issues

Most software providers will, at various times, provide patches and updates. Be aware of apply patches without understanding what they do, what problems they resolve, and whether your system needs to apply the patch.

- Read whatever documentation the patch contains, and make sure you know the issue(s) the patch resolves. Check to see if you can recreate the problem before you add a patch.
- If you have the capability apply the patch and test it prior to applying the patch to a production system.
- Always know how to back out of a patch, or have a plan in case there are unforeseen issues after applying the patch.
- Make sure to check software versions if any are specified in the patch documentation.

Compiler and Library Issues

- As stated above, read the documentation for any known compiler or library issues. Ensure the system has the proper versions of the compiler, libraries and other software.

- System level software may have kernel dependencies, and may require you to be root to install/update software.
- Use the **`ldd`** command previously discussed.
- If there is the capability, compile and test the software on a test system prior to loading it on a production system.

This page titled 11-G.7: Software Dependency Troubleshooting is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

CHAPTER OVERVIEW

Module 12: Linux Systems Security

12-A.1: Best Cybersecurity Practices

12-A.1.1: Best Cybersecurity Practices - Continued

12-A.2: chroot Jail

12-A.3: Encryption in Linux

12-A.4: Security Best Practices

12-B.5: Identity and Access Management

12-B.6: Pluggable Authentication Modules

12-B.7: TTY Security & PKI

12-B.8: OpenSSL / VPN

12-B.9: Authentication Issues

12-C.10: SELinux Configuration

12-C.11: AppArmor

12-D.12: Firewall Configuration in Linux

12-D.13: Access Control Lists / iptables

12-D.14: UFW & firewalld

12-D.15: Netfilter / IP Forwarding

12-E.16: Linux Logging

12-E.17: journalctl & last Commands

12-F.18: Backup and Restore in Linux

12-F.19: tar / cpio / dd

12-F.20: mirrorvg / rsync / scp

12-F.21: File Compression

12-F.22: Integrity Check and Checksums

Module 12: Linux Systems Security is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-A.1: Best Cybersecurity Practices

EXAM OBJECTIVES COVERED

3.3 Summarize security best practices in a Linux environment.

Objectives of this Module

In this lesson, you will:

- Learn what the best practices are for cybersecurity in Linux, and how to implement them
- Carry out identity and access management (IAM) methods in Linux
- Learn to configure firewalls to filter network traffic
- Configure and run event and system logging services
- Learn how to back up, restore, and verify data

Cybersecurity

Cybersecurity is procedure and strategy associated with insuring touchy information, PC frameworks, systems, and programming applications from digital assaults. Cyber assaults are general phrasing which covers an enormous number of themes, however, a portion of the famous are:

- Altering frameworks and information put away inside
- Abuse of assets
- Unapproved access to focus on framework and getting to delicate data
- Upsetting typical working of the business and its procedures
- Utilizing ransomware assaults to scramble information and coerce cash from casualties

The assaults that are equipped for disturbing security and hacking of frameworks are currently turning out to be increasingly inventive and complex. So it’s trying for each business and security expert to defeat this test and retaliate with these assaults.

Since Linux is no different from any other computer that is connected to the Internet, they are also open to attack from those same attacks. Linux does have some built in security concepts - permissions, userid, groups etc. - but that does not make them a "secure" system.

The CIA Triad

The CIA triad of confidentiality, integrity, and availability is at the heart of information security. However, debate continues about whether or not this CIA triad is sufficient to address rapidly changing technology and business requirements. There are recommendations to consider expanding on the intersections between availability and confidentiality, as well as the relationship between security and privacy. Other principles such as "accountability" have sometimes been proposed; it has been pointed out that issues such as non-repudiation do not fit well within the three core concepts.

Confidentiality

In information security, confidentiality "is the property, that information is not made available or disclosed to unauthorized individuals, entities, or processes." While similar to "privacy," the two words aren't interchangeable. Rather, confidentiality is a component of privacy that is implemented to protect our data from unauthorized viewers. Examples of confidentiality of electronic data being compromised include laptop theft, password theft, or sensitive emails being sent to the incorrect individuals.

Integrity

In information security, data integrity means maintaining and assuring the accuracy and completeness of data over its entire lifecycle. This means that data cannot be modified in an unauthorized or undetected manner. This is not the same thing as referential integrity in databases, although it can be viewed as a special case of consistency as understood in the classic ACID model of transaction processing. Information security systems typically provide message integrity alongside confidentiality.

Availability

For any information system to serve its purpose, the information must be available when it is needed. This means the computing

systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it must be functioning correctly. High availability systems aim to remain available at all times, preventing service disruptions due to power outages, hardware failures, and system upgrades. Ensuring availability also involves preventing denial-of-service attacks, such as a flood of incoming messages to the target system, essentially forcing it to shut down.

In the realm of information security, availability can often be viewed as one of the most important parts of a successful information security program. Ultimately end-users need to be able to perform job functions; by ensuring availability an organization is able to perform to the standards that an organization's stakeholders expect. This can involve topics such as proxy configurations, outside web access, the ability to access shared drives and the ability to send emails. Executives oftentimes do not understand the technical side of information security and look at availability as an easy fix, but this often requires collaboration from many different organizational teams, such as network operations, development operations, incident response and policy/change management. A successful information security team involves many different key roles to mesh and align for the CIA triad to be provided effectively.

Types of Authentication

Authentication is the act of verifying a claim of identity. When John Doe goes into a bank to make a withdrawal, he tells the bank teller he is John Doe, a claim of identity. The bank teller asks to see a photo ID, so he hands the teller his driver's license. The bank teller checks the license to make sure it has John Doe printed on it and compares the photograph on the license against the person claiming to be John Doe. If the photo and name match the person, then the teller has authenticated that John Doe is who he claimed to be. Similarly, by entering the correct password, the user is providing evidence that he/she is the person the username belongs to.

There are three different types of information that can be used for authentication:

Something you know: things such as a PIN, a password, or your mother's maiden name
Something you have: a driver's license or a magnetic swipe card
Something you are: biometrics, including palm prints, fingerprints, voice prints and retina (eye) scans

Strong authentication requires providing more than one type of authentication information (two-factor authentication). The username is the most common form of identification on computer systems today and the password is the most common form of authentication. Usernames and passwords have served their purpose, but they are increasingly inadequate. Usernames and passwords are slowly being replaced or supplemented with more sophisticated authentication mechanisms such as time-based One-Time Password algorithms.

There are several methods to provide authentication. We cover some of them in the table below:

Authentication Method	Implementation
Passwords	A password, sometimes called a passcode, is a memorized secret, typically a string of characters, usually used to confirm the identity of a user. Using the terminology of the NIST Digital Identity Guidelines, the secret is memorized by a party called the claimant while the party verifying the identity of the claimant is called the verifier. When the claimant successfully demonstrates knowledge of the password to the verifier through an established authentication protocol, the verifier is able to infer the claimant's identity.
Passphrase	A passphrase is a sequence of words or other text used to control access to a computer system, program or data. A passphrase is similar to a password in usage, but is generally longer for added security. Passphrases are often used to control both access to, and operation of, cryptographic programs and systems, especially those that derive an encryption key from a passphrase.
PINs	A personal identification number (PIN), or sometimes redundantly a PIN number, is a numeric or alpha-numeric passcode used in the process of authenticating a user accessing a system. The personal identification number has been the key to flourishing the exchange of private data between different data-processing centers in computer networks for financial institutions, governments, and enterprises. PINs may be used to authenticate banking systems with cardholders, governments with citizens, enterprises with employees, and computers with users, among other uses.
One Time Password (OTP)	A one-time password (OTP), also known as one-time PIN or dynamic password, is a password that is valid for only one login session or transaction, on a computer system or other digital device. OTPs avoid a number of shortcomings that are associated with traditional (static) password-based authentication; a number of implementations also incorporate two-factor authentication by ensuring that the one-time password requires access to something a person has (such as a small keyring fob device with the OTP calculator built into it, or a smartcard or specific cellphone) as well as something a person knows (such as a PIN). The most important advantage that is addressed by OTPs is that, in contrast to static passwords, they are not vulnerable to replay attacks. This means that a potential intruder who manages to record an OTP that was already used to log into a service or to conduct a transaction will not be able to abuse it, since it will no longer be valid. A second major advantage is that a user who uses the same (or similar) password for multiple systems is not made vulnerable on all of them, if the password for one of these is gained by an attacker.
Hardware Token	A security token is a peripheral device used to gain access to an electronically restricted resource. The token is used in addition to or in place of a password. It acts like an electronic key to access something. Examples include a wireless keycard opening a locked door, or in the case of a customer trying to access their bank

Authentication Method	Implementation
	account online, the use of a bank-provided token can prove that the customer is who they claim to be. Some tokens may store cryptographic keys that may be used to generate a digital signature, or biometric data, such as fingerprint details. Some may also store passwords. Some designs incorporate tamper resistant packaging, while others may include small keypads to allow entry of a PIN or a simple button to start a generating routine with some display capability to show a generated key number. Connected tokens utilize a variety of interfaces including USB, near-field communication (NFC), radio-frequency identification (RFID), or Bluetooth.
Biometric devices	A biometric device is a security identification and authentication device. Such devices use automated methods of verifying or recognizing the identity of a living person based on a physiological or behavioral characteristic. These characteristics include fingerprints, facial images, iris and voice recognition
RADIUS	Remote Authentication Dial-In User Service (RADIUS) is a networking protocol, operating on port 1812, that provides centralized Authentication, Authorization, and Accounting (AAA or Triple A) management for users who connect and use a network service. RADIUS was developed by Livingston Enterprises, Inc. in 1991 as an access server authentication and accounting protocol and later brought into the Internet Engineering Task Force (IETF) standards. RADIUS is a client/server protocol that runs in the application layer, and can use either TCP or UDP as transport. Network access servers, the gateways that control access to a network, usually contain a RADIUS client component that communicates with the RADIUS server. RADIUS is often the back-end of choice for 802.1X authentication as well.
TACACS	Terminal Access Controller Access-Control System (TACACS, /ˈtækæks/) refers to a family of related protocols handling remote authentication and related services for networked access control through a centralized server. Terminal Access Controller Access-Control System Plus (TACACS+) is a protocol developed by Cisco and released as an open standard beginning in 1993. Although derived from TACACS, TACACS+ is a separate protocol that handles authentication, authorization, and accounting (AAA) services. TACACS+ have largely replaced their predecessors.
LDAP	The Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. Directory services play an important role in developing intranet and Internet applications by allowing the sharing of information about users, systems, networks, services, and applications throughout the network. As examples, directory services may provide any organized set of records, often with a hierarchical structure, such

Authentication Method	Implementation
	as a corporate email directory. Similarly, a telephone directory is a list of subscribers with an address and a phone number.
Kerberos	Kerberos is a computer-network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. The protocol was named after the character Kerberos (or Cerberus) from Greek mythology, the ferocious three-headed guard dog of Hades. Its designers aimed it primarily at a client–server model and it provides mutual authentication—both the user and the server verify each other's identity. Kerberos protocol messages are protected against eavesdropping and replay attacks.

Kerberos in Linux

Linux has an installed version of Kerberos. In some distributions you may have to install the package to be able to use the software.

Here are a few of the more important commands for Linux Kerberos:

`kinit` - obtain and cache Kerberos ticket-granting ticket
`kpasswd` - change a user's Kerberos password
`klist` - list cached Kerberos tickets
`kdestroy` - destroy Kerberos tickets

To start Kerberos the **`kinit`** command is used:

```
pbmac@pbmac-server $ kinit pbmac@thisdomain.com
Password for pbmac@thisdomain.com:
pbmac@pbmac-server $
```

Use the **`klist`** tool to verify that a ticket was indeed obtained:

```
pbmac@pbmac-server $ klist -v
Credentials cache: API:501:9
        Principal: pbmac@thisdomain.com
    Cache version: 0

Server: pbmac-server/thisdomain.com@thisdomain.com
Client: pbmac@thisdomain.com
Ticket etype: aes128-cts-hmac-sha1-96
Ticket length: 256
Auth time:  Mon Nov  2 17:04:42 2020
End time:   Sat Oct 31  7:34:54 2020
Renew till: Mon Nov  9 17:04:42 2020
Ticket flags: pre-authent, initial, renewable, forwardable
Addresses: addressless
pbmac@pbmac-server $
```

Adapted from:
"Password" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Passphrase" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"Personal identification number" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"One-time password" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Security token" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Biometric device" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"RADIUS" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"TACACS" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Lightweight Directory Access Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Kerberos (protocol)" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

12-A.1: Best Cybersecurity Practices is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-A.1.1: Best Cybersecurity Practices - Continued

Multi-Factor Authentication (MFA)

MFA is quite simple, and organizations are focusing more than ever on creating a smooth user experience. In fact, you probably already use it in some form. For example, you've used MFA if you've:

- swiped your bank card at the ATM and then entered your PIN (personal ID number).
- logged into a website that sent a numeric code to your phone, which you then entered to gain access to your account.

MFA, sometimes referred to as two-factor authentication or 2FA, is a security enhancement that allows you to present two pieces of evidence – your credentials – when logging in to an account. Your credentials fall into any of these three categories: something you know (like a password or PIN), something you have (like a smart card), or something you are (like your fingerprint). Your credentials must come from two different categories to enhance security – so entering two different passwords would not be considered multi-factor.



Figure 12 – A.1.1.1: Multifactor Authentication. ("Multifactor Authentication" by NIST is in the Public Domain, CC0)

So look at a simple scenario: logging in to your bank account. If you've turned on MFA or your bank turned it on for you, things will go a little differently. First and most typically, you'll type in your username and password. Then, as a second factor, you'll use an authenticator app, which will generate a one-time code that you enter on the next screen. Then you're logged in – that's it!

In most cases it's even easier than that. Most MFA approaches will remember a device. So if you come back using the same phone or computer, the site remembers your device as the second factor. Between device recognition and analytics the bank is likely performing—such as whether you're logging in 20 minutes later from halfway around the world—most of the time the only ones that have to do any extra work are those trying to break into your account.

Understanding Privilege Escalation

Privilege escalation is the act of exploiting a bug, design flaw or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user. The result is that an application with more privileges than intended by the application developer or system administrator can perform unauthorized actions.

Most computer systems are designed for use with multiple user accounts, each of which has abilities known as privileges. Common privileges include viewing and editing files, or modifying system files.

Privilege escalation means a user receives privileges they are not entitled to. These privileges can be used to delete files, view private information, or install unwanted programs such as viruses. It usually occurs when a system has a bug that allows security to be bypassed or, alternatively, has flawed design assumptions about how it will be used. Privilege escalation occurs in two forms:

- **Vertical privilege escalation**, also known as *privilege elevation*, where a lower privilege user or application accesses functions or content reserved for higher privilege users or applications (e.g., Internet Banking users can access site administrative functions or the password for a smartphone can be bypassed).

- **Horizontal privilege escalation**, where a normal user accesses functions or content reserved for other normal users (e.g., Internet banking User A accesses the Internet bank account of User B)

Adapted from:
"Back to basics: Multi-factor authentication (MFA)" by unknown, NIST - APPLIED CYBERSECURITY DIVISION is in the Public Domain, CC0
"Privilege escalation" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

12-A.1.1: Best Cybersecurity Practices - Continued is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-A.2: chroot Jail

What is a chroot Jail?

A chroot on Linux systems is an operation that changes the apparent root directory for the current running process and its children. The programs that run in this modified environment cannot access the files outside the designated directory tree. This essentially limits their access to a directory tree and thus they get the name “chroot jail.”

The idea is that you create a directory tree where you copy or link in all the system files needed for a process to run. You then use the chroot system call to change the root directory to be at the base of this new tree and start the process running in that chrooted environment. Since it can’t actually reference paths outside the modified root, it can’t maliciously read or write to those locations.

Why is it required and how is it different from the virtual machines?
This is an operating-system level virtualization and is often used instead of virtual machines to create multiple isolated instances of the host OS. This is a kernel level virtualization and has practically no overhead as compared to Virtual Machines, which are an application layer virtualization. As a result it provides a very good method for creating multiple isolated instances on the same hardware. A virtual machine (VM) is a software implementation of a machine and they often exploit what is known as the Hardware Virtualization to render virtual images of a working operating system.

How to use chroot
The basic command to create a chroot jail is as follows. Steps to create a mini-jail for the ‘bash’ and the ‘ls’ command:

1. Create a directory which will act as the root of the command.

```
pbmac@pbmac-server $ mkdir newroot
pbmac@pbmac-server $ cd newroot
```

2. Create all the essential directories for the command to run: Depending on your operating system, the required directories may change. Logically, we create all these directories to keep a copy of required libraries. To see what all directories are required, see Step 4.

```
pbmac@pbmac-server $ mkdir -p bin lib64/x86_64-linux-gnu lib/x86_64-linux-gnu
```

3.Run the ‘which’ command: Run the **`which`** command to find the location of ls and bash command. After running **`which`** command, copy those binaries in the ‘bin’ directory of our jail. Make sure you don’t have any of these commands aliased. From now on, we would be referring to our directory as **‘**JAILED**’** directory for convenience.

```
pbmac@pbmac-server $ unalias ls          # Required only if you have aliased ls commar
pbmac@pbmac-server $ unalias bash        # Required only if you have aliased bash comr
pbmac@pbmac-server $ cp $(which ls) ./bin/
pbmac@pbmac-server $ cp $(which bash) ./bin/
```

4. Copy appropriate libraries/objects: For the executables in our newroot directory to work we need to copy the appropriate libraries/objects in the JAILED directory. By default, the executable looks at the locations starting with ‘/’. To find the dependencies we use the command ‘ldd.’

```
pbmac@pbmac-server $ ldd $(which bash)
    linux-vdso.so.1 =>  (0x00007ffc75dd4000)
    libtinfo.so.5 => /lib/x86_64-linux-gnu/libtinfo.so.5 (0x00007f6577768000)
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f6577564000)
```

```
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f657719a000)
    /lib64/ld-linux-x86-64.so.2 (0x000055979f3fd000)
```

Run the following commands to create appropriate directories:

```
pbmac@pbmac-server $ cp /lib/x86_64-linux-gnu/libtinfo.so.5 lib/x86_64-linux-gnu/
pbmac@pbmac-server $ cp /lib/x86_64-linux-gnu/libdl.so.2 lib/x86_64-linux-gnu/
pbmac@pbmac-server $ cp /lib/x86_64-linux-gnu/libc.so.6 lib/x86_64-linux-gnu/
pbmac@pbmac-server $ cp /lib64/ld-linux-x86-64.so.2 lib64/
```

So, ALL of the necessary file are copied into the new location to be a part of the chrooted environment. This is necessary because the user running in the chrooted environment cannot access the file system outside of the newroot folder.

5. Sudo chroot: Run this command to change the root to the JAILED directory, along with the path to the shell. By default it will try to load '/bin/sh' shell.

```
pbmac@pbmac-server $  cd ..
pbmac@pbmac-server $  sudo chroot newroot /bin/bash
```

You might face this error while running the chroot command:

```
chroot: failed to run command `/bin/bash': No such file or directory
```

This may be due to two reasons: either the file does not exist (which is obvious), or when the loading library fails or is not available. Double-check if the libraries are in the correct location.

6. A new shell must pop up: It's our newroot bash. We currently have only two commands installed, bash and ls. Fortunately cd and pwd are built-in commands in bash shell, so you can use them as well.

Roam around the directory - try accessing 'cd /../' or something similar. Try to break the jail; probably you won't be able to.

To exit from the jail:

```
pbmac@pbmac-server $ exit
```

The most important and interesting part is that, when you run,

```
pbmac@pbmac-server $ ps aux
```

and find the process, you'll find that there is only one process:

```
root     24958  …  03:21   0:00 /usr/bin/sudo -E chroot newroot/ /bin/bash
```

Interestingly, processes in the newroot shell run as a simple child process of this shell. All the processes inside the JAILED environment are just simple user level processes in the host OS and are isolated by the namespaces provided by the kernel. Thus there is minimal overhead and as an added benefit we get isolation.

Adapted from:
"Linux Virtualization – Chroot Jail" by Pinkesh Badjatiya, Geeks for Geeks is licensed under CC BY-SA 4.0

12-A.2: chroot Jail is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-A.3: Encryption in Linux

Encryption in Linux

In cryptography, encryption is the process of encoding information. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Ideally, only authorized parties can decipher a ciphertext back to plaintext and access the original information. Encryption does not itself prevent interference but denies the intelligible content to a would-be interceptor. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users. Historically, various forms of encryption have been used to aid in cryptography. Early encryption techniques were often utilized in military messaging. Since then, new techniques have emerged and become commonplace in all areas of modern computing. Modern encryption schemes utilize the concepts of public-key and symmetric-key. Modern encryption techniques ensure security because modern computers are inefficient at cracking the encryption.

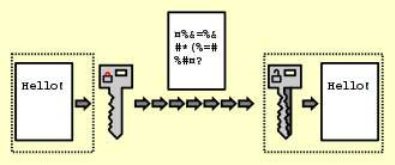


Figure 12 – A.3.1: Using keys in encryption (""Public_key_encryption_keys.svg" by Johannes Landin, Wikimedia Commons is licensed under CC BY-SA 3.00)

Types of Encryption

Data in transit, also referred to as data in motion and data in flight, is defined into two categories: information that flows over the public or untrusted network such as the Internet and data that flows in the confines of a private network such as a corporate or enterprise Local Area Network (LAN).

In addition to the concept of data/file encryption, there are tools that can be used to encrypt drives, partitions or volumes. Disk encryption is a technology which protects information by converting it into unreadable code that cannot be deciphered easily by unauthorized people. Disk encryption uses disk encryption software or hardware to encrypt every bit of data that goes on a disk or disk volume. It is used to prevent unauthorized access to data storage.

Expressions full disk encryption (FDE) or whole disk encryption signify that everything on the disk is encrypted, but the master boot record (MBR) or similar area of a bootable disk, with code that starts the operating system loading sequence, is not encrypted. Some hardware-based full disk encryption systems can truly encrypt an entire boot disk, including the MBR.

LUKS

The Linux Unified Key Setup (LUKS) is a disk encryption specification created by Clemens Fruhwirth in 2004 and was originally intended for Linux.

While most disk encryption software implements different, incompatible, and undocumented formats, LUKS implements a platform-independent standard on-disk format for use in various tools. This not only facilitates compatibility and interoperability among different programs, but also assures that they all implement password management in a secure and documented manner.

The reference implementation for LUKS operates on Linux and is based on an enhanced version of cryptsetup, using dm-crypt as the disk encryption backend.

The `shred` Command

When you delete a file from Linux or from any OS, then the file is not deleted permanently from the hard disk. When a file is deleted it first gets moved to the trash and as soon as you clear off the trash the files get deleted for the file system. But the file is still there in your hard drive and it could be recovered.

When you delete a file permanently or delete it from the trash, the pointer pointing to the file leaves the address of it and the data of the file is sent to a sector in hard disk. There it is considered as unallocated space and it can be recovered easily. The file gets permanently deleted when the OS writes over the sector of the file which was considered as unallocated. So, in order to delete a file completely from hard disk "**shred**" is used in Linux. This command overwrites the contents of a file multiple times, using patterns chosen to maximize the destruction of the residual data, making it harder for even very expensive hardware probing to recover it.

Syntax:

```
shred [ OPTION ] FILE...
```

To overwrite the contents of the file multiple times to make it unrecoverable.

```
pbmac@pbmac-server $ shred filename.txt
```

The `cryptsetup` Command

The **`cryptsetup`** command line tool is the interface with **`dm-crypt`** for creating, accessing and managing encrypted devices. The tool was later expanded to support different encryption types that rely on the Linux kernel device-mapper and the cryptographic modules. The most notable expansion was for the Linux Unified Key Setup (LUKS) extension, which stores all of the needed setup information for **`dm-crypt`** on the disk itself and abstracts partition and key management in an attempt to improve ease of use. Devices accessed via the device-mapper are called blockdevices.

Syntax:

```
cryptsetup [ OPTIONS ] [action] [action-specific-options] [device] [dmname]
```

As LUKS is the default encryption mode, all that is needed to create a new LUKS device with default parameters (`-v` is optional):

```
pbmac@pbmac-server $ cryptsetup -v luksFormat device
```

`cryptsetup` is used to conveniently setup dm-crypt managed device-mapper mappings. For basic (plain) dm-crypt mappings, there are four operations.

Operations	Descriptions
open <name> <device>	Opens (creates a mapping) with <name> backed by device <device>.
remove <name>	Removes an existing mapping <name>.
status <name>	Reports the status for the mapping <name>.
resize <name>	Resizes an active mapping <name>.

In addition, **`cryptsetup`** has LUKS extensions that allow certain actions to be enabled from within the **`cryptsetup`** command.

Extension	Meaning
luksFormat <device> [<key file>]	Initializes a LUKS partition and sets the initial passphrase (for key-slot 0), either via prompting or via <key file>.
isLuks <device>	Returns true, if <device> is a LUKS device, false otherwise.
luksAddKey <device> [<key file with new key>]	Adds a new passphrase. An existing passphrase must be supplied interactively or via --key-file.
luksKillSlot <device> <key slot number>	Wipes the key-slot number <key slot> from the LUKS device.luksKillSlot <device> <key slot number>.
open --type luks <device> <name>	Opens the LUKS device <device> and sets up a mapping <name> after successful verification of the supplied passphrase.

Hashing

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. The values are used to index a fixed-size table called a hash table. Use of a hash function to index a hash table is called hashing or scatter storage addressing.

Hash functions and their associated hash tables are used in data storage and retrieval applications to access data in a small and nearly constant time per retrieval, and storage space only fractionally greater than the total space required for the data or records themselves. Hashing is a computationally and storage space efficient form of data access which avoids the non-linear access time of ordered and unordered lists and structured trees, and the often exponential storage requirements of direct access of state spaces of large or variable-length keys.

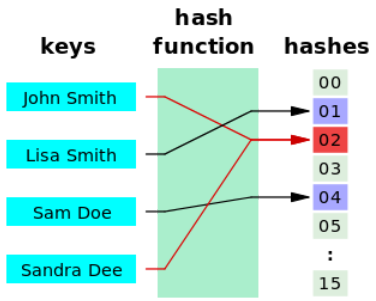


Figure **12 – A.3.1**: Illustration of a hash function mapping four names to the integers 0..15, with one collision. ("Hash_table_4_1_1_0_0_1_0_LL.svg" by Jorge Stolfi, Wikimedia is in the Public Domain, CC0)

Use of hash functions relies on statistical properties of key and function interaction: worst case behavior is intolerably bad with a vanishing small probability, and average case behavior can be nearly optimal (minimal collisions).

Hash functions are related to (and often confused with) checksums, check digits, fingerprints, loss compression, randomization functions, error-correcting codes, and ciphers. Although the concepts overlap to some extent, each one has its own uses and requirements and is designed and optimized differently.

Adapted from:
"Encryption" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Data in transit" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Disk encryption" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"shred command in Linux with Examples" by manav014, Geeks for Geeks is licensed under CC BY-SA 4.0

12-A.3: Encryption in Linux is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-A.4: Security Best Practices

Overall Best Security Practices for Linux

There are numerous steps to take that increase the overall security of a Linux system. Some of the steps should be implemented as a matter of fact, others may engage discussion within an organization. However, this is a discussion that needs to happen soon rather than later - it MUST be a priority. Cybersecurity is increasing in importance with the growing number of tools and technology at the hands of those attempting to infiltrate systems, steal data, and hold systems for ransom.

Following are some best practices:

- Strong passwords - Good options to consider would be: a minimum length of at least 10 characters, require the use of special characters or upper and lowercase letters. The same password should never be used for multiple users or software systems. Configure password expiration dates.
- Generate an SSH Key Pair - While strong passwords can make a difference, even stronger methods of logging into private servers are available. Secure shell (SSH) key pairs, in particular, are worth implementing because these systems are far more difficult to hack through brute force. This may not be popular among the general users, but it is a much more secure option
- Update software at regular intervals - Linux server security includes implementing regular software patches to address potential vulnerabilities. This could be enhanced by enabling automatic updates.
- Avoid installing unnecessary software. There is so much awesome software in the Linux realm, but does it need to be installed?
- Disable booting from external devices to reduce the potential for physical attacks, which can be just as damaging as hacking.
- Close any TCP/IP ports that aren't being used. Open ports may reveal network architecture information while extending attack opportunities.
- Create frequent system backups and test them for reliability.
- root login is disabled on most distributions - DO NOT enable root logins.
- Make sure no non-root accounts have UID set to 0.
- Lock user accounts after login failures.
- Prevent all users from using or reusing same old passwords.
- Avoid using ftp, telnet, and rlogin/rsh services - use secure versions of these utilities.
- Encrypt transmitted data whenever possible with passwords or using keys/certificates.
- Generate SSL/TLS certificates for free using tools like OpenSSL, or purchase them from public certificate authorities.
- Several steps to prevent unwanted console access:
    - Set a BIOS password. Even if the system is rebooted the attacker cannot simply login at boot time.
    - Set GRUB boot loader password. That way the attacker has one more hurdle to get past.
    - Enable authentication for Single-User Mode.
    - Disable interactive Hotkey startup at boot.
    - Setup time-out for login shells
    - Setup screen locking
    - Disable Ctrl+Alt+Delete
    - Verify no accounts have empty passwords

12-A.4: Security Best Practices is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-B.5: Identity and Access Management

EXAM OBJECTIVES COVERED

3.2 Given a scenario, configure and implement appropriate access and authentication methods.
4.3 Given a scenario, analyze and troubleshoot user issues.

Identity and Access Management

Identity and access management (IAM), is a framework of policies and technologies for ensuring that the proper people in an enterprise have the appropriate access to technology resources. IAM systems fall under the overarching umbrellas of IT security and data management.

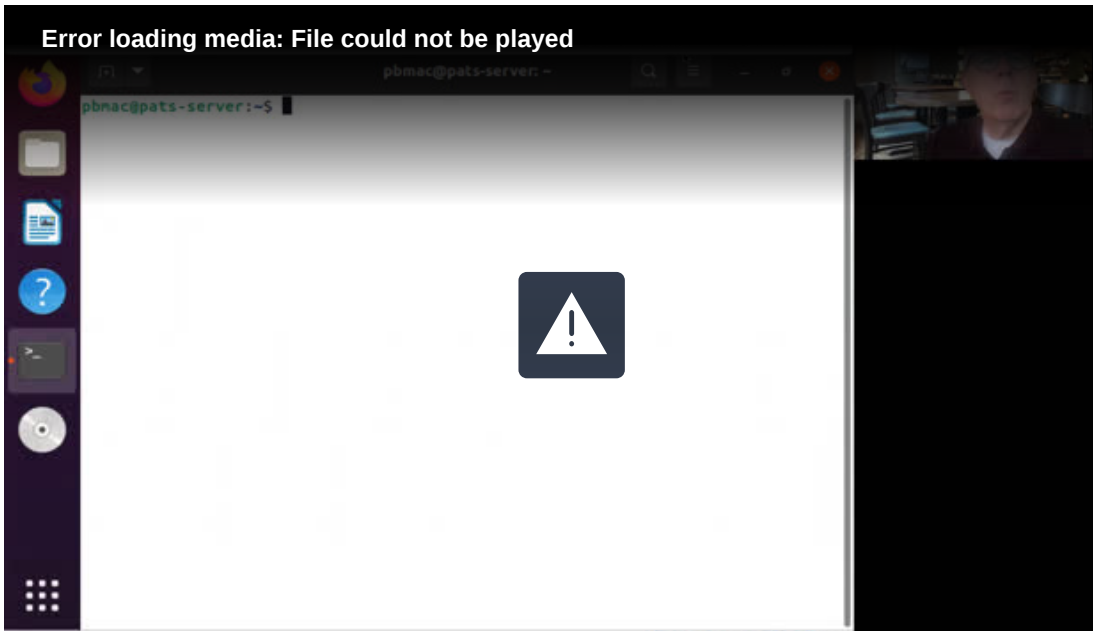
Identity and access management (IAM) is the organizational process for identifying, authenticating and authorizing individuals or groups of people to have access to applications, systems or networks by associating user rights and restrictions with established identities. IAM is the task of controlling information about users on computers. Such information includes information that authenticates the identity of a user, and information that describes data and actions they are authorized to access and/or perform. It also includes the management of descriptive information about the user and how and by whom that information can be accessed and modified. In addition to users, managed entities typically include hardware and network resources and even applications.

Using SSH Authentication

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

SSH provides a secure channel over an unsecured network by using a client–server architecture, connecting an SSH client application with an SSH server. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. SSH is generally used to access Unix-like operating systems, but it can also be used on Microsoft Windows. Windows 10 uses OpenSSH as its default SSH client and SSH server.

Despite popular misconception, SSH is not an implementation of Telnet with cryptography provided by the Secure Sockets Layer (SSL). SSH was designed as a replacement for Telnet and for unsecured remote shell protocols such as the Berkeley rsh and the related rlogin and rexec protocols. Those protocols send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis.



SSH Config Files in Linux

As is so common with Linux tools, there are configuration files for SSH. There are global configuration files that affect all users, and there are individual user configuration files. These files are shown in the table below.

File name	Description
/etc/ssh/ssh_config	This is the global configuration for all users of SSH on this system.
/etc/ssh/ssh_known_hosts	Used to verify the identity of other systems
~/.ssh/config or $HOME/.ssh/config (both are the same location)	The user's configuration file which overrides settings in the global configuration file.
~/.ssh/id_rsa or $HOME/.ssh/id_rsa	The user's private keys generated by ssh-keygen are stored here.
~/.ssh/id_rsa.pub or $HOME/.ssh/id_rsa.pub	The user's public keys generated by ssh-keygen are stored here.
~/.ssh/authorized_keys - ON THE REMOTE SERVER	The user's public keys.
~/.ssh/known_hosts - ON THE LOCAL CLIENT	Contains the known keys for remote hosts that are allowed to connect.

SSH keys

SSH keys are a way to authenticate SSH connections without using a password, either to speed up your access or as a security measure if you turn password access off and ensure only authorized keys are permitted. To create an SSH key, run the command:

```
pbmac@pbmac-server $ ssh-keygen
```

This will create a key-pair (a public and private key) in **~/.ssh/**. Keep the private key (id_rsa) on the PC and never share it. You can share the public key (id_rsa.pub) with others or place it on other servers.

ssh-copy-id

To get the public key over to your remote machine, use the ssh-copy-id. For this to work, you must verify that you have SSH access to the remote machine. If you can't log into the remote host with a password, you can't set up passwordless login either:

```
pbmac@pbmac-server $ ssh-copy-id myuser@192.168.1.20
```

ssh-add

The ssh-add command adds SSH private keys into the SSH authentication agent for implementing single sign-on with SSH.

Using ssh-agent and ssh-add allows the user to use any number of servers, spread across any number of organizations, without having to type in a password every time when moving between servers. This technique is commonly used by system administrators as it allows them to easily move between servers without constantly having to provide a password. It is also frequently found in universities and research institutions for accessing the variety of computing resources that researchers need in their work.

Open the terminal and type the following command and you will see the PID of the ssh-agent:

```
pbmac@pbmac-server $ eval `ssh-agent`
Agent pid 78932
```

Now our ssh-agent is running, and you need to provide the passphrase for your SSH private keys. For example, run the ssh-add command:

```
pbmac@pbmac-server $ ssh-add
Enter passphrase for /home/pbmac/.ssh/id_ed9923
Identity added: /home/pbmac/.ssh/id_ed9923 (pbmac@pbmac_server)
```

The /etc/ssh/sshd_config File

First, make a backup of the sshd_config file by copying it to your home directory, or by making a read-only copy in /etc/ssh:

```
pbmac@pbmac-server $ sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.factory-defaults
pbmac@pbmac-server $ sudo chmod a-w /etc/ssh/sshd_config.factory-defaults
```

Creating a read-only backup in /etc/ssh means you'll always be able to find a known-good configuration when you need it.

Once you've backed up your sshd_config file, you can make changes with any text editor, for example:

```
pbmac@pbmac-server $ sudo gedit /etc/ssh/sshd_config
```

Depending on the distribution of Linux, you can use different editors, and the restart will be different.

```
# Debian / Ubuntu
pbmac@pbmac-server $ sudo service ssh restart

# Red Hat / Fedora / CentOS
pbmac@pbmac-server $ sudo systemctl restart sshd
```

TCP Wrappers

TCP Wrapper is a host-based networking access control list (ACL) used to filter network access to a host. It makes use of two files /etc/hosts.allow and /etc/hosts.deny. This allows you to grant or deny access to a specific host, by making edits to the appropriate file. The hosts.allow file has precedence over the hosts.deny file.

To restrict SSH access to the server, the hosts.deny file would be configured as:

```
ALL : ALL
```

which denies access to all services from all sources. The hosts.allow file would then contain the allowable parameters.

```
sshd : 192.168.
sshd : .example.com
```

The first line allows any host whose IP address begins 192.168 - which is a private ip address, so we know there are local systems. In the second line the period preceding example.com allows any host in the example.com domain.

Adapted from
"Identity management" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Secure Shell Protocol" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"SSH/OpenSSH/Configuring" by kaddidle, Ubuntu Community Help Wiki is licensed under CC BY-SA 4.0

12-B.5: Identity and Access Management is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-B.6: Pluggable Authentication Modules

PAM

Linux Pluggable Authentication Modules (PAM) provide dynamic authentication support for applications and services in a Linux system. Linux PAM is evolved from the Unix Pluggable Authentication Modules architecture.

Linux-PAM separates the tasks of authentication into four independent management groups:

- **Account modules** check that the specified account is a valid authentication target under current conditions. This may include conditions like account expiration, time of day, and that the user has access to the requested service.
- **Authentication modules** verify the user's identity, for example by requesting and checking a password or other secret. They may also pass authentication information on to other systems like a keyring.
- **Password modules** are responsible for updating passwords, and are generally coupled to modules employed in the authentication step. They may also be used to enforce strong passwords.
- **Session modules** define actions that are performed at the beginning and end of sessions. A session starts after the user has successfully authenticated.

PAM offers the following advantages:

- Provides a common authentication scheme that can be used with a wide variety of applications.
- Allows a large amount of flexibility and control over authentication for both system administrators and application developers.
- Allows for the development of programs without creating their own authentication scheme.

PAM Configuration

The PAM configuration files are in the /etc/pam.d/ directory which contains the configuration files for each of the system's PAM-aware application/services.

The syntax of each file in /etc/pam.d/ is similar to that of the main file and is made up of lines of the following form:

```
module-interface control-flag module-name module-arguments
```

This is a example of a rule definition (without module-arguments) found in the /etc/pam.d/sshd file, which disallows non-root logins when /etc/nologin exists:

```
account required pam_nologin.so
```

Understanding PAM Management Groups and Control-flags

PAM authentication tasks are separated into four independent management groups. These groups manage different aspects of a typical user’s request for a restricted service.

A module is associated to each one of these management group types:

- **Account**: provide services for account verification: has the user’s password expired? Is this user permitted access to the requested service?
- **Authentication**: authenticate a user and set up user credentials.
- **Password**: responsible for updating user passwords and works together with authentication modules.
- **Session**: manage actions performed at the beginning of a session and end of a session.

PAM loadable object files (the modules) are to be located in the following directory: **/lib/security/** or **/lib64/security** depending on the architecture.

The supported **control-flags** are:

- **Requisite**: failure instantly returns control to the application indicating the nature of the first module failure.
- **Required**: all these modules are required to succeed for **libpam** to return success to the application.

- **Sufficient**: given that all preceding modules have succeeded, the success of this module leads to an immediate and successful return to the application (failure of this module is ignored).
- **Optional**: the success or failure of this module is generally not recorded.

Lockout Users

Linux Server hardening is one of the important tasks for system administrators. It is recommended that one should enable login or SSH attempts policy, meaning a user's account should be locked automatically after some predetermined numbers of failed login or SSH attempts.

In Linux distributions like CentOS, RHEL and Fedora this is achieved by using PAM module "pam_faillock" and for Debian-like distributions, this can be achieved using "pam_tally2" PAM module.

For CentOS/RHEL/Fedora

Add the following lines in two files /etc/pam.d/password-auth & /etc/pam.d/system-auth,

```
auth     required       pam_faillock.so preauth silent audit deny=3 unlock_time=600
auth     [default=die]  pam_faillock.so authfail audit deny=3 unlock_time=600
account  required       pam_faillock.so
```

Where,

- Audit will enable audit logs for user login attempt in secure log file.
- Deny=3 will lock the user after three unsuccessful login attempts. You can change this number as per your requirement.
- unlock_time=600 means user's account will remain locked for 10 minutes (600 seconds); if you want a user account to be locked forever then set this parameter as "unlock_time=never."

Note: To lock root account as well after n incorrect logins, add "**even_deny_root**" parameter in auth section lines; example is shown below:

```
auth    required       pam_faillock.so preauth silent audit even_deny_root deny=3 unlo
auth    [default=die]  pam_faillock.so authfail audit even_deny_root deny=3 unlock_tir
```

For Debian, Ubuntu and Linux Mint

Add the following line in the file "/etc/pam.d/common-auth"

```
auth    required          pam_tally2.so onerr=fail deny=3 unlock_time=600 audit
```

If you wish to lock root account as well after three incorrect logins then add the following line:

```
auth    required          pam_tally2.so onerr=fail deny=3 unlock_time=600 audit even_
```

Where:

- Onerr=fail - In case of error issue a fail.
- deny=3 - After three unsuccessful login attempts account will be locked.
- unlock_time=600 - Means account will remain locked for 10 minutes or 600 seconds.
- audit - Means audit the logs in audit.log file.
- even_deny_root - Lock the root account after three incorrect logins.
- root_unlock_time=600 - Root account will remain locked for 10 minutes or 600 seconds after three unsuccessful login attempts.

Adapted from:
"Lock User Account After n Failed Login attempts in Linux" by Pradeep Kumar is licensed under CC BY-NC 4.0

12-B.6: Pluggable Authentication Modules is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-B.7: TTY Security & PKI

The /etc/securetty File

The file /etc/securetty contains the names of terminals (one per line, without leading /dev/) which are considered secure for the transmission of certain authentication tokens.

It is used by (some versions of) login(1) to restrict the terminals on which root is allowed to login. See login.defs(5) if you use the shadow suite.

On PAM enabled systems, it is used for the same purpose by pam_securetty(8) to restrict the terminals on which empty passwords are accepted.

It is interesting that most distributions no longer allow root login at all, yet the securetty file is still in existence. It is possible for a system administrator to enable root login - but that is highly discouraged due to the security risks.

Public Key Infrastructure (PKI)

A public key infrastructure (PKI) is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption. The purpose of a PKI is to facilitate the secure electronic transfer of information for a range of network activities such as e-commerce, internet banking and confidential email. It is required for activities where simple passwords are an inadequate authentication method and more rigorous proof is required to confirm the identity of the parties involved in the communication and to validate the information being transferred.

Components of PKI

There is a long list of components that make up the entire PKI. A few of the more important terms and their meanings are listed below.

Certificate Authority (CA):

> Trusted certificates can be used to create secure connections to a server via the Internet. A certificate is essential in order to circumvent a malicious party which happens to be on the route to a target server, which acts as if it were the target. Such a scenario is commonly referred to as a man-in-the-middle attack. The client uses the CA certificate to authenticate the CA signature on the server certificate, as part of the authorizations before launching a secure connection. Usually, client software —for example, browsers—include a set of trusted CA certificates. This makes sense, as many users need to trust their client software. A malicious or compromised client can skip any security check and still fool its users into believing otherwise.

Certificate Signing Request:

> In public key infrastructure (PKI) systems, a certificate signing request (also CSR or certification request) is a message sent from an applicant to a registration authority of the public key infrastructure in order to apply for a digital identity certificate. It usually contains the public key for which the certificate should be issued, identifying information (such as a domain name) and integrity protection (e.g., a digital signature).

Digital Certificate

> A public key certificate, also known as a digital certificate or identity certificate, is an electronic document used to prove the ownership of a public key. The certificate includes information about the key, information about the identity of its owner (called the subject), and the digital signature of an entity that has verified the certificate's contents (called the issuer). If the signature is valid, and the software examining the certificate trusts the issuer, then it can use that key to communicate securely with the certificate's subject. In email encryption, code signing, and e-signature systems, a certificate's subject is typically a person or organization.

Digital Signature

> A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature, where the prerequisites are satisfied, gives a recipient very strong reason to believe that the message was created by a known sender (authentication), and that the message was not altered in transit (integrity).

Digital signatures are a standard element of most cryptographic protocol suites, and are commonly used for software distribution, financial transactions, contract management software, and in other cases where it is important to detect forgery or tampering.

Private Key

Private keys are used by the recipient to decrypt a message that is encrypted using a public key. Since the message is encrypted using a given public key, it can only be decrypted by the matching private key. This establishes the ownership of the private and public key, ensuring the message is only read by the approved parties.

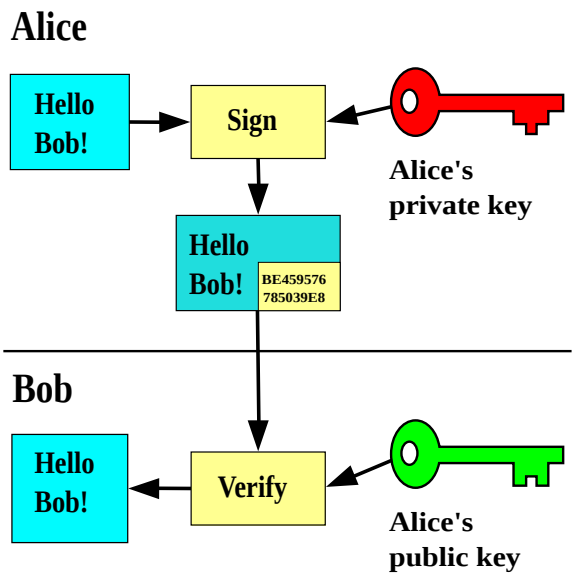


Figure 12 – *B*.7.1:

The message is only signed and not encrypted. 1) Alice signs a message with her private key. 2) Bob can verify that Alice send the message and that the message has not been modified. For verification the public key of Alice must be used.. ("Private_key_signing.svg" by FlippyFlink, Wikimedia Commons is licensed under CC BY-SA 4.0)

Public Key

A Public key is a cryptographic key that can be distributed to the public and does not require secure storage. Messages encrypted by the public key can only be decrypted by the corresponding private key.

Adapted from:
"Certificate authority" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Public key infrastructure" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Certificate signing request" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Public key certificate" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Digital signature" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0

12-B.7: TTY Security & PKI is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-B.8: OpenSSL / VPN

OpenSSL

OpenSSL is a robust, commercial-grade, full-featured Open Source Toolkit for the Transport Layer Security (TLS) protocol formerly known as the Secure Sockets Layer (SSL) protocol. The protocol implementation is based on a full-strength general purpose cryptographic library, which can also be used stand-alone.

The OpenSSL toolkit includes:

- **libssl** - an implementation of all TLS protocol versions up to TLSv1.3 (RFC 8446).

- **libcrypto** - a full-strength general purpose cryptographic library. It constitutes the basis of the TLS implementation, but can also be used independently.

- **openssl** - the OpenSSL command line tool, a Swiss army knife for cryptographic tasks, testing and analyzing. It can be used for:
  - creation of key parameters
  - creation of X.509 certificates, CSRs and CRLs
  - calculation of message digests
  - encryption and decryption
  - SSL/TLS client and server tests
  - handling of S/MIME signed or encrypted mail

Virtual Private Networks

VPN and IPSec

IPsec helps keep private data secure when it is transmitted over a public network. More specifically, IPsec is a group of protocols that are used together to set up secure connections between devices at layer three of the OSI model (the network layer). IPsec accomplishes this by encrypting all messages so that only authorized parties can understand them. IPsec is often used to set up virtual private networks (VPNs).

A VPN is an Internet security service that allows users to access the Internet as though they were connected to a private network. VPNs encrypt Internet communications as well as providing a strong degree of anonymity. VPNs are often used to allow remote employees to securely access corporate data. Meanwhile, individual users may choose to use VPNs in order to protect their privacy.

SSL/TLS

Secure Sockets Layer (SSL) is a protocol for encrypting HTTP traffic, such as connections between user devices and web servers. Websites that use SSL encryption have https:// in their URLs instead of http://. SSL was replaced several years ago by Transport Layer Security (TLS), but the term "SSL" is still in common use for referring to the protocol.

In addition to encrypting client-server communications in web browsing, SSL can also be used in VPNs.

DTLS

Datagram Transport Layer Security (DTLS) is a communications protocol that provides security for datagram-based applications by allowing them to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. The DTLS protocol is based on the stream-oriented Transport Layer Security (TLS) protocol and is intended to provide similar security guarantees. The DTLS protocol datagram preserves the semantics of the underlying transport—the application does not suffer from the delays associated with stream protocols, but because it uses UDP, the application has to deal with packet reordering, loss of datagram and data larger than the size of a datagram network packet. Because DTLS uses UDP rather than TCP, it avoids the "TCP meltdown problem" when being used to create a VPN tunnel.

Adapted from:

"Transport Layer Security" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"IPsec" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0

"Datagram Transport Layer Security" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0

12-B.8: OpenSSL / VPN is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-B.9: Authentication Issues

What Causes Authentication Issues?

Depending on what sort of authentication is configured there can be several issues that can keep users from successfully logging in.

Password Authentication

If a user is attempting to use login strictly with a userid and password:

- Make sure the user is using the correct username.
- User password authentication could be broken for some reason - did someone mess up a system config file?
- Attempt a password reset.
- Check that password authentication is allowed by the server.

Key Error

If the user is seeing some sort of key error:

- Make sure the authorized_keys file and the private key itself have the correct permissions and ownership.
- Check that key-based authentication is allowed by the server.
- Make sure the private key is readable by the SSH client. If the user is using PuTTY, make sure the SSH keys are properly configured for the session. If the user is using an OpenSSH client, be sure the private SSH key has the proper permissions.
- Make sure the authorized_keys file contains the matching public key.
- The private key may no longer supported on the SSH service.

12-B.9: Authentication Issues is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-C.10: SELinux Configuration

EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

4.3 Given a scenario, analyze and troubleshoot user issues.

4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

What is Context-Based Authentication?

In the simplest terms, context-based authentication adds flexibility to multi-factor authentication (MFA). It builds risk assessment capabilities into access decisions by analyzing users' behavior and context, such as which device or network they're logging in from. If used successfully, it helps better protect your resources and enhances employee login experiences—finding that crucial balance between security and usability that most organizations strive for.

In the Linux realm there are two main context-based authentication schemes that are available to system administrators: 1) SELinux and 2) AppArmor. We will take a look at both of these concepts in this section.

What is Mandatory Access Control?

In computer security, mandatory access control (MAC) refers to a type of access control by which the operating system constrains the ability of a subject or initiator to access or generally perform some sort of operation on an object or target. In practice, a subject is usually a process or thread; objects are constructs such as files, directories, TCP/UDP ports, shared memory segments, IO devices, etc. Subjects and objects each have a set of security attributes. Whenever a subject attempts to access an object, an authorization rule enforced by the operating system kernel examines these security attributes and decides whether the access can take place. Any operation by any subject on any object is tested against the set of authorization rules (aka policy) to determine if the operation is allowed.

With mandatory access control, this security policy is centrally controlled by a security policy administrator; users do not have the ability to override the policy and, for example, grant access to files that would otherwise be restricted. By contrast, discretionary access control (DAC), which also governs the ability of subjects to access objects, allows users the ability to make policy decisions and/or assign security attributes.

What is SELinux?

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies, including mandatory access controls (MAC).

SELinux is a set of kernel modifications and user-space tools that have been added to various Linux distributions. Its architecture strives to separate enforcement of security decisions from the security policy, and streamlines the amount of software involved with security policy enforcement. The key concepts underlying SELinux can be traced to several earlier projects by the United States National Security Agency (NSA).

Contexts Within SELinux

SELinux Users

> A SELinux user is an identifier that administrators can use to limit which SELinux roles can be used. A Linux account is mapped to one (and only one) SELinux user, whereas a SELinux user can be linked to multiple roles. As roles define the privileges of a user, this is effectively used to limit the ability of users to run or execute particular applications.

The following table gives a non-exhaustive list of SELinux users which can be found on most SELinux systems.

SELinux User	Description	Used For
unconfined_u	SELinux user meant for unrestricted users. Unconfined users have hardly any restrictions in a SELinux context and are meant for systems where only Internet-facing services should run confined (i.e. the *targeted* SELinux policy store).	All users on a targeted system
root	The SELinux user meant for the root account.	The Linux root account
sysadm_u	SELinux user with direct system administrative role assigned.	Linux accounts that only perform administrative tasks
staff_u	SELinux user for operators that need to run both non-administrative commands (through the staff_r role) and administrative commands (through the sysadm_r role).	Linux accounts used for both end user usage as well as administrative tasks
user_u	SELinux user for non-privileged accounts.	Unprivileged Linux accounts
system_u	Special SELinux user meant for system services.	Not used directly

SELinux Role

Part of SELinux is the Role-Based Access Control (RBAC) security model. The role is an attribute of RBAC. SELinux users are authorized for roles, and roles are authorized for domains. The role serves as an intermediary between domains and SELinux users. The roles that can be entered determine which domains can be entered - ultimately, this controls which object types can be accessed. This helps reduce vulnerability to privilege escalation attacks.

SELinux Type

The type is an attribute of Type Enforcement. The type defines a domain for processes, and a type for files. SELinux policy rules define how types can access each other, whether it be a domain accessing a type, or a domain accessing another domain. Access is only allowed if a specific SELinux policy rule exists that allows it.

Notice the output in the following example. There is information that we haven't seen before. This is the SELinux context information: **`unconfirmed_u`** - is the user; **`object_r`** - the role; and **`default_t`** - is the type.

```
# ls -dZ /web
drwxr-xr-x  root root unconfined_u:object_r:default_t:s0 /web
# ls -lZ /web
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file1
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file2
-rw-r--r--  root root unconfined_u:object_r:default_t:s0 file3
```

SELinux Modes

SELinux has three modes:

- Enforcing: SELinux policy is enforced. SELinux denies access based on SELinux policy rules.

- Permissive: SELinux policy is not enforced. SELinux does not deny access, but denials are logged for actions that would have been denied if running in enforcing mode.

- Disabled: SELinux is disabled. Only DAC rules are used. MAC is not implemented.

Use the /usr/sbin/setenforce command to change between enforcing and permissive mode. Changes made with /usr/sbin/setenforce do not persist across reboots. To change to enforcing mode, as the Linux root user, run the /usr/sbin/setenforce 1 command. To change to permissive mode, run the /usr/sbin/setenforce 0 command. Use the /usr/sbin/getenforce command to view the current SELinux mode.

SELinux Policies

SELinux follows the model of least-privilege; by default everything is denied and then a policy is written that gives each element of the system only the access required to function. This description best describes the **strict** policy. However, such a policy is difficult to write which would be suitable in the wide range of circumstances that a product such as Enterprise Linux is likely to be used in. The end result is that SELinux is likely to cause problems for system administrators and end users and rather than resolve these issues, system administrators may just disable SELinux, thereby defeating the built-in protections.

Gentoo, and several other distributions, support four policy types within SELinux: strict, targeted, mcs and mls.

The differentiation between strict and targeted is based upon the unconfined domain. When loaded, the processes on the system that are not specifically confined within a particular policy module will be part of the unconfined domains, whose purpose is to allow most activities by default (rather than deny by default). As a result, processes that run inside unconfined domains have no restrictions apart from those already enforced by standard Linux security. Although running without the unconfined domains is considered more secure, it will also be more challenging for the administrator to make sure the system still functions properly as there are no policy modules for each and every application "out there."

Next to targeted and strict, administrators can opt for mcs to allow categorization of resources. This is useful on multi-tenant systems such as web servers or virtualization hosts, where multiple processes will be running, most of them in the same security domain, but in different categories. Note though that to take advantage of the additional category support, either the applications themselves (such as the web server or hypervisor tools) need to configure the SELinux categories (so they need to support SELinux) or the administrator will need to script around to start the individual instances with separate categories. Otherwise, mcs is just the same as targeted or strict.

SELinux Commands

Command	Used for
semanage	SELinux Policy Management tool
sestatus	SELinux status tool
setenforce	Modifies the mode SELinux is running in
getenforce	Gets the current mode of SELinux
getsebool	Gets SELinux boolean value(s)
setsebool	Sets SELinux boolean value

Detecting Violations in SELinux

Once SELinux is enforced on a system the violation messages need to be captured. A violation occurs when an attempt is made to access an object, or an action goes against an existing policy. When a system is first put under SELinux, it is possible to get false violations due to policies that might need to be tweaked by it.

Using the **sealert** command the alert messages should be sent to the audit log.

```
pbmac@pbmac-server $ sealert -a /var/log/audit/audit.log
```

For example, let's assume you've got this line in your **/var/log/audit/audit.log** file:

```
type=AVC msg=audit(1415714880.156:29): avc:  denied  { name_connect } for  pid=1349 \
  comm="nginx" dest=8080 scontext=unconfined_u:system_r:httpd_t:s0 \
  tcontext=system_u:object_r:http_cache_port_t:s0 tclass=tcp_socket
```

Execute this command to get a diagnostic:

```
# grep 1415714880.156:29 /var/log/audit/audit.log | audit2why

        Was caused by:
        One of the following booleans was set incorrectly.
        Description:
        Allow httpd to act as a relay

        Allow access by executing:
        # setsebool -P httpd_can_network_relay 1
        Description:
        Allow HTTPD scripts and modules to connect to the network using TCP.

        Allow access by executing:
        # setsebool -P httpd_can_network_connect 1
```

Notice that at the end of the audit2why output there is a suggested command to resolve this problem. Make sure you understand what the command does before you simply execute it. You want to make sure that the command is NOT going to undo some necessary protection for the system.

Adapted from:
"5.5. SELinux Modes" by Multiple Contributors is licensed under CC BY-SA 3.0
"SELinux/Tutorials/How is the policy provided and loaded" by Multiple Contributors is licensed under CC BY-SA 3.0
"SELinux/Policy" by Multiple Contributors is licensed under CC BY-SA 3.0

12-C.10: SELinux Configuration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-C.11: AppArmor

AppArmor

AppArmor ("Application Armor") is a Linux kernel security module that allows the system administrator to restrict programs' capabilities with per-program profiles. Profiles can allow capabilities like network access, raw socket access, and the permission to read, write, or execute files on matching paths. AppArmor supplements the traditional Unix discretionary access control (DAC) model by providing mandatory access control (MAC). It has been partially included in the mainline Linux kernel since version 2.6.36 and its development has been supported by Canonical since 2009.

AppArmor Profiles/Modes

AppArmor confinement is provided via profiles loaded into the kernel via apparmor_parser(8), typically through the /etc/init.d/apparmor SysV initscript, which is used like this:

/etc/init.d/apparmor start
/etc/init.d/apparmor stop
/etc/init.d/apparmor restart

AppArmor can operate in two modes: enforcement, and complain or learning:

· Enforcement - Profiles loaded in enforcement mode will result in enforcement of the policy defined in the profile as well as reporting policy violation attempts to
syslogd.

· Complain - Profiles loaded in "complain" mode will not enforce policy. Instead, it will report policy violation attempts. This mode is convenient for developing
profiles.

Profiles are traditionally stored in files in /etc/apparmor.d/ under filenames with the convention of replacing the / in pathnames with . (except for the root /) so profiles are easier to manage (e.g. the /usr/sbin/nscd profile would be named usr.sbin.nscd).

Profiles are applied to a process at execution time; an already running process cannot be confined. However, once a profile is loaded for a program, that program will be confined on the next execution.

AppArmor also restricts what privileged operations a confined process may execute, even if the process is running as root.

Adjusting Tunables

AppArmor provides a mechanism for tuning your configuration without having to adjust your profiles. These tunables are stored in various files in /etc/apparmor.d/tunables. The most common tunable to adjust is /etc/apparmor.d/tunables/home. If AppArmor is denying access to files in your home directory and your home directory is not in /home, then you need to edit /etc/apparmor.d/tunables/home accordingly. For example, if your home directory is under /exports/home then change:

```
# @{HOMEDIRS} is a space-separated list of where user home directories
# are stored, for programs that must enumerate all home directories on a
# system.
@{HOMEDIRS}=/home/

to be:

# @{HOMEDIRS} is a space-separated list of where user home directories
# are stored, for programs that must enumerate all home directories on a
# system.
@{HOMEDIRS}=/home/ /exports/home/
```

There are other tunables available to you in /etc/apparmor.d/tunables. Most often, issues that are resolved by adjusting a tunable are not considered to be bugs in the profile.

AppArmor Command-line

Command	Used for
aa-status	aa-status will report various aspects of the current state of AppArmor confinement. By default, it displays the same information as if the *--verbose* argument were given.
aa-enabled	Tests whether AppArmor is enabled.
aa-exec	aa-exec is used to launch a program confined by the specified profile and or namespace. If both a profile and namespace are specified the command will be confined by profile in the new policy namespace. If only a namespace is specified, the profile name of the current confinement will be used. If neither a profile or namespace is specified the command will be run using standard profile attachment.
aa-teardown	Unloads all AppArmor profiles.

Adapted From:
"Debugging Apparmor" by Jamie Strandboge, Ubuntu Wiki is licensed under CC BY-SA 4.0
"AppArmor" by Alex Murray, Ubuntu Wiki is licensed under CC BY-SA 4.0

12-C.11: AppArmor is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-D.12: Firewall Configuration in Linux

EXAM OBJECTIVES COVERED

3.5 Given a scenario, implement and configure Linux firewalls.

4.1 Given a scenario, analyze system properties and remediate accordingly.

4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

Firewalls

In computing, a firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted network and an untrusted network, such as the Internet.

Firewalls are categorized as a network-based or a host-based system. Network-based firewalls can be positioned anywhere within a LAN or WAN. They are either a software appliance running on general-purpose hardware, a hardware appliance running on special-purpose hardware, or a virtual appliance running on a virtual host controlled by a hypervisor. Firewall appliances may also offer non firewall functionality, such as DHCP or VPN services. Host-based firewalls are deployed directly on the host itself to control network traffic or other computing resources. This can be a daemon or service as a part of the operating system or an agent application for protection.

Firewall Types

Packet Filter

The first reported type of network firewall is called a packet filter, which inspect packets transferred between computers. The firewall maintains an access control list which dictates what packets will be looked at and what action should be applied, if any, with the default action set to silent discard. Three basic actions regarding the packet consist of a silent discard, discard with Internet Control Message Protocol or TCP reset response to the sender, and forward to the next hop.

Stateful Firewall

A stateful firewall keeps track of the state of network connections, such as TCP streams, UDP datagrams, and ICMP messages, and can apply labels such as LISTEN, ESTABLISHED, or CLOSING. State table entries are created for TCP streams or UDP datagrams that are allowed to communicate through the firewall in accordance with the configured security policy. Once in the table, all RELATED packets of a stored session are streamlined allowed, taking less CPU cycles than standard inspection. Related packets are also permitted to return through the firewall even if no rule is configured to allow communications from that host. If no traffic is seen for a specified time (implementation dependent), the connection is removed from the state table. Applications can send keepalive messages periodically to prevent a firewall from dropping the connection during periods of no activity or for applications which by design have long periods of silence.

Application Firewall

An application firewall is a form of firewall that controls input/output or system calls of an application or service. It operates by monitoring and blocking communications based on a configured policy, generally with predefined rule sets to choose from. The application firewall can control communications up to the application layer of the OSI model, which is the highest operating layer, and where it gets its name. The two primary categories of application firewalls are network-based and host-based.

Next Generation Firewall

A next-generation firewall (NGFW) is a part of the third generation of firewall technology, combining a traditional firewall with other network device filtering functions, such as an application firewall using in-line deep packet inspection (DPI), or an intrusion prevention system (IPS). Other techniques might also be employed, such as TLS/SSL encrypted traffic inspection, website filtering, QoS/bandwidth management, antivirus inspection and third-party identity management integration (i.e., LDAP, RADIUS, Active Directory).

NGFWs include the typical functions of traditional firewalls such as packet filtering, network- and port-address translation (NAT), stateful inspection, and virtual private network (VPN) support. The goal of next-generation firewalls is to include more layers of

the OSI model, improving filtering of network traffic that is dependent on the packet contents.

NGFWs perform deeper inspection compared to stateful inspection performed by the first- and second-generation firewalls. NGFWs use a more thorough inspection style, checking packet payloads and matching signatures for harmful activities such as exploitable attacks and malware.

Adapted from:
"Firewall (computing)" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Stateful firewall" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Application firewall" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Next-generation firewall" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0

12-D.12: Firewall Configuration in Linux is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-D.13: Access Control Lists / iptables

ACL

Access-list (ACL) is a set of rules defined for controlling the network traffic and reducing network attack. ACLs are used to filter traffic based on the set of rules defined for the incoming or outgoing of the network.

ACL features

1. The set of rules defined are matched serial wise i.e matching starts with the first line, then 2nd, then 3rd and so on.
2. The packets are matched only until it matches the rule. Once a rule is matched then no further comparison takes place and that rule will be performed.
3. There is an implicit deny at the end of every ACL, i.e., if no condition or rule matches then the packet will be discarded.

Once the access-list is built, then it should be applied to inbound or outbound of the interface:

- **Inbound access lists –** When an access list is applied on inbound packets of the interface, first the packets will be processed according to the access list and then routed to the outbound interface.
- **Outbound access lists –** When an access list is applied on outbound packets of the interface, first the packet will be routed and then processed at the outbound interface.

Types of ACL

There are two main different types of access-list, namely:

1. **Standard access-list –** These are the access-lists which are made using the source IP address only. These ACLs permit or deny the entire protocol suite. They don't distinguish between the IP traffic such as TCP, UDP, Https etc. By using numbers 1-99 or 1300-1999, a router will understand it as a standard ACL and the specified address as source IP address.
2. **Extended access-list –** These are the ACLs which uses both source and destination IP address. In these type of ACLs, we can also mention which IP traffic should be allowed or denied. These use the range 100-199 and 2000-2699.

Also there are two categories of access-list:

1. **Numbered access-list –** These are the access-lists which cannot be deleted specifically once created i.e., if we want to remove any rule from an access-list then this is not permitted in the case of numbered access list. If we try to delete a rule from the access-list then the whole access -ist will be deleted. The numbered access-list can be used with both standard and extended access-list.
2. **Named access-list –** In this type of access-list, a name is assigned to identify an access-list. It is allowed to delete a named access-list unlike numbered access-lists. Like numbered access-lists, these can be used with both standard and extended access-lists.

Rules for ACL

1. The standard access-list is generally applied close to the destination (but not always).
2. The extended access-list is generally applied close to the source (but not always).
3. We can assign only one ACL per interface per protocol per direction, i.e., only one inbound and outbound ACL is permitted per interface.
4. We can't remove a rule from an access-list if we are using a numbered access-list. If we try to remove a rule then the whole ACL will be removed. If we are using named access-lists then we can delete a specific rule.
5. Every new rule which is added into the access-list will be placed at the bottom of the access-list, therefore before implementing the access-lists, analyze the whole scenario carefully.
6. As there is an implicit deny at the end of every access-list, we should have at least a permit statement in our access-list, otherwise all traffic will be denied.
7. Standard access-lists and extended access-lists cannot have the same name.

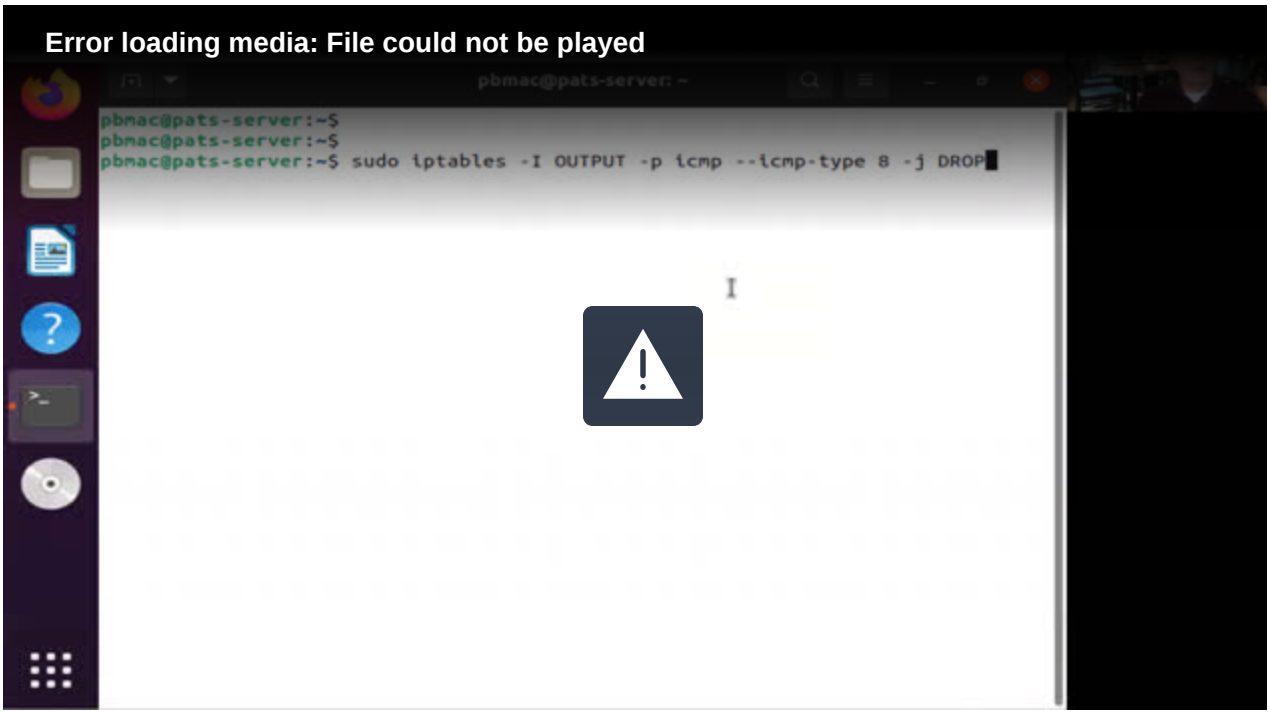
Advantages of ACL

- Improves network performance.

- Provides security as administrator can configure the access-list according to the needs and deny the unwanted packets from entering the network.
- Provides control over the traffic as it can permit or deny according to the needs of network.

iptables

One simple method for adding security is to use iptables. The following video introduces this concept.



iptables is used to inspect, modify, forward, redirect, and/or drop IP packets. The code for filtering IP packets is already built into the kernel and is organized into a collection of tables, each with a specific purpose. The tables are made up of a set of predefined chains, and the chains contain rules which are traversed in order. Each rule consists of a predicate of potential matches and a corresponding action (called a target) which is executed if the predicate is true; i.e., the conditions are matched. If the IP packet reaches the end of a built-in chain, including an empty chain, then the chain's policy target determines the final destination of the IP packet. **iptables** is the user utility which allows you to work with these chains/rules. Most new users find the complexities of Linux IP routing quite daunting, but, in practice, the most common use cases (NAT and/or basic Internet firewall) are considerably less complex.

Tables

iptables contains five tables:

Raw is used only for configuring packets so that they are exempt from connection tracking.
Filter is the default table, and is where all the actions typically associated with a firewall take place.
Nat is used for network address translation (e.g., port forwarding).
Mangle is used for specialized packet alterations.
Security is used for Mandatory Access Control networking rules (e.g., SELinux -- see this article for more details).

In most common use cases you will only use two of these: filter and nat. The other tables are aimed at complex configurations involving multiple routers and routing decisions and are in any case beyond the scope of these introductory remarks.

Chains

Tables consist of chains, which are lists of rules which are followed in order. The default table, filter, contains three built-in chains: INPUT, OUTPUT and FORWARD which are activated at different points of the packet filtering process.

See iptables(8) for a description of built-in chains in other tables.

By default, none of the chains contain any rules. It is up to you to append rules to the chains that you want to use. Chains do have a default policy, which is generally set to ACCEPT, but can be reset to DROP, if you want to be sure that nothing slips through your ruleset. The default policy always applies at the end of a chain only. Hence, the packet has to pass through all existing rules in the chain before the default policy is applied.

User-defined chains can be added to make rulesets more efficient or more easily modifiable.

iptables LOGGING

To enable logging in **iptables**:

```
pbmac@pbmac-server $ iptables -A INPUT -j LOG
```

Define the source IP or range for which log will be created:

```
pbmac@pbmac-server $ iptables -A INPUT -s 192.168.10.0/24 -j LOG
```

To define level of LOG generated by **iptables** use –log-level followed by level number:

```
pbmac@pbmac-server $ iptables -A INPUT -s 192.168.10.0/24 -j LOG --log-level 4
```

Add some prefix in generated Logs, so it will be easy to search for logs in a huge file:

```
pbmac@pbmac-server $ iptables -A INPUT -s 192.168.10.0/24 -j LOG --log-prefix '** SUSP
```

After enabling **iptables** logs, check following log files to view logs generated by **iptables** as per your operating system - **iptables** log to either /var/log/kern.log or /var/log/messages.

Adapted from:
"Access-Lists (ACL)" by saurabhsharma56, Geeks for Geeks is licensed under CC BY-SA 4.0
"Iptables How To" by Gunnar Hjalmarsson, Ubuntu Community Wiki is licensed under CC BY-SA 4.0
"iptables" by Multiple COntributors, Arch Linux Wiki is in the Public Domain, CC0

12-D.13: Access Control Lists / iptables is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-D.14: UFW & firewalld

UFW

The uncomplicated firewall (ufw) is a frontend for **`iptables`** and is particularly well-suited for host-based firewalls. **`ufw`** provides a framework for managing netfilter, as well as a command-line interface for manipulating the firewall. **`ufw`** aims to provide an easy-to-use interface for people unfamiliar with firewall concepts, while at the same time simplifies complicated **`iptables`** commands to help an administrator who knows what he or she is doing. **`ufw`** is an upstream for other distributions and graphical frontends.

Basic Usage

Getting started with **`ufw`** is easy. For example, to enable firewall, allow SSH access, enable logging, and check the status of the firewall, perform:

```
pbmac@pbmac-server $ sudo ufw allow ssh/tcp
pbmac@pbmac-server $ sudo ufw logging on
pbmac@pbmac-server $ sudo ufw enable
pbmac@pbmac-server $ sudo ufw status
Firewall loaded

To                         Action  From
--                         ------  ----
22:tcp                     ALLOW   Anywhere
```

This sets up a default deny (DROP) firewall for incoming connections, with all outbound connections allowed with state tracking.

On Ubuntu Core, simply replace ' **`ufw`** ' with '**`ufw.cmd`**':

```
pbmac@pbmac-server $ sudo ufw.cmd enable
```

Advanced Functionality

As mentioned, the **`ufw`** application is capable of doing anything that **`iptables`** can do. This is achieved by using several sets of rules files, which are nothing more than iptables-restore compatible text files. Fine-tuning **`ufw`** and/or adding additional **`iptables`** commands not offered via the **`ufw`** command is a matter of editing various text files:

- **/etc/default/ufw**: high level configuration, such as default policies, IPv6 support and kernel modules to use.
- **/etc/ufw/before[6].rules**: rules in these files are evaluated before any rules added via the **`ufw`** command.
- **/etc/ufw/after[6].rules**: rules in these files are evaluated after any rules added via the **`ufw`** command.
- **/etc/ufw/sysctl.conf**: kernel network tunables.
- **/var/lib/ufw/user[6].rules** or **/lib/ufw/user[6].rules** (0.28 and later): rules added via the **`ufw`** command (should not normally be edited by hand).
- **/etc/ufw/ufw.conf**: sets whether or not **`ufw`** is enabled on boot, and in 9.04 (ufw 0.27) and later, sets the LOGLEVEL.

- **/etc/ufw/after.init**: initialization customization script runs after **ufw** is initialized (ufw 0.34 and later).

- **/etc/ufw/before.init**: initialization customization script runs before **ufw** is initialized (ufw 0.34 and later).

After modifying any of the above files, activate the new settings with:

```
pbmac@pbmac-server $ sudo ufw disable
pbmac@pbmac-server $ sudo ufw enable
```

firewalld

firewalld provides a dynamically managed firewall with support for network/firewall “zones” to assign a level of trust to a network and its associated connections, interfaces or sources. It has support for IPv4, IPv6, Ethernet bridges and also for IPSet firewall settings. There is a separation of the runtime and permanent configuration options. It also provides an interface for services or applications to add iptables, ip6tables and ebtables rules directly. This interface can also be used by advanced users.

firewalld supports both IPv4 and IPv6 networks and can administer separate firewall zones with varying degrees of trust as defined in zone profiles. Administrators can configure Network Manager to automatically switch zone profiles based on known Wi-Fi (wireless) and Ethernet (wired) networks, but firewalld cannot do this on its own.

Services and applications can use the D-Bus interface to query and configure the firewall. firewalld supports timed rules, meaning the number of connections (or "hits") to a service can be limited globally. There is no support for hit-counting and subsequent connection rejection per source IP; a common technique deployed to limit the impact of brute-force hacking and distributed denial-of-service attacks.

firewalld's command syntax is similar to but more verbose than other iptables front-ends like Ubuntu's Uncomplicated Firewall (ufw). The command-line interface allows managing firewall rulesets for protocol, ports, source and destination; or predefined services by name.

The **firewall-cmd** Command

firewall-cmd is the command line client of the firewalld daemon. firewalld uses the command line utility **firewall-cmd** to configure and manipulate rules. Before we begin to configure this, we need to make sure that the service is running. Using the **systemctl** command, you can enable, disable, start, stop, and restart the firewalld service. For most of these commands, there is no meaningful output from **systemctl**. To verify each of these, you should always check the status of firewalld.

The runtime configuration in firewalld is separated from the permanent configuration. This means that things can get changed in the runtime or permanent configuration.

To get the firewalld state with **firewall-cmd**, use the following command:

```
$ firewall-cmd --state
running
```

Another example: to open port 80/tcp with **firewall-cmd**:

```
firewall-cmd --zone=public --add-port=80/tcp
```

This will open the port 80 with protocol tcp in the public zone of the runtime environment. The runtime environment is only effective until the machine has been rebooted or the firewalld service has been restarted. The zone option can be omitted here if the port should be added to the default zone.

In order to reload firewalld, the command line client **`firewall-cmd`**:

```
firewall-cmd --reload
```

Reload firewall rules and keep state information. Current permanent configuration will become new runtime configuration, i.e., all runtime only changes done until reload are lost with reload if they have not been also put into the permanent configuration.

```
firewall-cmd --complete-reload
```

Reload firewall completely, even netfilter kernel modules. This will most likely terminate active connections, because state information is lost. This option should only be used in case of severe firewall problems, for example, if there are state information problems that no connection can be established with correct firewall rules.

Adapted from:
"Uncomplicated Firewall" by Alex Murray, Ubuntu Wiki is licensed under CC BY-SA 3.0
"Firewalld Documentation" by Unknown, firewalld is in the Public Domain, CC0

12-D.14: UFW & firewalld is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-D.15: Netfilter / IP Forwarding

Netfilter

The netfilter project is a community-driven collaborative FOSS project that provides packet filtering software for the Linux 2.4.x and later kernel series. The netfilter project is commonly associated with iptables.

The netfilter project enables packet filtering, network address [and port] translation (NA[P]T), packet logging, userspace packet queueing and other packet mangling.

The netfilter hooks are a framework inside the Linux kernel that allows kernel modules to register callback functions at different locations of the Linux network stack. The registered callback function is then called back for every packet that traverses the respective hook within the Linux network stack.

iptables is a generic firewalling software that allows you to define rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).

nftables is the successor of iptables; it allows for much more flexible, scalable and performance packet classification.

IP Forwarding

IP forwarding algorithms take into account the size of each packet and the type of service specified in the header, as well as characteristics of the available links to other routers in the network, such as link capacity, utilization rate, and maximum datagram size that is supported on the link. In general, most routing software determines a route through a shortest path algorithm. However, other routing protocols may use other metrics for determining the best path. Based on the metrics required and present for each link, each path has an associated cost. The routing algorithm attempts to minimize the cost when choosing the next hop.

To determine the current state of IP forwarding:

```
pbmac@pbmac-server $ sudo sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

The number 1 indicates that IP forwarding is enabled. The above value is read from the Linux proc file system /proc/sys/net/ipv4/ip_forward file:

```
pbmac@pbmac-server $ cat /proc/sys/net/ipv4/ip_forward
1
```

If it is desired to disable IP forwarding use the command:

```
pbmac@pbmac-server $ sysctl -w net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

The above command actually writes number 0 into the above mentioned file /proc/sys/net/ipv4/ip_forward.

Be aware that the above method does not persist across reboots. To make permanent changes the /etc/sysctl.conf file needs to be changed. Uncomment the line:

```
pbmac@pbmac-server $ net.ipv4.ip_forward=1
```

Make sure it is set to 1, and then save the file. IP forwarding will continue even after a reboot.

IP Set

IP set is used to set up, maintain and inspect so called IP sets in the Linux kernel. Depending on the type of the set, an IP set may store IP(v4/v6) addresses, (TCP/UDP) port numbers, IP and MAC address pairs, IP address and port number pairs, etc. See the set

type definitions below.

Iptables matches and targets referring to sets create references, which protect the given sets in the kernel. A set cannot be destroyed while there is a single reference pointing to it.

Blocking a List of Network Addresses

Start by creating a new "set" of network addresses. This creates a new "hash" set of "net" network addresses named "myset."

```
pbmac@pbmac-server $ ipset create myset hash:net
```

Add any IP address that you'd like to block to the set.

```
pbmac@pbmac-server $ ipset add myset 14.144.0.0/12
pbmac@pbmac-server $ ipset add myset 27.8.0.0/13
pbmac@pbmac-server $ ipset add myset 58.16.0.0/15
pbmac@pbmac-server $ ipset add myset 1.1.1.0/24
```

Finally, configure iptables to block any address in that set. This command will add a rule to the top of the "INPUT" chain to "-m" match the set named "myset" from ipset (--match-set) when it's a "src" packet and "DROP", or block, it.

```
pbmac@pbmac-server $ iptables -I INPUT -m set --match-set myset src -j DROP
```

Blocking a list of IP addresses

Start by creating a new "set" of IP addresses. This creates a new "hash" set of "ip" addresses named "myset-ip."

```
pbmac@pbmac-server $ ipset create myset-ip hash:ip
```

Add any IP address that you'd like to block to the set.

```
pbmac@pbmac-server $ ipset add myset-ip 1.1.1.1
pbmac@pbmac-server $ ipset add myset-ip 2.2.2.2
```

Finally, configure iptables to block any address in that set.

```
pbmac@pbmac-server $ iptables -I INPUT -m set --match-set myset-ip src -j DROP
```

Trusted (or Privileged) Ports

On transport protocols such as TCP, UDP, and SCTP, ports 1-1023 are by default privileged ports. To bind to a privileged port, a process must be running with root permissions. Ports that are greater than 1023 are by default non-privileged. You can use the **`ipadm`** command to extend the range of privileged ports, or you can mark specific ports in the non-privileged range as privileged ports.

Intrusion Detection System (IDS)

An intrusion detection system (IDS) is a device or software application that monitors a network or systems for malicious activity or policy violations. Any intrusion activity or violation is typically reported either to an administrator or collected centrally using a security information and event management (SIEM) system. A SIEM system combines outputs from multiple sources and uses alarm filtering techniques to distinguish malicious activity from false alarms.

IDS types range in scope from single computers to large networks. The most common classifications are network intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS). A system that monitors important operating system

files is an example of an HIDS, while a system that analyzes incoming network traffic is an example of an NIDS. It is also possible to classify IDS by detection approach. The most well-known variants are signature-based detection (recognizing bad patterns, such as malware) and anomaly-based detection (detecting deviations from a model of "good" traffic, which often relies on machine learning). Another common variant is reputation-based detection (recognizing the potential threat according to the reputation scores). Some IDS products have the ability to respond to detected intrusions. Systems with response capabilities are typically referred to as an intrusion prevention system. Intrusion detection systems can also serve specific purposes by augmenting them with custom tools, such as using a honeypot to attract and characterize malicious traffic.

DenyHosts

DenyHosts is a log-based intrusion-prevention security tool for SSH servers written in Python. It is intended to prevent brute-force attacks on SSH servers by monitoring invalid login attempts in the authentication log and blocking the originating IP addresses.

DenyHosts checks the end of the authentication log for recent failed login attempts. It records information about their originating IP addresses and compares the number of invalid attempts to a user-specified threshold. If there have been too many invalid attempts it assumes a dictionary attack is occurring and prevents the IP address from making any further attempts by adding it to /etc/hosts.deny on the server. DenyHosts 2.0 and above support centralized synchronization, so that repeat offenders are blocked from many computers. The site denyhosts.net gathers statistics from computers running the software.

DenyHosts is restricted to connections using IPv4. It does not work with IPv6.

DenyHosts may be run manually, as a daemon, or as a cron job.

Configuration

In a Debian or Ubuntu Linux based system, the default running mode is daemon mode and the configuration file is /etc/denyhosts.conf:

```
pbmac@pbmac-server $ sudo vi /etc/denyhosts.conf
```

Make sure SECURE_LOG set as follows:

```
SECURE_LOG = /var/log/auth.log
```

HOSTS_DENY set as follows:

HOSTS_DENY = **/etc/hosts.deny**

To block only sshd:

```
BLOCK_SERVICE  = sshd
```

Set deny threshold limit for login attempts:

```
DENY_THRESHOLD_INVALID = 5
DENY_THRESHOLD_VALID = 10
DENY_THRESHOLD_ROOT = 1
DENY_THRESHOLD_RESTRICTED = 1
```

To block incoming connections using the Linux firewall IPTABLES:

```
IPTABLES = /sbin/iptables
```

Save and close the file.

Fail2Ban

Fail2Ban is an intrusion prevention software framework that protects computer servers from brute-force attacks. Written in the Python programming language, it is able to run on POSIX systems that have an interface to a packet-control system or firewall installed locally, for example, iptables or TCP Wrapper.

Fail2Ban operates by monitoring log files (e.g., /var/log/auth.log, /var/log/apache/access.log, etc.) for selected entries and running scripts based on them. Most commonly this is used to block selected IP addresses that may belong to hosts that are trying to breach the system's security. It can ban any host IP address that makes too many login attempts or performs any other unwanted action within a time frame defined by the administrator. Includes support for both IPv4 and IPv6

Adapted from:
"The netfilter.org project" by About the netfilter/iptables project, The netfilter.org project is in the Public Domain, CC0
"ipset" by Multiple Contributors, Arch Linux Wiki is licensed under CC BY-SA 3.0
"Intrusion detection system" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 4.0
"DenyHosts" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Fail2ban" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0

12-D.15: Netfilter / IP Forwarding is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-E.16: Linux Logging

EXAM OBJECTIVES COVERED

3.4 Given a scenario, implement logging services.

Log Files

One of the things which makes GNU/Linux a great operating system is that virtually anything and everything happening on and to the system may be logged in some manner. This information is invaluable for using the system in an informed manner, and should be one of the first resources you use to trouble-shoot system and application issues. The logs can tell you almost anything you need to know, as long as you have an idea where to look first.

Your Ubuntu system provides vital information using various system log files. These log files are typically plain ASCII text in a standard log file format, and most of them sit in the traditional system log subdirectory /var/log. Many are generated by the system log daemon syslogd on behalf of the system and certain applications, while some applications generate their own logs by writing directly to files in /var/log.

Log File Location

Location	Description
/var/log/messages	General message and system related stuff
/var/log/auth.log	Authentication logs
/var/log/kern.log	Kernel logs
/var/log/cron.log	Crond logs (cron job)
/var/log/maillog	Mail server logs
/var/log/qmail/	Qmail log directory (more files inside this directory)
/var/log/httpd/	Apache access and error logs directory
/var/log/lighttpd/	Lighttpd access and error logs directory
/var/log/boot.log	System boot log
/var/log/mysqld.log	MySQL database server log file
/var/log/secure or /var/log/auth.log	Authentication log
/var/log/utmp or /var/log/wtmp	Login records file
/var/log/yum.log	Yum command log file

The `logrotate` Command

The **`logrotate`** utility is designed to simplify the administration of log files on a system which generates a lot of log files. **`Logrotate`** allows for the automatic rotation compression, removal and mailing of log files. **`Logrotate`** can be set to handle a log file daily, weekly, monthly or when the log file gets to a certain size.

By default, **`logrotate's`** rotation consists of renaming existing log files with a numerical suffix, then recreating the original empty log file. For example, /var/log/syslog.log is renamed /var/log/syslog.log.1. If /var/log/syslog.log.1 already exists from a previous rotation, it is first renamed /var/log/syslog.log.2.

Configuration

The primary configuration file for **`logrotate`** which sets default parameters is /etc/logrotate.conf; additional application-specific configuration files are included from the /etc/logrotate.d directory. Values set in application-specific configuration files override

those same parameters in the primary configuration file. See logrotate.conf(5) for configuration examples and a reference of available directives.

To verify if **logrotate** works correctly, run it in debug mode; in this mode it does nothing except producing debug output:

```
pbmac@pbmac-server $ logrotate --debug /etc/logrotate.conf
```

Usage

logrotate is usually run through the systemd service: logrotate.service.

To run **logrotate** manually:

```
pbmac@pbmac-server $ logrotate /etc/logrotate.conf
```

To rotate a single log file:

```
pbmac@pbmac-server $ logrotate /etc/logrotate.d/mylog
```

To simulate running your configuration file (dry run):

```
pbmac@pbmac-server $ logrotate --debug /etc/logrotate.d/mylog
```

To force running rotations even when conditions are not met, run:

```
pbmac@pbmac-server $ logrotate -vf /etc/logrotate.d/mylog
```

Linux rsyslogd

Rsyslogd is a system utility providing support for message logging. Support of both internet and Unix domain sockets enables this utility to support both local and remote logging.

Rsyslogd(8) is derived from the sysklogd package which in turn is derived from the stock BSD sources.

Rsyslogd provides a kind of logging that many modern programs use. Every logged message contains at least a time and a hostname field, normally a program name field, too, but that depends on how trusty the logging program is. The rsyslog package supports free definition of output formats via templates. It also supports precise timestamps and writing directly to databases. If the database option is used, tools like phpLogCon can be used to view the log data.

While the rsyslogd sources have been heavily modified, a couple of notes are in order. First of all there has been a systematic attempt to ensure that rsyslogd follows its default, standard BSD behavior. Of course, some configuration file changes are necessary in order to support the template system. However, rsyslogd should be able to use a standard syslog.conf and act like the original syslogd. However, an original syslogd will not work correctly with a rsyslog-enhanced configuration file. At best, it will generate funny looking file names. The second important concept to note is that this version of rsyslogd interacts transparently with the version of syslog found in the standard libraries. If a binary linked to the standard shared libraries fails to function correctly we would like an example of the anomalous behavior.

The main configuration file /etc/rsyslog.conf or an alternative file, given with the **-f** option, is read at startup. Any lines that begin with the hash mark ("#") and empty lines are ignored. If an error occurs during parsing the error element is ignored. It tries to parse the rest of the line.

Third Party Logging Tools

There are other Linux logging tools available. Some of them are open source, and some of them are commercially licensed. They can be found by doing a simple web search. Some of these tools implement some different type of options. Make sure you understand how to install and configure these tools if you decide to use them.

Adapted From:
"Logrotate" by Multiple Contributors, Arch Linux Wiki is licensed under CC BY-SA 3.0
"RSyslog Documentation" by Multiple Contributors is in the Public Domain, CC0

12-E.16: Linux Logging is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-E.17: journalctl & last Commands

The **journalctl** Command

The **journalctl** command is used for viewing logs collected by **systemd**. The systemd-journald service is responsible for systemd's log collection, and it retrieves messages from the kernel, **systemd** services, and other sources.

These logs are gathered in a central location, which makes them easy to review. The log records in the journal are structured and indexed, and as a result **journalctl** is able to present your log information in a variety of useful formats.

By default the systemnd journal only stores the logs in memory, therefore, the logs do not persist across reboots. In order to have any persistence in the logs, the /var/log/journal directory must be created. The logs are then kept in that directory.

```
pbmac@pbmac-server $ journalctl
-- Logs begin at Sat 2020-03-21 15:10:38 PDT, end at Mon 2020-11-16 14:59:05 PST
Mar 21 15:10:38 pbmac-server systemd-journald[294]: Runtime journal (/run/log/jo
Mar 21 15:10:38 pbmac-server kernel: Initializing cgroup subsys cpuset
Mar 21 15:10:38 pbmac-server kernel: Initializing cgroup subsys cpu
Mar 21 15:10:38 pbmac-server kernel: Initializing cgroup subsys cpuacct
Mar 21 15:10:38 pbmac-server kernel: Linux version 4.4.0-176-generic (buildd@lgw
Mar 21 15:10:38 pbmac-server kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.4.
Mar 21 15:10:38 pbmac-server kernel: KERNEL supported cpus:
Mar 21 15:10:38 pbmac-server kernel:   Intel GenuineIntel
Mar 21 15:10:38 pbmac-server kernel:   AMD AuthenticAMD
Mar 21 15:10:38 pbmac-server kernel:   Centaur CentaurHauls
Mar 21 15:10:38 pbmac-server kernel: x86/fpu: Supporting XSAVE feature 0x01: 'x8
Mar 21 15:10:38 pbmac-server kernel: x86/fpu: Supporting XSAVE feature 0x02: 'SS
Mar 21 15:10:38 pbmac-server kernel: x86/fpu: Enabled xstate features 0x3, conte
Mar 21 15:10:38 pbmac-server kernel: e820: BIOS-provided physical RAM map:
Mar 21 15:10:38 pbmac-server kernel: BIOS-e820: [mem 0x0000000000000000-0x000000
```

journalctl allows for the filtering of the output by specific fields. If there are many messages to display or filtering of large time span has to be done, the output of this command can be extensively delayed.

Examples:

- Show all messages from this boot:

  ```
  pbmac@pbmac-server $ journalctl -b
  ```

  However, often one is interested in messages not from the current, but from the previous boot (e.g. if an unrecoverable system crash happened). This is possible through optional offset parameter of the `-b` flag: `journalctl -b -0` shows messages from the current boot, `journalctl -b -1` from the previous boot, `journalctl -b -2` from the second previous and so on – you can see the list of boots with their numbers by using `journalctl --list-boots` . See journalctl(1) for a full description; the semantics are more powerful than indicated here.
- Include explanations of log messages from the message catalog where available:

  ```
  pbmac@pbmac-server $ journalctl -x
  ```

  Note that this feature should not be used when attaching logs to bug reports and support threads, so as to limit extraneous output. You can list all known catalog entries by running journalctl --list-catalog.
- Show all messages from date (and optional time):

```
pbmac@pbmac-server $ journalctl --since="2012-10-30 18:17:16"
```

- Show all messages since 20 minutes ago:

```
pbmac@pbmac-server $ journalctl --since "20 min ago"
```

The `last` Command

The **`last`** command in Linux is used to display the list of all the users logged in and out since the file /var/log/wtmp was created. One or more usernames can be given as an argument to display their login in (and out) time and their host-name.

```
pbmac@pbmac-server $ last
pbmac    :0           :0               Mon Nov 16 09:48   still logged in
reboot   system boot  4.15.0-91-generi Mon Nov 16 09:46   still running
pbmac    :0           :0               Fri Nov  6 11:20 - crash (9+22:26)
reboot   system boot  4.15.0-91-generi Fri Nov  6 10:53   still running
```

12-E.17: journalctl & last Commands is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-F.18: Backup and Restore in Linux

EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.
2.6 Given a scenario, back up, restore, and compress files.

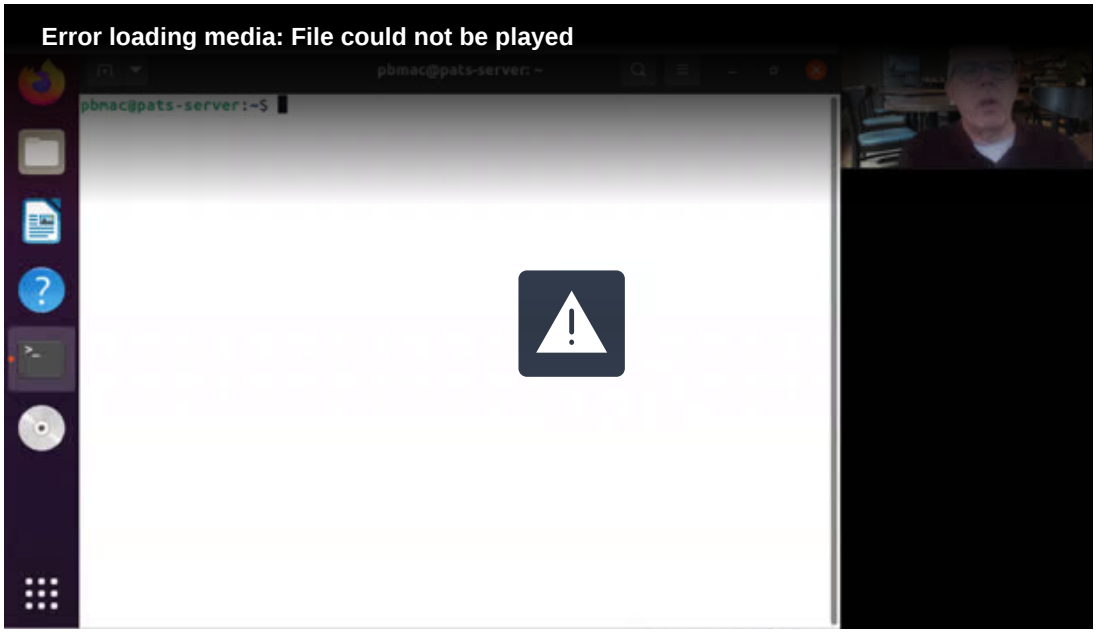
System Backup

In information technology, a backup, or data backup is a copy of computer data taken and stored elsewhere so that it may be used to restore the original after a data loss event. The verb form, referring to the process of doing so, is "back up," whereas the noun and adjective form is "backup." Backups can be used to recover data after its loss from data deletion or corruption, or to recover data from an earlier time. Backups provide a simple form of disaster recovery; however not all backup systems are able to reconstitute a computer system or other complex configuration such as a computer cluster, active directory server, or database server.

A backup system contains at least one copy of all data considered worth saving. The data storage requirements can be large. An information repository model may be used to provide structure to this storage. There are different types of data storage devices used for copying backups of data that is already in secondary storage onto archive files. There are also different ways these devices can be arranged to provide geographic dispersion, data security, and portability.

Data is selected, extracted, and manipulated for storage. The process can include methods for dealing with live data, including open files, as well as compression, encryption, and de-duplication. Additional techniques apply to enterprise client-server backup. Backup schemes may include dry runs that validate the reliability of the data being backed up. There are limitations and human factors involved in any backup scheme.

The video covers the basics of backup/restore in Linux.



Full Backup

A full backup is when every single file and folder in the system is backed up. A full backup takes longer and requires more space than other types of backups but the process of restoring lost data from backup is much faster.

Incremental Backup

Subsequent backups only store changes that were made since the previous backup. The process of restoring lost data from backup is longer but the backup process is much quicker.

Differential Backup

The initial backup is full and subsequent backups only store changes made to files since the last backup. This type of backup requires more storage space than incremental backup does, but it also allows for a faster restore time.

Mirror Backup

An exact copy is made of the source data. The advantage of mirror backup is that there is no storing of old, obsolete files. When obsolete files are deleted, they disappear from the mirror backup as well when the system backs up. The downside to mirror backup is that if files are accidentally deleted, they can be lost from the backup as well if the deletion isn't discovered before the next scheduled backup.

Difference Between Incremental Backup and Differential Backup

1. Incremental Backup

Incremental backup only preserves the data that has been changed from the last full backup or incremental backup. This type of backup uses less storage and can do backup and recovery quickly. The data in this type of backup contains redundancy as recovery includes last full backup and all the incremental backups.

2. Differential Backup

Differential backup only preserves data change from the last full backup. This type of backup uses more memory as it saves all the data changes since last full backup. It contains less redundant data and recovery is easy but the time needed for creating backup is high.

Difference between Backup and Snapshot

1. Backup

   Backup generally suggests the duplicate of your data. When a backup is initiated, it generates copies of your files, comprising files pertaining to your website and mailboxes. These copies are conventionally kept in a different location than the original content, thus making them ideal for disaster rehabilitation. Backups are the mechanism that could take minutes, hours, or days to complete, depending on the data. This conveys that the data at the end of the backup may not be compatible with the data at the time when the backup initiated. Backups are planned to be stored for long periods of time and, if they are stored off server, they can be used to restore servers after a server failure.

2. Snapshot

   Snapshot refers to an instantaneous "picture" of your server's file system at a certain period of time. This picture apprehends the entire file system as it was when the snapshot was taken. When a snapshot is accustomed to restore the server, the server will revert to exactly how it was at the time of the snapshot. Snapshots are designed for short term storage. When space departs, new snapshots eventually overwrite older ones. For this reason, snapshots are usually only good if you want to revert to a recent version of your server.

Difference Between Backup and Snapshot:

Backup	Snapshot
Backup can be stored in another location, same server, or even the same drive in this case.	Snapshots can be stored only in the same location where original data is present.
Backup may have differences based on when the backup started and ended.	Snapshot is a "picture" of your server at present period of time.
It is a long term process and might take a lot of time to complete the procedure.	It is a short term process and takes much less time, as compared to backup, to complete the procedure.
Backup comprises only the file system.	Snapshots comprises different types of systems like the files, software, and settings of that type.
Backup takes a long time to copy data.	Snapshot takes much less time to copy the data.
In case of SQL server, backup is a paid service. A person can use this service to automatically backup the most recent data every night.	In case of SQL server, snapshot is a free feature that is used to manually create duplicate images of the servers. One can easily generate them at any time.
Backup is not a backup unless the data exists in several distinct locations. Backups should be easily and quickly restored. Backups should be verifiable.	Snapshot by itself is not a backup. But it can be used as an essential part of the backup process. Snapshot is used as a part of the data movement process to a backup file. It is removed when the backup job is completed.

Adapted from:
"Backup" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Difference between Backup and Snapshot" by sommprasad, Geeks for Geeks is licensed under CC BY-SA 4.0
"Difference between incremental backup and differential backup" by kapilsparshi, Geeks for Geeks is licensed under CC BY-SA 4.0

12-F.18: Backup and Restore in Linux is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-F.19: tar / cpio / dd

The `tar` Command

The Linux command **`tar`** stands for tape archive. It is used to create Archive and extract the Archive files. The **`tar`** command in Linux is one of the most important commands which is used for archiving. We can use the Linux **`tar`** command to create compressed or uncompressed Archive files and also maintain and modify them.

Syntax:

```
tar [ OPTIONS ] [archive-file] [file or directory to be archived]
```

Options:

Options	Option Meaning
`-c`	Creates archive
`-x`	Extracts the archive
`-f`	Creates archive with given filename
`-t`	Displays or lists files in archived file
`-v`	Displays Verbose Information
`-z`	Zip, tells **`tar`** command that creates the tar file to compress the created file using gzip

An example of creating an uncompressed tar file using option -cvf: This command creates a tar file called file.tar which is the archive of all .c files in current directory.

```
pbmac@pbmac-server $ tar -cvf file.tar *.c
```

Because the -v option was specified, each filename is echoed to the screen as it is included in the tar file:

```
os2.c
os3.c
os4.c
```

Extracting files from archive using option -xvf: This command extracts files from tar file.

```
pbmac@pbmac-server $ tar -xvf file.tar
```

Because the -v option was specified, each filename is echoed to the screen as it is extracted from the tar file:

```
os2.c
os3.c
```

```
os4.c
```

The following command shows gzip compression on the tar file, using option -z: This command creates a tar file called Cfiles.tar.gz of all the .c files in the current directory.

```
pbmac@pbmac-server $ tar -cvzf Cfiles.tar.gz *.c
```

Showing the extraction of a gzip tar Cfiles.tar.gz using option -xvzf.

```
pbmac@pbmac-server $ tar -xvzf Cfiles.tar.gz
```

The `dar` Command

The **`dar`** command can store a backup in several files (called "slices" in the following) of a given size, eventually pausing or running a user command/script before starting the next slice. This can allow, for example, the burning of the last generated slice on a DVD-R(W), Blue-ray Disk, or changing of USB key before continuing on the next one. Like its grand-brother, the great "**`tar`**" command, **`dar`** may also use compression, with the difference that compression is used inside the archive to be able to have compressed slices of the defined size.

But the most important feature of **`dar`** is its ability to make differential, incremental and decremental backups. In other words, backups that contain only new files or files that have changed from a backup of reference. Binary delta is available but not activated by default: in combination with differential and incremental backups, it leads not only to not save a file that has not changed (something **`dar`** does without binary delta), but also to only save an rsync patch of any modified file, which lead to even smaller backups.

The `cpio` Command

The **`cpio`** command is a tool for creating and extracting archives, or copying files from one place to another. It handles a number of **`cpio`** formats as well as reading and writing tar files.

The following archive formats are supported: binary, old ASCII, new ASCII, crc, HPUX binary, HPUX old ASCII, old tar, and POSIX.1 tar. The tar format is provided for compatibility with the tar program. By default, **`cpio`** creates binary format archives for compatibility with older **`cpio`** programs. When extracting from archives, **`cpio`** automatically recognizes which kind of archive it is reading and can read archives created on machines with a different byte-order.

- Copy-out mode: Copy files named in name-list to the archive.

  ```
  cpio -o < name-list > archive
  ```

  To create a *.cpio archive with selected files: We can create *.cpio files containing specific files with the help of **`cpio`** command. In the example we are using .txt files.

  ```
  pbmac@pbmac-server $ find . -iname "*.txt" | cpio -ov > archive
  ```

- Copy-in mode: Extract files from the archive.

  ```
  cpio -i < archive
  ```

  To extract a *.cpio file: We can extract *.cpio files containing files and directory with the help of **`cpio`** command.

  ```
  cpio -iv < archive
  ```

- Copy-pass mode: Copy files named in name-list to destination-directory.

```
cpio -p destination-directory < name-list
```

The **dd** Command

The **dd** is a command-line utility for Linux whose primary purpose is to convert and copy files.

- On Linux, device drivers for hardware (such as hard disk drives) and special device files (such as /dev/zero and /dev/random) appear in the file system just like normal files.
- **dd** can also read and/or write from/to these files, provided that function is implemented in their respective drivers
- As a result, **dd** can be used for tasks such as backing up the boot sector of a hard drive and obtaining a fixed amount of random data.
- The **dd** program can also perform conversions on the data as it is copied, including byte order swapping and conversion to and from the ASCII and EBCDIC text encodings.

Some practical examples on **dd** command :

1. To backup the entire hard disk: To back up an entire copy of a hard disk to another hard disk connected to the same system, execute the **dd** command as shown. In this **dd** command example, the UNIX device name of the source hard disk is /dev/hda, and device name of the target hard disk is /dev/hdb.

   ```
   pbmac@pbmac-server $ dd if = /dev/sda of = /dev/sdb
   ```

   - “if” represents inputfile, and “of” represents output file. So the exact copy of /dev/sda will be available in /dev/sdb.
   - If there are any errors, the above command will fail. If you give the parameter “conv=noerror” then it will continue to copy if there are read errors.
   - Input file and output file should be mentioned very carefully. Just in case you mention source device in the target and vice versa, you might lose all your data.
   - **To copy hard drive to hard drive using dd command given below, sync option allows you to copy everything using synchronized I/O.**

     ```
     pbmac@pbmac-server $ dd if = /dev/sda of = /dev/sdb conv=noerror, sync
     ```

2. To backup a partition : You can use the device name of a partition in the input file, and in the output either you can specify your target path or image file as shown in the **dd** command.

   ```
   pbmac@pbmac-server $ dd if=/dev/hda1 of=~/partition.img
   ```

3. To create an image of a hard disk: Instead of taking a backup of the hard disk, you can create an image file of the hard disk and save it in other storage devices. There are many advantages of backing up your data to a disk image, one being the ease of use. This method is typically faster than other types of backups, enabling you to quickly restore data following an unexpected catastrophe. It creates the image of a hard disk /dev/hda.

   ```
   pbmac@pbmac-server $ dd if = /dev/hda of = ~/hdadisk.img
   ```

4. To restore using the hard disk image: To restore a hard disk with the image file of a another hard disk, the following **dd** command can be used:

   ```
   pbmac@pbmac-server $ dd if = hdadisk.img of = /dev/hdb
   ```

   The image file hdadisk.img file is the image of a /dev/hda, so the above command will restore the image of /dev/hda to /dev/hdb.

5. **dd** command reads one block of input and processes it and writes it into an output file. You can specify the block size for input and output file. In the above **dd** command example, the parameter “bs” specifies the block size for the both the input and output file. So **dd** uses 2048 bytes as a block size in the above command.

Adapted from:
"tar command in Linux with examples" by Akansh Gupta., Geeks for Geeks is licensed under CC BY-SA 4.0
"cpio command in Linux with Examples" by Mandeep_Sheoran, Geeks for Geeks is licensed under CC BY-SA 4.0
"‘dd’ command in Linux" by Kishlay Verma, Geeks for Geeks is licensed under CC BY-SA 4.0

12-F.19: tar / cpio / dd is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-F.20: mirrorvg / rsync / scp

The `mirrorvg` Command

The **`mirrorvg`** command takes all the logical volumes on a given volume group and mirrors those logical volumes. This same functionality may also be accomplished manually if you execute the **`mklvcopy`** command for each individual logical volume in a volume group. As with **`mklvcopy`**, the target physical drives to be mirrored with data must already be members of the volume group. To add disks to a volume group, run the **`extendvg`** command.

By default, **`mirrorvg`** attempts to mirror the logical volumes onto any of the disks in a volume group. If you wish to control which drives are used for mirroring, you must include the list of disks in the input parameters, physicalvolume. Mirror strictness is enforced. Additionally, **`mirrorvg`** mirrors the logical volumes, using the default settings of the logical volume being mirrored. If you wish to violate mirror strictness or affect the policy by which the mirror is created, you must execute the mirroring of all logical volumes manually with the **`mklvcopy`** command.

When **`mirrorvg`** is executed, the default behavior of the command requires that the synchronization of the mirrors must complete before the command returns to the user. If you wish to avoid the delay, use the -S or -s option. Additionally, the default value of 2 copies is always used. To specify a value other than 2, use the -c option.

Offsite Data Protection

Off-site data protection, or vaulting, is the strategy of sending critical data out of the main location (off the main site) as part of a disaster recovery plan. Data is usually transported off-site using removable storage media such as magnetic tape or optical storage. Data can also be sent electronically via a remote backup service, which is known as electronic vaulting or e-vaulting. Sending backups off-site ensures systems and servers can be reloaded with the latest data in the event of a disaster, accidental error, or system crash. Sending backups off-site also ensures that there is a copy of pertinent data that isn't stored on-site.

Although some organizations manage and store their own off-site backups, many choose to have their backups managed and stored by third parties who specialize in the commercial protection of off-site data.

The `scp` Command

The **`scp`** (secure copy) command in Linux system is used to copy file(s) between servers in a secure way. The SCP command or secure copy allows secure transferring of files in between the local host and the remote host or between two remote hosts. It uses the same authentication and security as it is used in the Secure Shell (SSH) protocol. SCP is known for its simplicity, security and pre-installed availability.

Syntax:

scp [OPTIONS }

Options:

Options	Option Meaning
-P port	Specifies the port to connect on the remote host.
-p	Preserves modification times, access times, and modes from the original file.
-q	Disables the progress meter.
-r	Recursively copies entire directories.
-S program	Name of program to use for the encrypted connection. The program must understand ssh(1) options.
-v	Verbose mode. Causes scp and ssh to print debugging messages about their progress. This is helpful in debugging connection, authentication, and configuration problems.

The use of **scp** is pretty simple.

- Select another cipher as to encrypt files: By default, **scp** is using the "AES-128" to encrypt files. If you want to change to any another cipher to encrypt it, you can do that by using "-c" parameter.

```
pbmac@pbmac-server $ scp -c 3des-cbc myFile.ppdf pbmac@pbmac-server.com:/PDF
```

- To specify a specific port to use with **scp**: Usually, **scp** is using port 22 as a default port. But for security reasons, you can change the port into another port. For example, we are going to use port 2249. Then the command needs to be like this.

```
pbmac@pbmac-server $ scp -P 2249 Label.pdf pbmac@192.168.1.24:.
```

The **sftp** Command

The SSH File Transfer Protocol (also Secure File Transfer Protocol, or SFTP) is a network protocol that provides file access, file transfer, and file management over any reliable data stream. It was designed by the Internet Engineering Task Force (IETF) as an extension of the Secure Shell protocol (SSH) version 2.0 to provide secure file transfer capabilities. The IETF Internet Draft states that, even though this protocol is described in the context of the SSH-2 protocol, it could be used in a number of different applications, such as secure file transfer over Transport Layer Security (TLS) and transfer of management information in VPN applications.

This protocol assumes that it is run over a secure channel, such as SSH, that the server has already authenticated the client, and that the identity of the client user is available to the protocol.

If it is possible to connect to a remote system using SSH, then it will be possible to use SFTP to manage files. Test SSH access with the following command:

```
pbmac@pbmac-server $ ssh pbmac@pbmac-server.com
```

You should use a valid userid and a valid hostname, NOT the credentials in the above example. If that works, exit back out by typing:

```
exit
```

Now use the **sftp** command to access the remote system and transfer files:

```
pbmac@pbmac-server $ sftp pbmac@pbmac-server.com:pdfFiles/Resume.pdf
```

When you run this command, **sftp** will connect to pbmac-server.com, ask you for your password, and once you're authenticated it will attempt to download the file pdfFiles/Resume.pdf. Since we didn't put a slash at the beginning of the directory name, it will look for pdfFiles in the user's home directory on the server. If it finds Resume.pdf, it will download it.

The output will look like this:

```
Fetching /home/pbmac/pdfFiles/Resume.pdf to Resume.pdf
```

...and then **sftp** will exit. You can also specify a location for the file to be downloaded.

It is possible to save the downloaded file to a different location. In the following example it downloads the Resume.pdf file to the user's tmp folder

```
pbmac@pbmac-server $ sftp pbmac@pbmac-server.com:pdfFiles/Resume.pdf tmp/Resume.pdf
```

...and the output will indicate the new filename:

```
Fetching /home/pbmac/pdfFiles/Resume.pdf to tmp/Resume.pdf
```

The **rsync** Command

The **rsync** utility efficiently transfers and synchronizes files between a computer and an external hard drive and across networked computers by comparing the modification times and sizes of files. It is commonly found on Unix-like operating systems. **rsync** is written in C as a single threaded application. The **rsync** algorithm is a type of delta encoding, and is used for minimizing network usage. Zlib may be used for additional data compression, and SSH or stunnel can be used for security. **rsync** is the facility typically used for synchronizing software repositories on mirror sites used by package management systems.

rsync is typically used for synchronizing files and directories between two different systems. For example, if the command rsync local-file user@remote-host:remote-file is run, **rsync** will use SSH to connect as user to remote-host. Once connected, it will invoke the remote host's **rsync** and then the two programs will determine what parts of the local file need to be transferred so that the remote file matches the local one.

Adapted from:
"scp command in Linux with Examples" by rahulkumarmandal, Geeks for Geeks is licensed under CC BY-SA 4.0
"SSH File Transfer Protocol" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"rsync" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0

12-F.20: mirrorvg / rsync / scp is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-F.21: File Compression

Data Compression

Data compression is the process of encoding information using fewer bits than the original representation. Any particular compression is either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. Lossy compression reduces bits by removing unnecessary or less important information. Typically, a device that performs data compression is referred to as an encoder, and one that performs the reversal of the process (decompression) as a decoder.

The process of reducing the size of a data file is often referred to as data compression. In the context of data transmission, it is called source coding; encoding done at the source of the data before it is stored or transmitted. Source coding should not be confused with channel coding, for error detection and correction or line coding, the means for mapping data onto a signal.

The `gzip` Command

The **`gzip`** command compresses files. Each single file is compressed into a single file. The compressed file consists of a GNU zip header and deflated data.
If given a file as an argument, **`gzip`** compresses the file, adds a ".gz" suffix, and deletes the original file. With no arguments, **`gzip`** compresses the standard input and writes the compressed file to standard output.

Difference between `gzip` and `zip` commands in Unix and when to use which command

- ZIP and GZIP are two very popular methods of compressing files, in order to save space or to reduce the amount of time needed to transmit the files across the network, or internet.
- In general, GZIP is much better compared to ZIP; it is better in terms of compression, especially when compressing a huge number of files.
- The common practice with GZIP is to archive all the files into a single tarball before compression. In ZIP files, the individual files are compressed and then added to the archive.
- When you want to pull a single file from a ZIP, it is simply extracted, then decompressed. With GZIP, the whole file needs to be decompressed before you can extract the file you want from the archive.
- When pulling a 1MB file from a 10GB archive, it is quite clear that it would take a lot longer in GZIP, than in ZIP.
- GZIP's disadvantage in how it operates is also responsible for GZIP's advantage. Since the compression algorithm in GZIP compresses one large file instead of multiple smaller ones, it can take advantage of the redundancy in the files to reduce the file size even further.
- If you archive and compress 10 identical files with ZIP and GZIP, the ZIP file would be over 10 times bigger than the resulting GZIP file.

Syntax:

```
gzip [ OPTIONS ] [filenames]
```

Command Options:

Options	Option Meaning
`-d`	Reverse file compression (decompression).
`-r`	Enable directory recursion during compression or decompression.
`-t`	Perform an integrity check on the compressed file.
`-v`	Display the name and percentage reduction of the compressed or decompressed file.

In the following example, the .txt file is compressed; notice that the original file is no longer present, but copy the compressed file.

```
pbmac@pbmac-server $ ls myFile.txt
myFile.txt
pbmac@pbmac-server $ gzip myFile.txt
pbmac@pbmac-server $ ls myFile.txt*
myFile.txt.gz
```

The xz Utilities

XZ Utils is a set of free software command-line lossless data compressors, including **lzma** and **xz**, for Linux operating systems.

xz achieves higher compression rates than alternatives like **gzip** and **bzip2**. Decompression speed is faster than **bzip2**, but slower than **gzip**. Compression can be much slower than **gzip**, and is slower than **bzip2** for high levels of compression, and is most useful when a compressed file will be used many times.

XZ Utils consists of two major components:

- **xz**, the command-line compressor and decompressor (analogous to **gzip**)
- **liblzma**, a software library with an API similar to zlib

Various command shortcuts exist, such as lzma (for **xz** --format=lzma), unxz (for **xz** --decompress; analogous to gunzip) and xzcat (for unxz --stdout; analogous to zcat).

XZ Utils can compress and decompress both the **xz** and **lzma** file formats, but since the **lzma** format is now legacy, XZ Utils compresses by default to **xz**.

The bzip2 Command

The **bzip2** command is used to compress and decompress the files i.e. it helps in binding the files into a single file which takes less storage space than the original file used to take. It has a slower decompression time and higher memory use. It uses Burrows-Wheeler block sorting text compression algorithm, and Huffman Coding. Each file is replaced by a compressed version of itself, with the original name of the file followed by extension bz2.

bzip2 is actually part of a larger suite of commands all based around this compression algorithm. The other commands that are a part of this suite are listed in the table below:

Command	Description
bzip2	Compress a file.
bunzip2	Decompress a file.
bzcat	Decompress a file to standard output.
bzdiff	Run the diff command on compressed files.
bzip2recover	Recover data from damaged .bz2 files.
bzless	Run the less command on compressed files.
bzmore	Run the more command on compressed files.

The zip Command

The **zip** command is a compression and file packaging utility for Unix. Each file is stored in single .zip {.zip-filename} file with the extension .zip.

- **zip** is used to compress the files to reduce file size and is also used as file package utility. **zip** is available in many operating systems like Unix, Linux, Windows, etc.
- If you have a limited bandwidth between two servers and want to transfer the files faster, then zip the files and transfer.
- The **zip** program puts one or more compressed files into a single zip archive, along with information about the files (name, path, date, time of last modification, protection, and check information to verify file integrity). An entire directory structure can be packed into a zip archive with a single command.
- Compression ratios of 2:1 to 3:1 are common for text files. **zip** has one compression method (deflation) and can also store files without compression. **zip** automatically chooses the better of the two for each file to be compressed.
  The program is useful for packaging a set of files for distribution, for archiving files, and for saving disk space by temporarily compressing unused files or directories.

Syntax:

```
zip [ OPTIONS ] zipfile files_list
```

Command Options:

Options	Option Meaning
-d, --delete	Remove (delete) entries from a zip archive.
-e, --encrypt	Encrypt the contents of the zip archive using a password that is entered on the terminal.
-F, --fix, -FF, --fixfix	Fix the zip archive. The -F option can be used if some portions of the archive are missing, but requires a reasonably intact central directory.
-r, --recurse-paths	Travel the directory structure recursively.

To create a zip file you specify the zip file name, followed by the files to include in the zip file:

```
pbmac@pbmac-server $ zip myfile.zip filename.txt otherfile.txt picture.jpg
```

Extracting files from zip file

Unzip will list, test, or extract files from a zip archive, commonly found on Unix systems. The default behavior (with no options) is to extract into the current directory (and sub-directories below it) all files from the specified zip archive.

```
pbmac@pbmac-server $ unzip myfile.zip
```

Adapted from:
"Data compression" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"Gzip Command in Linux" by Shubrodeep Banerjee, Geeks for Geeks is licensed under CC BY-SA 4.0
"XZ Utils" by Multiuple Contributors, Wikipedia is licensed under CC BY-SA 3.0
"bzip2 command in Linux with Examples" by sarthak_ishu11, Geeks for Geeks is licensed under CC BY-SA 4.0
"ZIP command in Linux with examples" by Mohak Agrawal, Geeks for Geeks is licensed under CC BY-SA 4.0

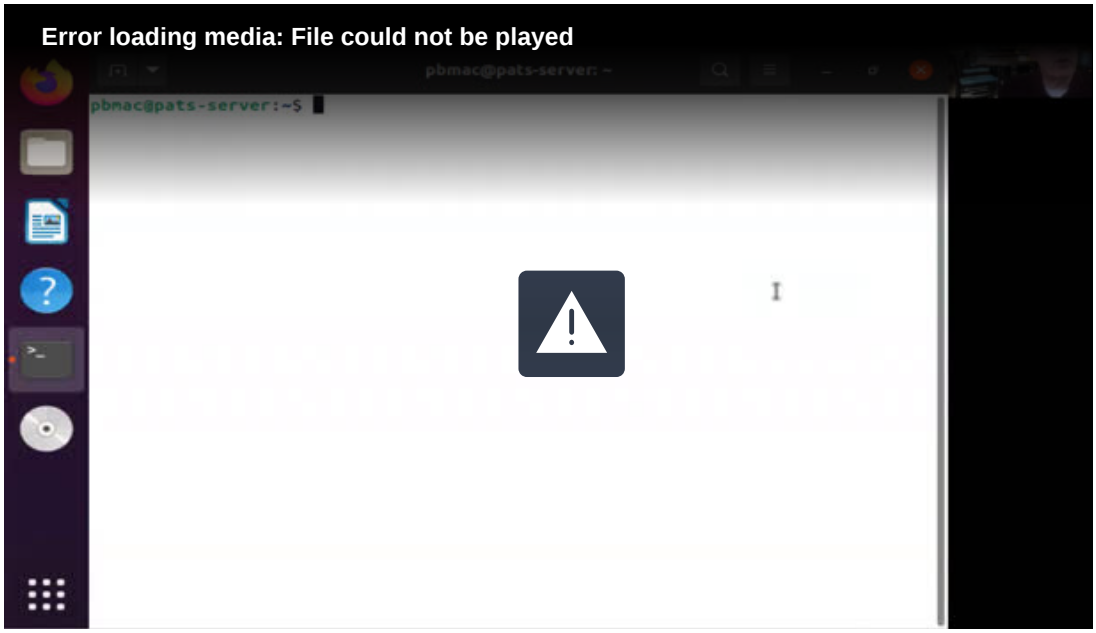
12-F.21: File Compression is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

12-F.22: Integrity Check and Checksums

Integrity Checking

File integrity monitoring (FIM) is an internal control or process that performs the act of validating the integrity of an operating system and application software files using a verification method between the current file state and a known, good baseline. This comparison method often involves calculating a known cryptographic checksum of the file's original baseline and comparing with the calculated checksum of the current state of the file. Other file attributes can also be used to monitor integrity.

How FIM is incorporated in Linux is discussed in this video.



Generally, the act of performing file integrity monitoring is automated using internal controls such as an application or process. Such monitoring can be performed randomly, at a defined polling interval, or in real-time.

Changes to configurations, files and file attributes across the IT infrastructure are common, but hidden within a large volume of daily changes can be the few that impact file or configuration integrity. These changes can also reduce security posture and in some cases may be leading indicators of a breach in progress. Values monitored for unexpected changes to files or configuration items include:

- Credentials
- Privileges and security settings
- Content
- Core attributes and size
- Hash values
- Configuration values

Hash Function

A cryptographic hash function (CHF) is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit array of a fixed size (the "hash value," "hash," or "message digest"). It is a one-way function, that is, a function which is practically infeasible to invert. Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Cryptographic hash functions are a basic tool of modern cryptography.

The ideal cryptographic hash function has the following main properties:

- It is deterministic, meaning that the same message always results in the same hash.
- It is quick to compute the hash value for any given message.
- It is infeasible to generate a message that yields a given hash value (i.e., to reverse the process that generated the given hash value).
- It is infeasible to find two different messages with the same hash value.
- A small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value (avalanche effect).

Cryptographic hash functions have many information-security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information-security contexts, cryptographic hash values are sometimes called (digital) fingerprints, checksums, or just hash values, even though all these terms stand for more general functions with rather different properties and purposes.

The MD5 Checksum

The md5sum is designed to verify data integrity using MD5 (Message Digest Algorithm 5). MD5 is 128-bit cryptographic hash and if used properly it can be used to verify file authenticity and integrity.

Syntax:

```
md5sum [ OPTION ] [FILE]...
```

Using the md5sum command will calculate the checksum on the specified file.

```
pbmac@pbmac-server $  md5sum /home/mandeep/test/test.cpp
c6779ec2960296ed9a04f08d67f64422  /home/pbmac/test/test.cpp
```

The `sha256sum` Command

The program **`sha256sum`** is designed to verify data integrity using the SHA-256 (SHA-2 family with a digest length of 256 bits). SHA-256 hashes used properly can confirm both file integrity and authenticity. SHA-256 serves a similar purpose to a prior algorithm recommended by Ubuntu, MD5, but is less vulnerable to attack.

Comparing hashes makes it possible to detect changes in files that would cause errors. The possibility of changes (errors) is proportional to the size of the file; the possibility of errors increases as the file becomes larger. It is a very good idea to run a SHA-256 hash comparison check when you have a file like an operating system install CD that has to be 100% correct.

There are several iterations of the sha command: sha1sum; sha256sum; sha384sum; and sha512sum/. They are basically identical but use a different number of bits to compute the checksum.

Adapted from:
"md5sum Command in Linux with Examples" by msdeep14, Geeks for Geeks is licensed under CC BY-SA 4.0
"HowToSHA256SUM" by Anthony Geoghegan, Ubuntu Community Wiki is licensed under CC BY-SA 3.0

12-F.22: Integrity Check and Checksums is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 13: Working with Bash Scripts

13-A.1: How to Customize the Bash Shell

13-A.2: env / alias / PATH / time

13-B.3: Fundamentals of Linux Scripts

13-B.4: Shell Operators

13-B.5: Strings / Arrays / Functions

13-C.6: Writing and Running Linux Scripts

13-C.7: Shell Expansion / Variable Substitution / Command Substitution

13-C.8: Positional Parameters / exec Command / source Command

13-C.9: Script Permissions / Script Extensions

13-D.10: Shell Control Statements

13-D.11: Script Loops

Module 13: Working with Bash Scripts is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-A.1: How to Customize the Bash Shell

EXAM OBJECTIVES COVERED

1.6 Given a scenario, configure localization options.
4.4 Given a scenario, analyze and troubleshoot application and hardware issues.
5.1 Given a scenario, deploy and execute basic Bash scripts.

Objectives of this Module

In this lesson, you will:

- Learn how to customize the Bash shell environment for script execution.
- Determine concepts fundamental to both scripting and programming.
- Create a simple Bash script and successfully execute it.
- Create more complicated Bash scripts that include flow control like conditional statements and loops.

Linux Shell

The most generic sense of the term "Linux shell" means any program that users employ to type commands. A shell hides the details of the underlying operating system and manages the technical details of the operating system kernel interface, which is the lowest-level, or "inner-most" component of most operating systems.

In the Linux operating systems, users typically have many choices of command-line interpreters for interactive sessions. When a user logs into the system interactively, a shell program is automatically executed for the duration of the session. The type of shell, which may be customized for each user, is typically stored in the user's profile, for example in the local passwd file or in a distributed configuration system such as NIS or LDAP; however, the user may execute any other available shell interactively.

The environment for any simple command or function may be augmented temporarily by prefixing it with parameter assignments, as described in Shell Parameters. These assignment statements affect only the environment seen by that command.

In Linux, processes spawn other processes, and each process has its own process identification number - the process ID. In the following example we initially see the bash process with a process ID (PID) of 14274. Notice that the other process has a parent PID (PPID) that is the PID of the bash process. So, the bash process spawned the ps -f process in this parent-child analogy.

Then a new bash process is started from the command line, and we see that its PPID is the original process ID. Now the ps -f command is run again, BUT, the PPID is the PID of the new bash process.

```
pbmac@pbmac-server $ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
pbmac      591 14274  0 14:59 pts/0    00:00:00 ps -f
pbmac    14274 14264  0 Nov16 pts/0    00:00:00 bash

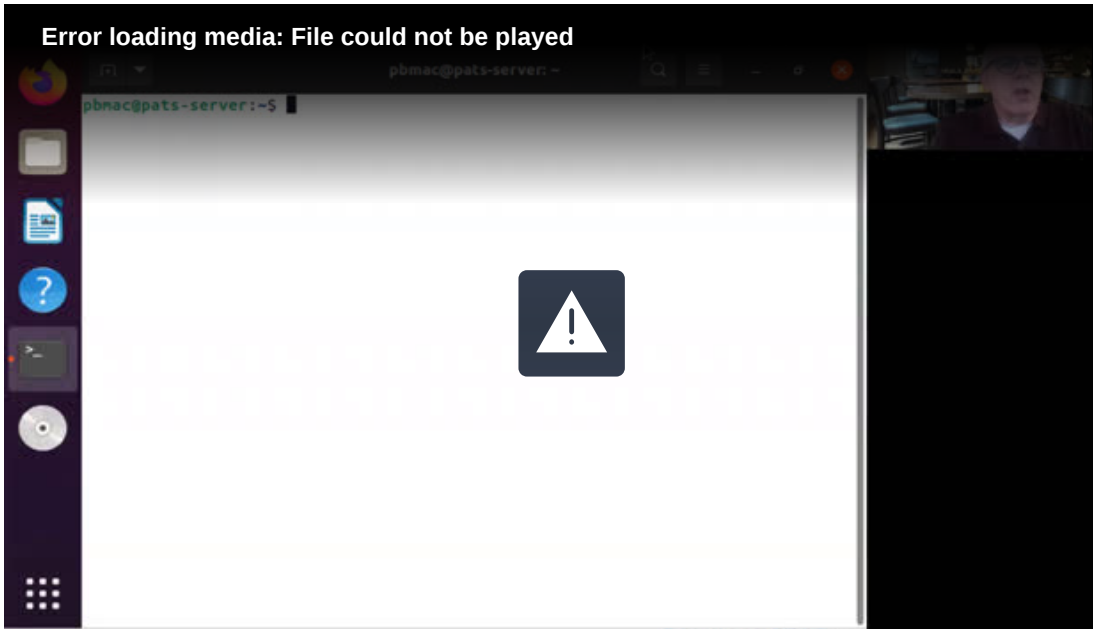
pbmac@pbmac-server $ bash

pbmac@pbmac-server $ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
pbmac      599 14274  0 14:59 pts/0    00:00:00 bash
pbmac      673   599  0 14:59 pts/0    00:00:00 ps -f
pbmac    14274 14264  0 Nov16 pts/0    00:00:00 bash
```

Linux Shells Scripts

A shell script is a computer program designed to be run by the Linux shell, a command-line interpreter. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text. A script which sets up the environment, runs the program, and does any necessary cleanup, logging, etc. is called a wrapper.

There is a great deal that creates the shell environment in Linux, and it is important to understand all of these pieces to properly use the shell to its fullest extent.



Shell Variables

What are Variables For?

Whether you need variables in Bash depends on what you do in a terminal. For some users, variables are an essential means of managing data, while for others they're minor and temporary conveniences, and for still others, they may as well not exist.

Ultimately, variables are a tool. You can use them when you find a use for them, or leave them alone in the comfort of knowing they're managed by your OS. Knowledge is power, though, and understanding how variables work in Bash can lead you to all kinds of unexpected creative problem-solving.

How to Set a Variable

You don't need special permissions to create a variable. They're free to create, free to use, and generally harmless. In a Bash shell (on Linux and Mac), you can set them by defining a variable name, and then setting its value.

Setting variables can be a common thing for people who use the shell often, so the process has become somewhat informal. When a string is followed by an equal sign (=) and a value, Bash quietly assumes that you're setting a variable. The following example creates a new variable called **FOO** and sets the value to the string **/home/pbmac/Documents**:

```
pbmac@pbmac-server $ FOO="/home/pbmac/Documents
```

Success is eerily silent, so you may not feel confident that your variable got set. You can see the results for yourself with the **echo** command, recalling your variable by prepending it with a dollar sign (**$**). To ensure that the variable is read exactly as you defined it, you can also wrap it in braces and quotes. Doing this preserves any special characters that might appear in the variable; in this example, that doesn't apply, but it's still a good habit to form:

```
pbmac@pbmac-server $ echo "${FOO}"
/home/pbmac/Documents
```

Setting variables can be a common thing for people who use the shell often, so the process has become somewhat informal. When a string is followed by an equal sign (=) and a value, Bash quietly assumes that you're setting a variable, making the **declare** keyword unnecessary:

```
pbmac@pbmac-server $ FOO="/home/pbmac/Documents"
```

Variables usually are meant to convey information from one system to another. In this simple example, your variable is not very useful, but it can still communicate information. For instance, because the content of the **FOO** variable is a file path, you can use the variable as a shortcut to the ~/Documents directory:

```
pbmac@pbmac-server $ pwd
/home/pbmac
pbmac@pbmac-server $cd $FOO
pbmac@pbmac-server $ pwd
/home/pbmac/Documents
```

Variables can be any non-reserved string (along with integers and underscores). They don't have to be capitalized, but they often are so that they're easy to identify as variables.

How to Clear a Variable

You can clear a variable with the **unset** command:

```
pbmac@pbmac-server $ unset FOO
pbmac@pbmac-server $ echo $FOO
```

Environment Variables

Environment variables contain information about your login session, stored for the system shell to use when executing commands. They exist whether you're using Linux, Mac, or Windows. Many of these variables are set by default during installation or user creation.

Environment variables are no different, technically, than variables. They can be set, recalled, and cleared with exactly the same syntax used for variables.

You don't often use environment variables directly. They're referenced by individual applications and daemons as needed. For instance, your home directory is set as an environment variable when you log in. For example, on Linux you can see your **HOME** environment variable's contents like this:

```
$ echo $HOME
HOME=/home/seth
```

Environment variables can be useful when you want to override default settings, or when you need to manage new settings that your system has no reason to create on its own. For instance, when you type a command, the only reason your computer knows how to find the application corresponding to that command is that the **PATH** environment variable tells it where to look. This variable lists valid directories for your operating system to search for commands, whether that command is **ls** or **cp**, or a graphical application like Firefox or Chrome, or anything else.

Different environment variables get used by different systems. Your PATH variable is vital to your terminal emulator, for instance, but a lot less significant to, say, Java (which has its own paths, which point to important Java libraries). However, the USER variable is used by several systems as a way to identify who is requesting a service. For example, if you're on a multi-user system and need to check your local mailbox, your mail command knows which mail spool to retrieve based on the MAIL and USER variables.

Setting an Environment Variable

Usually, the installer program, whether it's dnf on Fedora, apt on Ubuntu, brew on Mac, or a custom installer, updates your environment variables for a new application. Sometimes, though, when you're installing something outside of your distribution's intended toolset, you may have to manage an environment variable yourself. Or you might choose to add an environment variable to suit your preferences. If you decide you want to keep some applications in a bin folder located in your home directory, then you must add that directory to your PATH so your operating system knows to look there for applications to run when you issue a command.

Temporary Environment Variables

You can add a location to your path the way you create throw-away variables. It works, but only as long as the shell you used to modify your system path remains open. For instance, open a terminal window and modify your system path:

```
pbmac@pbmac-server $ export PATH=$PATH:/home/pbmac/bin
```

Confirm the changes - the /home/pbmac/bin has been added at the end of the list:

```
pbmac@pbmac-server $ echo $PATH
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/snap/bin:/home/pbmac/bin:
```

```
/home/pbmac/.local/bin:/snap/bin:/home/pbmac/bin
```

Close the terminal window:

```
pbmac@pbmac-server $ exit
```

Open a new terminal window and take a look at the PATH variable:

```
pbmac@pbmac-server $ echo $PATH
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/snap/bin:/home/seth/bin:/home/seth/.local
```

This variable has reverted to its default state because **PATH** isn't getting set with each new shell. For that, you must configure your variables to load any time a shell is launched.

Permanent Environment Variables

You can set your own persistent environment variables in your shell configuration file, the most common of which is ~/.bashrc. If you're a system administrator managing several users, you can also set environment variables in a script placed in the /etc/profile.d directory.

The syntax for setting a variable by configuration file is the same as setting a variable in your shell:

```
export PATH=$PATH:/snap/bin:/home/pbmac/bin
```

Close the current shell, or else force it to load the updated config. To reload the file, simply enter "`. ~/.bashrc`" - a period, followed by a space, and ~/.bashrc

```
$ . ~/.bashrc
```

Finally, take another look at your system path:

```
pbmac@pbmac-server $ echo $PATH
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/snap/bin:/home/pbmac/bin:/home/pbmac/.loc
/home/pbmac/bin
```

It is now set correctly to include your additional custom directory.

The `export` Command

The **`export`** command is a bash shell BUILTINS commands, which means it is part of the shell. It marks environment variables to be exported to child-processes.

`Export` is defined in POSIX as the shell shall give the export attribute to the variables corresponding to the specified names, which shall cause them to be in the environment of subsequently executed commands. If the name of a variable is followed by = word, then the value of that variable shall be set to the word.

In simple terms, environment variables are set when you open a new shell session. If you change any of the variable values, the shell has no way of picking that change. The export command, on the other hand, provides the ability to update the current shell session about the change you made to the exported variable. You don't have to wait until a new shell session to use the value of the variable you changed.

Syntax :

```
export [-f] [-n] [name[=value] ...] or export -p
```

To assign value before exporting use the following syntax:

```
pbmac@pbmac-server $ export name[=value]
```

An example of using the **`export`** command to set vim as a text editor:

```
pbmac@pbmac-server $ export EDITOR=/usr/bin/vim
pbmac@pbmac-server $ export | grep EDITOR
```

```
declare -x EDITOR="/usr/bin/vi"
```

Adapted from:

"Unix shell" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 4.0

"Using variables in Bash" by Seth Kenlon, OpenSource.com is licensed under CC BY-SA 4.0

"What are environment variables in Bash?" by Seth Kenlon, OpenSource.com is licensed under CC BY-SA 4.0

"export command in Linux with Examples" by Prasoon_Mishra, Geeks for geeks is licensed under CC BY-SA 4.0

13-A.1: How to Customize the Bash Shell is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-A.2: env / alias / PATH / time

The **env** Command

The **env** command is used to print environment variables. It is also used to run a utility or command in a custom environment. In practice, **env** has another common use. It is often used by shell scripts to launch the correct interpreter. In this usage, the environment is typically not changed.

Environmental Variable	Meaning	View Variable by Entering:
HOSTNAME	Name of this computer.	`echo $HOSTNAME`
HISTSIZE	Number of commands kept in the history defaults to 500.	`echo $HISTSIZE`
HOME	Home directory of the current user.	`echo $HOME`
LANG	Used to determine the locale category for any category not specifically selected with a variable starting with LC_.	`echo $LANG`
PATH	Search path for commands. It is a colon-separated list of directories where the shell looks for commands being entered.	`echo $PATH`
PS1	Current prompt settings.	`echo $PS1`
TERM	Login terminal type.	`echo $TERM` `export TERM=vt100`
SHELL	Path to login shell.	`echo $SHELL`
EDITOR	Name of default text editor for this user.	`export EDITOR=/usr/bin/vim`

You can view all environment variables set on your system with the **env** command. The list is long, so pipe the output through **more** to make it easier to read:

```
$ env | less
TERM=xterm-256color
LANG=en_US.UTF-8
DISPLAY=:0
GNOME_SHELL_SESSION_MODE=ubuntu
EDITOR=/usr/bin/vi
GTK2_MODULES=overlay-scrollbar
COLORTERM=truecolor
USERNAME=pbmac
JAVA_HOME=/usr/lib/jvm/java-11-oracle
J2SDKDIR=/usr/lib/jvm/java-11-oracle
XDG_VTNR=2
S_COLORS=auto
XDG_SESSION_ID=2
USER=pbmac
[...]
```

It can appear that there are several ways to set and view variables, and environmental variables in Linux. The table below is to help sort out the differences.

Command	What it does
declare	Declare variables and give them attributes. If no names are given, then display the values of variables instead.
set	This builtin is so complicated that it deserves its own section. set allows you to change the values of shell options and set the positional parameters, or to display the names and values of shell variables.
env	Without any options, shows current environment variables with their values; However can be used to set environment variable for a single command with the -i flag.
export	Makes shell variables environment variables.

Linux PATH Variable

When you type a command into your Linux shell, it doesn't look in every directory to see if there's a program by that name. It only looks to the ones you specify. How does it know to look in the directories mentioned above? It's simple: They are a part of an environment variable, called $PATH, which your shell checks in order to know where to look.

Sometimes, you may wish to install programs into other locations on your computer, but be able to execute them easily without specifying their exact location. You can do this easily by adding a directory to your $PATH. To see what's in your $PATH right now, type this into a terminal:

```
pbmac@pbmac-server $ echo $PATH
```

You'll probably see the directories mentioned above, as well as perhaps some others, and they are all separated by colons. Now let's add another directory to the list.

Set your PATH

Let's say you wrote a little shell script called hello.sh and have it located in a directory called /place/with/the/file. This script provides some useful function to all of the files in your current directory that you'd like to be able to execute no matter what directory you're in.

Simply add /place/with/the/file to the $PATH variable with the following command:

```
export PATH=$PATH:/place/with/the/file
```

The script anywhere on the system should now be found by just typing in its name, without having to include the full path as you type it.

Set the PATH Permanently

When the computer is restarted or a new terminal instance gets created the addition to the path is gone! This is by design. The variable $PATH is set by the shell every time it launches, but it can be set so that it always includes the new path with every new shell that gets opened. The exact way to do this depends on which shell is being run.

Not sure which shell is running? If you're using pretty much any common Linux distribution, and haven't changed the defaults, chances are you're running Bash. But you can confirm this with a simple command:

```
pbmac@pbmac-server $ echo $0
```

That's the "echo" command followed by a dollar sign ($) and a zero. $0 represents the zeroth segment of a command (in the command echo $0, the word "echo" therefore maps to $1), or in other words, the thing running your command. Usually this is the Bash shell, although there are others, including Dash, Zsh, Tcsh, Ksh, and Fish.

For Bash, simply add the line from above - export PATH=$PATH:/place/with/the/file - to the appropriate file that will be read when your shell launches. There are a few different places where you could conceivably set the variable name: potentially in a file called ~/.bash_profile, ~/.bashrc, or ~/.profile. The difference between these files is (primarily) when they get read by the shell. If you're not sure where to put it, ~/.bashrc is a good choice.

For other shells, you'll want to find the appropriate place to set a configuration at start time; ksh configuration is typically found in ~/.kshrc, zsh uses ~/.zshrc. Check your shell's documentation to find what file it uses.

The alias Command

A Bash alias is a method of supplementing or overriding Bash commands with new ones. Bash aliases make it easy for users to customize their experience in a POSIX terminal. They are often defined in $HOME/.bashrc or $HOME/bash_aliases (which must be loaded by $HOME/.bashrc).

Most distributions add at least some popular aliases in the default .bashrc file of any new user account. These are simple ones to demonstrate the syntax of a Bash alias:

```
alias ls='ls -F'
alias ll='ls -lh'
```

Not all distributions ship with pre-populated aliases, though. If you add aliases manually, then you must load them into your current Bash session:

```
pbmac@pbmac-server $ source ~/.bashrc
```

Otherwise, you can close your terminal and re-open it so that it reloads its configuration file. To find if any aliases are currently configured, simply use the alias command.

```
pbmac@pbmac-server $ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo erro
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ls='ls --color=auto'
```

So, if you get tired of typing 'ls -al' - you can use alias to create the ll command

```
pbmac@pbmac-server $ alias ll='ls -al'
pbmac@pbmac-server $ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo erro
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
```

```
alias ll='ls -al'
alias ls='ls --color=auto'
```

The `time` Command

The **`time`** command in Linux is used to execute a command and prints a summary of real-time, user CPU time and system CPU time spent by executing a command when it terminates. 'Real' time is the time elapsed wall clock time taken by a command to get executed, while 'user' and 'sys' time are the number of CPU seconds that command uses in user and kernel mode respectively.

Syntax:

```
time [ OPTION ] [COMMAND]
```

To see how much time it takes to run the **`ls -l`** command on a specific directory we can use:

```
pbmac@pbmac-server $ time ls -l > /dev/null

real    0m0.004s
user    0m0.000s
sys     0m0.004s
```

Meaning the real clock time was 0.004 seconds, 0.0 seconds running in user space, and 0.004s running in system space.

Adapted from:
"env command in Linux with Examples" by Prasoon_Mishra, Geeks for Geeks is licensed under CC BY-SA 4.0
"env" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 4.0
"How to set your $PATH variable in Linux" by Jason Baker, OpenSource.com is licensed under CC BY-SA 4.0
"time command in Linux with examples" by DrRoot_, Geeks for Geeks is licensed under CC BY-SA 4.0

13-A.2: env / alias / PATH / time is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-B.3: Fundamentals of Linux Scripts

EXAM OBJECTIVES COVERED

5.1 Given a scenario, deploy and execute basic Bash scripts.

Shells

There are several shells available for Linux systems. Each shell does the same job but understands different commands and provides different built-in functions.

- BASH (Bourne Again SHell) – The most widely used shell in Linux systems. It is used as default login shell in Linux systems and in macOS. It can also be installed on Windows OS.
- csh (C SHell) – The C shell's syntax and usage are very similar to the C programming language.
- ksh (Korn SHell) – The Korn Shell also was the base for the POSIX Shell standard specifications, etc.
- tcsh – An enhanced version of csh, the C shell.
- zsh – A powerful interactive shell.
- scsh – An open-source Unix shell embedded within Scheme programming language.

In this discussion we will be using the BASH shell for all examples since it is the default shell of most Linux distributions.

Shell Scripting

Usually shells are interactive, which means the shell accepts commands as input from users and executes them. However, there are times it is necessary to execute a group of several commands routinely, so we must type each command into the terminal.

A shell can accept commands as input from a file, and these commands can be written into a text file which can then be executed to avoid the repetition of having to type each command numerous times. These files are called shell scripts or shell programs, or simply scripts. By tradition, each shell script is saved with **.sh** file extension e.g., **myscript.sh**

A shell script has syntax just like any other programming language. If you have any prior experience with any programming languages like Python, C/C+,+ etc. it would be very easy to get started with shell scripts.

A shell script is normally comprised of the following elements:

- Shell Keywords – if, else, break, etc.
- Shell commands – cd, ls, echo, pwd, touch, etc.
- Functions
- Control flow – if..then..else, case and shell loops, etc.

Why Shell Scripts Are Needed

There are many reasons to write shell scripts:

- To avoid repetitive work and automate tasks
- System admins use shell scripting to create routine sets of commands
- System monitoring
- Adding new functionality to the shell, etc.

Advantages of Shell Scripts

- The command and syntax are exactly the same as when entered directly on the command line, so programmers use the familiar command syntax.
- Writing shell scripts is quicker than writing in a typical programming language (C++, Java).
- Quick start
- Interactive debugging, etc.

Disadvantages of Shell Scripts

- Prone to costly errors, a single mistake can change the command which might be harmful.
- Slow execution speed.

- Design flaws within the language syntax or implementation.
- Not well suited for large and complex tasks.
- Provide minimal data structure unlike other scripting languages, etc.

Variable Substitution

The *name* of a variable is a placeholder for its *value*, the data it holds. Referencing (retrieving) its value is called *variable substitution*.

```
pbmac@pbmac-server $ variable1=23

pbmac@pbmac-server $ echo variable1
variable1

pbmac@pbmac-server $ echo $variable1
23
```

Let us carefully distinguish between the name of a variable and its value. If variable1 is the name of a variable, then $variable1 is a reference to its value, the data item it contains.

The only times a variable appears "naked" -- without the $ prefix -- is when declared or assigned, when unset, when exported, in an arithmetic expression within double parentheses ((...)), or in the special case of a variable representing a signal (see Example 32-5). Assignment may be with an = (as in var1=27), in a read statement, and at the head of a loop (for var2 in 1 2 3).

Enclosing a referenced value in double quotes (" ... ") does not interfere with variable substitution. This is called partial quoting, sometimes referred to as "weak quoting." Using single quotes (' ... ') causes the variable name to be used literally, and no substitution will take place. This is full quoting, sometimes referred to as 'strong quoting.'

Note that $variable is actually a simplified form of ${variable}. In contexts where the $variable syntax causes an error, the longer form may work (see Section 10.2, below).

Example 4-1. Variable assignment and substitution

```
#!/bin/bash
# ex9.sh

# Variables: assignment and substitution

a=375
hello=$a
#   ^ ^  - NOTICE there are NO spaces on either side of the = sign....
```

Variable Assignment

The assignment operator (no space before and after):



Do not confuse this with = and -eq, which test, rather than assign! Note that = can be either an *assignment* or a *test* operator, depending on context.

Example 4-2. Plain Variable Assignment

```
#!/bin/bash
# Naked variables
echo

# When is a variable "naked", i.e., lacking the '$' in front?
# When it is being assigned, rather than referenced.
# Assignment
a=879
echo "The value of \"a\" is $a."
# NOTICE - the \ character escapes the following" when we want to use quotations with

# Assignment using 'let'
let a=16+5
echo "The value of \"a\" is now $a."
```

Adapted from:

"Introduction to Linux Shell and Shell Scripting" by Atul Kumar, Geeks for Geeks is licensed under CC BY-SA 4.0

"4.1. Variable Substitution" by Multiple Contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0

"4.2. Variable Assignment" by Multiple Contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0

13-B.3: Fundamentals of Linux Scripts is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-B.4: Shell Operators

Operators in Shell Scripts

There are five basic operators in bash/shell scripting:

- Arithmetic Operators
- Relational Operators
- Boolean Operators
- Bitwise Operators
- File Test Operators

1. **Arithmetic Operators**: These operators are used to perform normal arithmetics/mathematical operations. There are seven arithmetic operators:
   - **Addition (+)**: Binary operation used to add two operands.
   - **Subtraction (-)**: Binary operation used to subtract two operands.
   - **Multiplication (*)**: Binary operation used to multiply two operands.
   - **Division (/)**: Binary operation used to divide two operands.
   - **Modulus (%):** Binary operation used to find remainder of two operands.
   - **Increment Operator (++)**: Unary operator used to increase the value of operand by one.
   - **Decrement Operator (–)**: Unary operator used to decrease the value of a operand by one
2. **Relational Operators**: Relational operators are those operators which define the relation between two operands. They give either true or false depending upon the relation. There are six types:
   - **‘==’ Operator**: Double equal to operator compares the two operands. It returns true if they are equal otherwise returns false.
   - **‘!=’ Operator**: Not equal to operator returns true if the two operands are not equal otherwise it returns false.
   - **‘<' Operator**: Less than operator returns true if first operand is less than second operand otherwise returns false.
   - **‘<=' Operator**: Less than or equal to operator returns true if first operand is less than or equal to second operand otherwise returns false
   - **‘>’ Operator:** Greater than operator returns true if the first operand is greater than the second operand otherwise returns false.
   - **‘>=’ Operator**: Greater than or equal to operator returns true if first operand is greater than or equal to second operand otherwise returns false.
3. **Logical Operators**: They are also known as boolean operators. These are used to perform logical operations. There are three types:
   - **Logical AND (&&)**: This is a binary operator, which returns true if both the operands are true otherwise returns false.
   - **Logical OR (||)**: This is a binary operator, which returns true if either of the operands is true or both the operands are true and returns false if none of them is false.
   - **Not Equal to (!)**: This is a unary operator which returns true if the operand is false and returns false if the operand is true.
4. **Bitwise Operators**: A bitwise operator is an operator used to perform bitwise operations on bit patterns. There are six types.
   - **Bitwise And (&)**: Bitwise & operator performs binary AND operation bit by bit on the operands.
   - **Bitwise OR (|)**: Bitwise | operator performs binary OR operation bit by bit on the operands.
   - **Bitwise XOR (^)**: Bitwise ^ operator performs binary XOR operation bit by bit on the operands.
   - **Bitwise compliment (~)**: Bitwise ~ operator performs binary NOT operation bit by bit on the operand.
   - **Left Shift (<<)**: This operator shifts the bits of the left operand to left by number of times specified by right operand.
   - **Right Shift (>>)**: This operator shifts the bits of the left operand to right by number of times specified by right operand.
5. **File Test Operator**: These operators are used to test a particular property of a file.
   - **-b operator**: This operator checks whether a file is a block special file or not. It returns true if the file is a block special file, otherwise returns false.
   - **-c operator**: This operator checks whether a file is a character special file or not. It returns true if it is a character special file otherwise returns false.

- **-d operator**: This operator checks if the given directory exists or not. If it exists then operators returns true otherwise returns false.
- **-e operator**: This operator checks whether the given file exists or not. If it exists this operator returns true otherwise returns false.
- **-r operator**: This operator checks whether the given file has read access or not. If it has read access then it returns true otherwise returns false.
- **-w operator**: This operator checks whether the given file has write access or not. If it has write then it returns true otherwise returns false.
- **-x operator**: This operator checks whether the given file has execute access or not. If it has execute access then it returns true otherwise returns false.
- **-s operator**: This operator checks the size of the given file. If the size of given file is greater than 0 then it returns true otherwise it returns false.

Some Things About Operations

Variables are assigned values and can be used to refer to those values in CLI programs and scripts. The value of a variable is set using its name but not preceded by a $ sign. The assignment VAR=10 sets the value of the variable VAR to 10. To print the value of the variable, you can use the statement echo $VAR. Start with text (i.e., non-numeric) variables.

Bash variables become part of the shell environment until they are unset.

Check the initial value of a variable that has not been assigned; it should be null. Then assign a value to the variable and print it to verify its value. You can do all of this in a single CLI program: (there are multiple commands on a single line - separated by a ; - this is totally okay to enter commands this way).

```
pbmac@pbmac-server $ echo $MyVar ; MyVar="Hello World" ; echo $MyVar ;

Hello World
pbmac@pbmac-server $
```

Note: The syntax of variable assignment is very strict. There must be no spaces on either side of the equal (=) sign in the assignment statement.

The empty line indicates that the initial value of MyVar is null. Changing and setting the value of a variable is done the same way. This example shows both the original and the new value.

As mentioned, Bash can perform integer arithmetic calculations, which is useful for calculating a reference to the location of an element in an array or doing simple math problems. It is not suitable for scientific computing or anything that requires decimals, such as financial calculations. There are much better tools for those types of calculations.

Here's a simple calculation: (NOTICE that arithmetic calculations are enclosed in DOUBLE parenthesis - also that the $ character is OUTSIDE the parenthesis).

```
pbmac@pbmac-server $ Var1="7" ; Var2="9" ; echo "Result = $((Var1*Var2))"
Result = 63
```

What happens when you perform a math operation that results in a floating-point number?

```
pbmac@pbmac-server $ Var1="7" ; Var2="9" ; echo "Result = $((Var1/Var2))"
Result = 0
pbmac@pbmac-server $ Var1="7" ; Var2="9" ; echo "Result = $((Var2/Var1))"
Result = 1
```

The result is the nearest integer. Notice that the calculation was performed as part of the echo statement. The math is performed before the enclosing echo command due to the Bash order of precedence. For details see the Bash man page and search "precedence."

Here are some simple examples of logical operations. The first instance sets the variable $X to 1, then tests to see if $X is equal to 1. In the second instance, X is set to 0, so the comparison is not true.

```
pbmac@pbmac-server $ X=1 ; if [ $X -eq 1 ] ; then echo "X equals 1" ; else echo "X doe
X equals 1
pbmac@pbmac-server $ X=0 ; if [ $X -eq 1 ] ; then echo "X equals 1" ; else echo "X doe
X does not equal 1
```

Some things to notice in these examples: 1) when using square brackets - [ ] - there MUST be a space after the left bracket and before the right bracket; 2) when this command is entered on the command line, the semi-colon is required; 3) when using an if statement you MUST close it with a fi statement; 4) if this command is in a text file and is broken up into multiple lines, the semi-colon is NOT required; and 5) the fi to end the if statement: (NOTICE the first line, often referred to as the "shebang" line).

```
#!/bin/bash

if [ $X -eq 1 ]
then
   echo "X equals 1"
else
   echo "X does not equal 1"
fi
```

The control operator syntax using some flow control takes this general form when the && and || control operators are used:

```
preceding commands ; command1 && command2 || command3 ; following commands
```

This syntax can be stated like so: "If command1 exits with a return code of 0, then execute command2, otherwise execute command3." Try it:

```
pbmac@pbmac-server $ Dir=~/testdir ; mkdir $Dir && cd $Dir || echo "$Dir was not creat
```

The control operator syntax, like command1 && command2, works because every command sends a return code (RC) to the shell that indicates if it completed successfully or whether there was some type of failure during execution. By convention, an RC of zero (0) indicates success, and any positive number indicates some type of failure.

Again, if we put this command line into a text file we have a few changes to the syntax:

```
#!/bin/bash

Dir=~/testdir
if [ "mkdir $Dir" ] && [ "cd $Dir" ]
then
   echo "$Dir was not created."
fi
```

The script has 1) the shebang line; 2) notice the content within the [ ] is enclosed in quotations...this is required, otherwise an error is generated; 3) and again, the fi to end the if.

Adapted from:

"Basic Operators in Shell Scripting" by DrRoot_, Geeks for Geeks is licensed under CC BY-SA 4.0
"How to program with Bash: Syntax and tools" by David Both, OpenSource.com is licensed under CC BY-SA 4.0
"How to program with Bash: Logical operators and shell expansions" by David Both, OpenSource.com is licensed under CC BY-SA 4.0
"How to program with Bash: Loops" by David Both, OpenSource.com is licensed under CC BY-SA 4.0

13-B.4: Shell Operators is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-B.5: Strings / Arrays / Functions

Linux String Literals

A lot of keys have special meanings in some context or other. Quoting is used to remove the special meaning of characters or words. Quotes can disable special treatment for special characters, they can prevent reserved words from being recognized as such and they can disable parameter expansion.

Escape Characters

Escape characters are used to remove the special meaning from a single character. A non-quoted backslash, \, is used as an escape character in Bash. It preserves the literal value of the next character that follows, with the exception of newline. If a newline character appears immediately after the backslash, it marks the continuation of a line when it is longer than the width of the terminal; the backslash is removed from the input stream and effectively ignored.

```
pbmac@pbmac-server $ date=20021226
pbmac@pbmac-server $ echo $date
20021226
pbmac@pbmac-server $ echo \$date
$date
```

In this example, the variable date is created and set to hold a value. The first echo displays the value of the variable, but for the second, the dollar sign is escaped.

Single Quotes

Single quotes (') are used to preserve the literal value of each character enclosed within the quotes. A single quote may not occur between single quotes, even when preceded by a backslash.

We continue with the previous example:

```
pbmac@pbmac-server $ echo '$date'
$date
```

Double Quotes

Using double quotes the literal value of all characters enclosed is preserved, except for the dollar sign, the backticks (backward single quotes, ``) and the backslash.

The dollar sign and the backticks retain their special meaning within the double quotes.

The backslash retains its meaning only when followed by dollar, backtick, double quote, backslash or newline. Within double quotes, the backslashes are removed from the input stream when followed by one of these characters. Backslashes preceding characters that don't have a special meaning are left unmodified for processing by the shell interpreter.

A double quote may be quoted within double quotes by preceding it with a backslash.

```
pbmac@pbmac-server $ echo "$date"
20021226

Notice the back quote (its the key to the left of the number 1 key) - this runs the co
displayed
pbmac@pbmac-server $ echo "`date`"
Sun Apr 20 11:22:06 CEST 2003
```

```
pbmac@pbmac-server $ echo "I'd say: \"Go for it!\""
I'd say: "Go for it!"

pbmac@pbmac-server $ echo "\"
More input>"

pbmac@pbmac-server $ echo "\\"
\
```

Linux Shell Arrays

An array is a variable containing multiple values. Any variable may be used as an array. There is no maximum limit to the size of an array, nor any requirement that member variables be indexed or assigned contiguously. Arrays are zero-based: the first element is indexed with the number 0.

Indirect declaration is done using the following syntax to declare a variable:

ARRAY[INDEXNR]=value

The INDEXNR is treated as an arithmetic expression that must evaluate to a positive number.

Explicit declaration of an array is done using the declare built-in:

declare -a ARRAYNAME

A declaration with an index number will also be accepted, but the index number will be ignored. Attributes to the array may be specified using the declare and readonly built-ins. Attributes apply to all variables in the array; you can't have mixed arrays.

Array variables may also be created using compound assignments in this format:

ARRAY=(value1 value2 ... valueN)

Each value is then in the form of [indexnumber]=string. The index number is optional. If it is supplied, that index is assigned to it; otherwise the index of the element assigned is the number of the last index that was assigned, plus one. This format is accepted by declare as well. If no index numbers are supplied, indexing starts at zero.

Adding missing or extra members in an array is done using the syntax:

ARRAYNAME[indexnumber]=value

Remember that the read built-in provides the -a option, which allows for reading and assigning values for member variables of an array.

Dereferencing the Variables in an Array

In order to refer to the content of an item in an array, use curly braces. This is necessary, as you can see from the following example, to bypass the shell interpretation of expansion operators. If the index number is @ or *, all members of an array are referenced.

```
Declare an array with 3 elements
pbmac@pbmac-server $ ARRAY=(one two three)

NOTICE - the curly brackets around the array name
pbmac@pbmac-server $ echo ${ARRAY[*]}
one two three

The brackets are not present - this changes how the shell interprets the value
pbmac@pbmac-server $ echo $ARRAY[*]
```

```
one[*]

both the brackets and the index are supplied.
pbmac@pbmac-server $ echo ${ARRAY[2]}
three

add one more value, then print out the entire array
pbmac@pbmac-server $ ARRAY[3]=four
pbmac@pbmac-server $ echo ${ARRAY[*]}
one two three four
```

Linux Shell Functions

As in almost any programming language, you can use functions to group pieces of code in a more logical way or practice the divine art of recursion.

Declaring a function is just a matter of writing function my_func { my_code }.

Calling a function is just like calling another program, you just write its name.

Example - call this script hello.sh:

```
#!/bin/bash
function quit {
     exit
}
function hello {
     echo Hello!
}

# The next line calls the hello function
hello
# The next line calls the quit function
quit
# The next line simply echos foo
echo foo
```

Notice that functions don't need to be declared in any specific order.

Lines 2-4 contain the 'quit' function. Lines 5-7 contain the 'hello' function. If you are not absolutely sure about what this script does, please try it!

When running the script you'll notice that first, the function 'hello' is called, second the 'quit' function, and the program never reaches line 10.

Functions with Parameters Sample

```
#!/bin/bash

function quit {
      exit
```

```
}

function e {
   echo $1
}

e Hello
e World
quit
```

This script is almost identical to the previous one. The main difference is the function 'e.' This function prints the first argument it receives. Arguments within functions are treated in the same manner as arguments given to the script.

Script Comments

Using comments in a script will make the script more understandable. Comments are like inline documentation for the script. Anyone reading the script can easily understand each step if it has comments included. Comments will be ignored when the script executes. A single line can be commented very easily by inserting a hash tag - # - in the line. Anything after the hash tag until the end of the line is a comment. It is also possible to have comments across multiple lines.

Using an earlier script example, we can show single line comments:

```
#!/bin/bash

# Set the Dir variable to some directory
Dir=~/testdir

# Test to see if the mkdir command completes successfully
# IF it succeeds - then does the cd command complete successfully
if [ "mkdir $Dir" ] && [ "cd $Dir" ]
then
   echo "$Dir was not created." # We print a failure message
fi
```

To create a multi-line comment the syntax is a bit odd, but it works just as well:

```
#!/bin/bash

# Set the Dir variable to some directory
Dir=~/testdir

<<MYCOMMENT-1
Test to see if the mkdir command completes successfully
IF it succeeds - then does the cd command complete successfully
MYCOMMENT-1
if [ "mkdir $Dir" ] && [ "cd $Dir" ]
then
   echo "$Dir was not created." # We print a failure message
fi
```

The multi-line comment MUST begin with <<, followed by some sequence of characters.There is NOTHING special about the MYCOMMENT-1 - you can use any sequence of characters as a delimiter, it just has to be the same at the beginning and the end.

Adapted from:
"3.3. Quoting characters" by Multiple Contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0
"10.2. Array variables" by Multiple Contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0

13-B.5: Strings / Arrays / Functions is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-C.6: Writing and Running Linux Scripts

EXAM OBJECTIVES COVERED

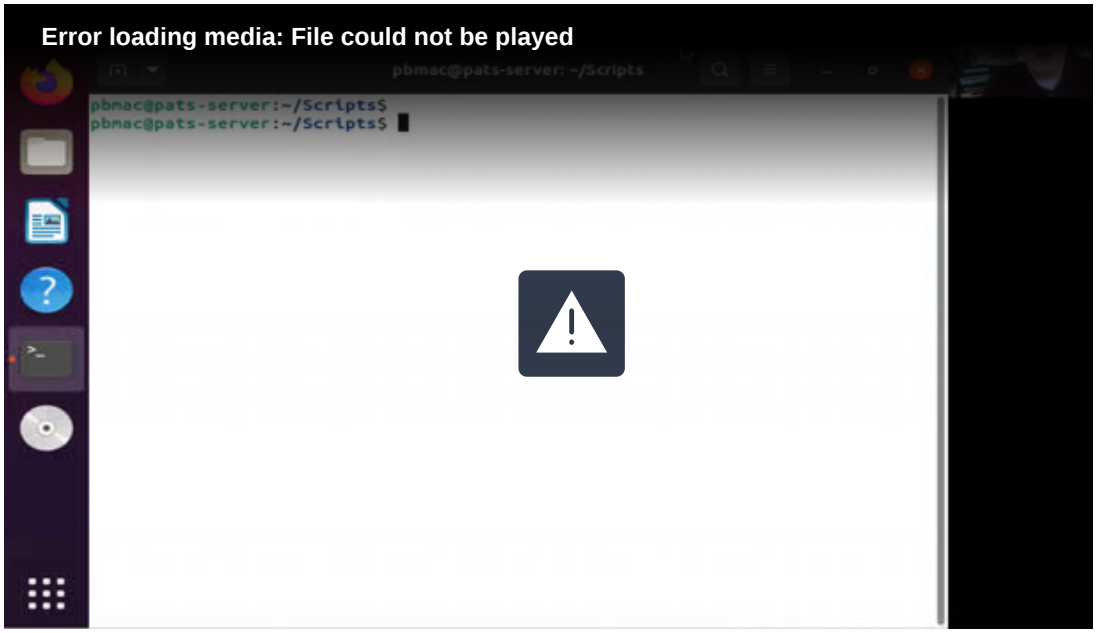
5.1 Given a scenario, deploy and execute basic Bash scripts.

What is shebang?

The shebang is the character sequence consisting of the characters number sign and exclamation mark (#!) at the beginning of a script. It is also called sha-bang, hashbang, pound-bang, or hash-pling.

When a text file with a shebang is used as if it is an executable in Linux, the program loader mechanism parses the rest of the file's initial line as an interpreter directive. The loader executes the specified interpreter program, passing to it as an argument the path that was initially used when attempting to run the script, so that the program may use the file as input data.

The fun begins when we start writing out own shell scripts, a file that contains a mix of Linux commands and programming concepts. This video is the beginning of the shell scripting journey.



Shell Metacharacters

Metacharacters are special characters that are used to represent something other than themselves. As a rule of thumb, characters that are neither letters nor numbers may be metacharacters. Like grep, sed, and awk, the shell has its own set of metacharacters, often called shell wildcards.

Symbol	Meaning
>	Output redirection
>>	Output redirection (append)
<	Input redirection
*	File substitution wildcard; zero or more characters
?	File substitution wildcard; one character
[ ]	File substitution wildcard; any character between brackets
`cmd`	Command substitution
$(cmd)	Command substitution
|	The Pipe (|)
;	Command sequence, sequences of commands
[ ]	File substitution wildcard; any character between brackets
||	OR conditional execution
&&	AND conditional execution
( )	Group commands, sequences of commands
&	Run command in the background, background processes
#	Comment
$	Expand the value of a variable
\	Prevent or escape interpretation of the next character
<<	Input redirection

Exit Codes

The exit command terminates a script, just as in a C program. It can also return a value, which is available to the script's parent process.

Every command returns an *exit status* (sometimes referred to as a *return status* or *exit code*). A successful command returns a 0, while an unsuccessful one returns a non-zero value that usually can be interpreted as an *error code*. Well-behaved UNIX commands, programs, and utilities return a 0 exit code upon successful completion, though there are some exceptions.

Likewise, functions within a script and the script itself return an exit status. The last command executed in the function or script determines the exit status. Within a script, an exit *nnn* command may be used to deliver an *nnn* exit status to the shell (*nnn* must be an integer in the 0 - 255 range).

When a script ends with an exit that has no parameter, the exit status of the script is the exit status of the last command executed in the script (previous to the exit).

```
#!/bin/bash
COMMAND_1
. . .

COMMAND_LAST
# Will exit with status of last command.

exit
```

The equivalent of a bare exit is exit $? or even just omitting the exit.

```
#!/bin/bash
COMMAND_1
. . .
```

```
COMMAND_LAST

# Will exit with status of last command.
exit $?
```

```
#!/bin/bash
COMMAND1
. . .

COMMAND_LAST
# Will exit with status of last command.
```

$? reads the exit status of the last command executed. After a function returns, $? gives the exit status of the last command executed in the function. This is Bash's way of giving functions a "return value."

Following the execution of a pipe, a $? gives the exit status of the last command executed.

After a script terminates, a $? from the command-line gives the exit status of the script, that is, the last command executed in the script, which is, by convention, **0** on success or an integer in the range 1 - 255 on error.

```
#!/bin/bash

echo hello
echo $?    # Exit status 0 returned because command executed successfully.

lskdf      # Unrecognized command.
echo $?    # Non-zero exit status returned -- command failed to execute.

echo
exit 113   # Will return 113 to shell.
           # To verify this, type "echo $?" after script terminates.

#  By convention, an 'exit 0' indicates success,
#+ while a non-zero exit value means an error or anomalous condition.
#  See the "Exit Codes With Special Meanings" appendix.
```

Adapted from:
"Chapter 6. Exit and Exit Status" by **Multiple Contributors**, The Linux Documentation Project is licensed under CC BY-SA 3.0

13-C.6: Writing and Running Linux Scripts is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-C.7: Shell Expansion / Variable Substitution / Command Substitution

Shell Expansion

After the command has been split into tokens, these tokens or words are expanded or resolved. There are eight kinds of expansion performed, which we will discuss in the next sections, in the order that they are expanded.

After all expansions, quote removal is performed.

Brace Expansion

Brace expansion is a mechanism by which arbitrary strings may be generated. Patterns to be brace-expanded take the form of an optional PREAMBLE, followed by a series of comma-separated strings between a pair of braces, followed by an optional POSTSCRIPT. The preamble is prefixed to each string contained within the braces, and the postscript is then appended to each resulting string, expanding left to right.

Brace expansions may be nested. The results of each expanded string are not sorted; left to right order is preserved:

```
pbmac@pbmac-server $ echo sp{el,il,al}l
spell spill spall
```

Brace expansion is performed before any other expansions, and any characters special to other expansions are preserved in the result. It is strictly textual. Bash does not apply any syntactic interpretation to the context of the expansion or the text between the braces. To avoid conflicts with parameter expansion, the string "${" is not considered eligible for brace expansion.

A correctly-formed brace expansion must contain unquoted opening and closing braces, and at least one unquoted comma. Any incorrectly formed brace expansion is left unchanged.

Tilde Expansion

If a word begins with an unquoted tilde character ("~"), all of the characters up to the first unquoted slash (or all characters, if there is no unquoted slash) are considered a tilde-prefix. If none of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the tilde are treated as a possible login name. If this login name is the null string, the tilde is replaced with the value of the HOME shell variable. If HOME is unset, the home directory of the user executing the shell is substituted instead. Otherwise, the tilde-prefix is replaced with the home directory associated with the specified login name.

If the tilde-prefix is "~+", the value of the shell variable PWD replaces the tilde-prefix. If the tilde-prefix is "~-", the value of the shell variable OLDPWD , if it is set, is substituted.

If the characters following the tilde in the tilde-prefix consist of a number N, optionally prefixed by a "+" or a "-", the tilde-prefix is replaced with the corresponding element from the directory stack, as it would be displayed by the dirs built-in invoked with the characters following tilde in the tilde-prefix as an argument. If the tilde-prefix, without the tilde, consists of a number without a leading "+" or "-", "+" is assumed.

If the login name is invalid, or the tilde expansion fails, the word is left unchanged.

Each variable assignment is checked for unquoted tilde-prefixes immediately following a ":" or "=". In these cases, tilde expansion is also performed. Consequently, one may use file names with tildes in assignments to PATH, MAILPATH, and CDPATH, and the shell assigns the expanded value.

Example:

```
pbmac@pbmac-server $ export PATH="$PATH:~/testdir"
```

~/testdir will be expanded to $HOME /testdir , so if $HOME is /var/home/franky , the directory /var/home/franky/testdir will be added to the content of the PATH variable.

Shell Parameter and Variable Expansion

The "$" character introduces parameter expansion, command substitution, or arithmetic expansion. The parameter name or symbol to be expanded may be enclosed in braces, which are optional but serve to protect the variable to be expanded from characters immediately following it which could be interpreted as part of the name.

When braces are used, the matching ending brace is the first "}" not escaped by a backslash or within a quoted string, and not within an embedded arithmetic expansion, command substitution, or parameter expansion.

The basic form of parameter expansion is "${PARAMETER}". The value of "PARAMETER" is substituted. The braces are required when "PARAMETER" is a positional parameter with more than one digit, or when "PARAMETER" is followed by a character that is not to be interpreted as part of its name.

If the first character of "PARAMETER" is an exclamation point, Bash uses the value of the variable formed from the rest of "PARAMETER" as the name of the variable; this variable is then expanded and that value is used in the rest of the substitution, rather than the value of "PARAMETER" itself. This is known as indirect expansion.

You are certainly familiar with straight parameter expansion, since it happens all the time, even in the simplest of cases, such as the one above or the following:

```
pbmac@pbmac-server $ echo  $SHELL
/bin/bash
```

The following is an example of indirect expansion:

```
pbmac@pbmac-server $ echo  ${!N*}
NNTPPORT NNTPSERVER NPX_PLUGIN_PATH
```

Note that this is not the same as echo $N* .

The following construct allows for creation of the named variable if it does not yet exist:

${ VAR := *value* }

Example:

```
pbmac@pbmac-server $ echo  $PBMAC

pbmac@pbmac-server $ echo  ${PBMAC:=pbmac}
pbmac
```

However special parameters, among others the positional parameters, may not be assigned this way.

Command Substitution

Command substitution allows the output of a command to replace the command itself. Command substitution occurs when a command is enclosed like this:

$(command)

or like this using backticks:

`command`

Bash performs the expansion by executing COMMAND and replacing the command substitution with the standard output of the command, with any trailing newlines deleted. Embedded newlines are not deleted, but they may be removed during word splitting.

```
pbmac@pbmac-server $ echo `date`
Thu Nov 19 17:44:55 PST 2020
```

When the old-style backquoted form of substitution is used, backslash retains its literal meaning except when followed by "$", "`", or "\". The first backticks not preceded by a backslash terminates the command substitution. When using the "$(COMMAND)" form, all characters between the parentheses make up the command; none are treated specially.

Command substitutions may be nested. To nest when using the backquoted form, escape the inner backticks with backslashes.

If the substitution appears within double quotes, word splitting and file name expansion are not performed on the results.

Arithmetic Expansion

Arithmetic expansion allows the evaluation of an arithmetic expression and the substitution of the result. The format for arithmetic expansion is:

$((EXPRESSION))

The expression is treated as if it were within double quotes, but a double quote inside the parentheses is not treated specially. All tokens in the expression undergo parameter expansion, command substitution, and quote removal. Arithmetic substitutions may be nested.

Evaluation of arithmetic expressions is done in fixed-width integers with no check for overflow, although division by zero is trapped and recognized as an error. The operators are roughly the same as in the C programming language. In order of decreasing precedence, the list looks like this:

Arithmetic operators

Operator	Meaning
VAR++ and VAR--	Variable post-increment and post-decrement
++VAR and --VAR	Variable pre-increment and pre-decrement
- and +	Unary minus and plus
! and ~	Logical and bitwise negation
**	Exponentiation
*, / and %	Multiplication, division, remainder
+ and -	Addition, subtraction
<< and >>	Left and right bitwise shifts
<=, >=, < and >	Comparison operators
== and !=	Equality and inequality
&	Bitwise AND
^	Bitwise exclusive OR
|	Bitwise OR
&&	Logical AND
||	Logical OR
expr ? expr : expr	Conditional evaluation
=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^= and |=	Assignments
,	Separator between expressions

Shell variables are allowed as operands; parameter expansion is performed before the expression is evaluated. Within an expression, shell variables may also be referenced by name without using the parameter expansion syntax. The value of a variable is evaluated as an arithmetic expression when it is referenced. A shell variable need not have its integer attribute turned on to be used in an expression.

Constants with a leading 0 (zero) are interpreted as octal numbers. A leading "0x" or "0X" denotes hexadecimal. Otherwise, numbers take the form "[BASE'#']N", where "BASE" is a decimal number between 2 and 64 representing the arithmetic base, and N is a number in that base. If "BASE'#'" is omitted, then base 10 is used. The digits greater than nine are represented by the lowercase letters, the uppercase letters, "@", and "_", in that order. If "BASE" is less than or equal to 36, lowercase and uppercase letters may be used interchangeably to represent numbers between 10 and 35.

Operators are evaluated in order of precedence. Sub-expressions in parentheses are evaluated first and may override the precedence rules above.

Wherever possible, Bash users should try to use the syntax with square brackets:

$[EXPRESSION]

However, this will only calculate the result of EXPRESSION, and do no tests:

```
pbmac@pbmac-server $ echo  $[365*24]
8760
```

Globbing

Bash does carry out *filename expansion* -- a process known as *globbing* -- but this does not use the standard RE set. Instead, globbing recognizes and expands *wild cards*. Globbing interprets the standard wild card characters -- * and ?, character lists in square brackets, and certain other special characters (such as ^ for negating the sense of a match). There are important limitations on wild card characters in globbing, however. Strings containing * will not match filenames that start with a dot, as, for example, .bashrc. Likewise, the ? has a different meaning in globbing than as part of a regular expression.

```
pbmac@pbmac-server $ ls -l
total 2
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 a.1
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1
-rw-rw-r--    1 bozo  bozo       466 Aug  6 17:48 t2.sh
-rw-rw-r--    1 bozo  bozo       758 Jul 30 09:02 test1.txt

pbmac@pbmac-server $ ls -l t?.sh
-rw-rw-r--    1 bozo  bozo       466 Aug  6 17:48 t2.sh

pbmac@pbmac-server $ ls -l [ab]*
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 a.1
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1

pbmac@pbmac-server $ ls -l [a-c]*
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 a.1
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1

pbmac@pbmac-server $ ls -l [^ab]*
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1
-rw-rw-r--    1 bozo  bozo       466 Aug  6 17:48 t2.sh
-rw-rw-r--    1 bozo  bozo       758 Jul 30 09:02 test1.txt

pbmac@pbmac-server $ ls -l {b*,c*,*est*}
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 b.1
-rw-rw-r--    1 bozo  bozo         0 Aug  6 18:42 c.1
-rw-rw-r--    1 bozo  bozo       758 Jul 30 09:02 test1.txt

pbmac@pbmac-server $ echo *
a.1 b.1 c.1 t2.sh test1.txt

pbmac@pbmac-server $ echo t*
t2.sh test1.txt

pbmac@pbmac-server $ echo t?.sh
```

Bash performs filename expansion on unquoted command-line arguments. The echo command demonstrates this.

```
pbmac@pbmac-server $ echo *
a.1 b.1 c.1 t2.sh test1.txt

pbmac@pbmac-server $ echo t*
t2.sh test1.txt

pbmac@pbmac-server $ echo t?.sh
t2.sh
```

Adapted from:
"3.4. Shell expansion" by Multiple Contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0
"18.2. Globbing" by Multiple Contributors, The Linux Documentation Project is licensed under CC BY-SA 3.0

13-C.7: Shell Expansion / Variable Substitution / Command Substitution is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-C.8: Positional Parameters / exec Command / source Command

Command Line Arguments/Positional Parameters

A command line argument is nothing but an argument sent to a program being called. A program can take any number of command line arguments. For example, type the following command:

```
pbmac@pbmac-server $ ls grate_stories_of
grate_stories_of: No such file or directory
pbmac@pbmac-server $ ls great_stories_of
great_stories_of
```

There is no file or directory named **`grate_stories_of`** so the system issues an error message.

`ls` is the name of an actual command and shell executed this command when you type command at shell prompt. The first word on the command line is:

- **`ls`** - name of the command to be executed.
- Everything else on command line is taken as arguments to this command. (**ALTHOUGH** many texts, this course included, make a distinction between options and arguments - with options specifying behavior of the command, whereas the actual arguments specify what to apply the options to.)

Consider the following example:

```
pbmac@pbmac-server $ tail +10 /path/to/file.txt
```

- **tail**: Command name.
- **+10 /path/to/file.txt**: The arguments.

Examples

Try the following command and note down its command line arguments:

Command	Command name	Total number of arguments	Argument name(s)
`ls`	`ls`	`0`	`N/A`
`ls /etc/resolv.conf`	`ls`	`1`	`/etc/resolv.conf`
`cp /etc/resolv.conf /tmp/test.txt`	`cp`	`2`	`/etc/resolv.conf, and /tmp/test.txt`
`sort -r -n /path/to/file`	`sort`	`3`	`-r, -n, and /path/to/file`
`date +"%d-%m-%Y"`	`date`	`1`	`+"%d-%m-%Y"`

Command Line Parameters Different Names

A command line parameter is also known as:

- Command line options
- Options
- Positional parameters
- Flag
- Switches or switch
- Command-line arguments

Why Use Command Line Arguments

- Telling the command/utility which option to use.
- Informing the utility/command which file or group of files to process (reading/writing of files).

Accessing Command-Line Arguments

All command line parameters (positional parameters) are available via special shell variable $1, $2, $3,...,$9.

The simple shell script called cmdargs.sh is shown in this example:

```
#!/bin/bash
echo "The script name : $0"
echo "The value of the first argument to the script : $1"
echo "The value of the second argument to the script : $2"
echo "The value of the third argument to the script : $3"
echo "The number of arguments passed to the script : $#"
echo "The value of all command-line arguments (\$* version) : $*"
echo "The value of all command-line arguments (\$@ version) : $@"
```

When the script is run, the output looks like:

```
pbmac@pbmac-server $ cmdargs.sh bmw ford toyota
The script name : ./cmdargs.sh
The value of the first argument to the script : bmw
The value of the second argument to the script : ford
The value of the third argument to the script : toyota
The number of arguments passed to the script : 3
The value of all command-line arguments ($* version) : bmw ford toyota
The value of all command-line arguments ($@ version) : bmw ford toyota
```

The following are additional examples; the results are in the table that follows.

```
pbmac@pbmac-server $ ls /tmp
pbmac@pbmac-server $ ./math 10 + 3
pbmac@pbmac-server $ ~/scripts/addzone cyberciti.com
pbmac@pbmac-server $ ~/scripts/adddomain cyberciti.biz '74.86.48.99' '2607:f0d0:1002:11::4'
pbmac@pbmac-server $ /etc/init.d/named reload
pbmac@pbmac-server $ /usr/local/etc/rc.d/jail restart cyberciti.biz
```

Shell script name ($0)	Total number of arguments ($#)	Actual Command line argument ($1,..,$9)
ls	1	/tmp
./math	3	10, +, and 3
~/scripts/addzone	1	cyberciti.com
~/scripts/adddomain	3	cyberciti.biz, 74.86.48.99, and 2607:f0d0:1002:11::4
/etc/init.d/named reload	1	reload
/usr/local/etc/rc.d/jail	2	restart, and cyberciti.biz

The Difference Between $@ and $*

- $@ expanded as "$1" "$2" "$3" ... "$n"
- $* expanded as "$1y$2y$3y...$n", where y is the value of $IFS (Input Field Seperator) variable i.e. "$*" is one long string and $IFS act as a separator or token delimiters.

An example showing difference between these two variables:

```
#!/bin/bash

# Set Input FIeld Separator
IFS=", "
echo "* Displaying all pizza names using \$@"
echo "$@"
echo
```

```
# Output includes the field separator - the comma
echo "* Displaying all pizza names using \$*"
echo "$*"
```

When the script runs the output is:

```
pbmac@pbmac-server $ ./pizza.sh
* Displaying all pizza names using $@
Margherita Tomato Panner Gourmet

*Displaying all pizza names using $*
Margherita,Tomato,Panner,Gourmet
```

The **exec** Command

The **exec** command in Linux is used to execute a command from the bash itself. This command does not create a new process, it just replaces the bash with the command to be executed. If the exec command is successful, it does not return to the calling process.

Syntax:

```
exec [-cl] [-a name] [command [arguments]] [redirection ...]
```

Options:

- **-c:** Used to execute the command with empty environment.
- **-a name:** Used to pass a name as the zeroth argument of the command.
- **- l:** Used to pass dash as the zeroth argument of the command.

Note: exec command does not create a new process. When we run the exec command from the terminal, the ongoing terminal process is replaced by the command that is provided as the argument for the exec command.

The exec command can be used in two modes:

1. Exec with a command as an argument**:** In the first mode, the exec tries to execute it as a command passing the remaining arguments, if any, to that command and managing the redirections, if any.

```
# The given command is executed in a different process...and returns the results, then the original
pbmac@pbmac-server $ exec wc -w < tmp
9
pbmac@pbmac-server $
```

2. Exec without a command**:** If no command is supplied, the redirections can be used to modify the current shell environment. This is useful as it allows us to change the file descriptors of the shell as per our desire. The process continues even after the exec command unlike the previous case but now the standard input, output, and error are modified according to the redirections.

```
pbmac@pbmac-server $ bash

This exec creates a new process that overlays the bash shell started by the previous command
pbmac@pbmac-server $ exec > tmp

We are now in a different process....and all of this is being directed to the file names tmp by t
pbmac@pbmac-server $ ls
pbmac@pbmac-server $ echo "This message will not be displayed"
pbmac@pbmac-server $ exit
exit

The exit command exits the other process...and returns to the original context
We see all of the output that had been sent to the file named tmp
pbmac@pbmac-server $ cat tmp
```

```
test2.sh
test.sh
tmp
This message will not be displayed
```

The `source` Command

The **`source`** command is a shell built-in command which is used to read and execute the content of a file (generally a set of commands), passed as an argument in the current shell script. After taking the content of the specified files, the command passes it to the TCL interpreter as a text script which then gets executed. If any arguments are supplied, they become the positional parameters when filename is executed. Otherwise, the positional parameters remain unchanged. The entries in $PATH are used to find the directory containing FILENAME, however if the file is not present in $PATH it will search the file in the current directory. The source command has no option and the argument is the file only.

Syntax:

```
source FILENAME [arguments]
```

Using the source command is very similar to writing a shell script, but there is no requirement for a shebang line, and the file does NOT have to be set to executable. If there is a file that has the following two lines:

```
pbmac@pbmac-server $ cat test-3.txt
ls
cal
```

Then to source the file:

```
pbmac@pbmac-server $ source test-3.txt
CPP      get-pip.py   Linux           t1       test2
'CS 11'   Java         myFile.txt.gz   t1.cpp   test2.cpp
CSP31B   JavaScript   Python          test1    WebStuff
   November 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
```

Adapted from:
"How to use positional parameters" by Vivek Gite is licensed under CC BY-NC 3.0
"exec command in Linux with examples" by anindo_7, Geeks for Geeks is licensed under CC BY-SA 4.0
"source command in Linux with Examples" by Suraj1994, Geeks for Geeks is licensed under CC BY-SA 4.0

13-C.8: Positional Parameters / exec Command / source Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-C.9: Script Permissions / Script Extensions

Script Permissions

A shell script can be executed using the following syntax:

```
pbmac@pbmac-server $ chmod +x script.sh
pbmac@pbmac-server $ ./script.sh

Normally the cwd, that is the current directory, is not in the PATH. If it is, then yo
the script name.
```

The script can also be run directly without setting the script execute permission by running a new instance of bash, or by using the dot format, which is the same as using the source command.

```
bash script.sh
. script.sh
```

Script Extensions

An often asked question is what is the appropriate extension for a shell script. In reality, the extension has no bearing, in Linux, on how the script gets executed. It is the shebang line that determines what shell is used for the execution of the script. There are some exceptions, noted above, in the discussion of permissions, where a script can be executed in the current shell environment.

The convention is that shell scripts have the .sh extension. However, this is only a visual help to someone looking through a list of files, knowing that all of the .sh files are shell scripts. An organization can create their own standards, without fear of going against any standard.

13-C.9: Script Permissions / Script Extensions is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-D.10: Shell Control Statements

EXAM OBJECTIVES COVERED

5.1 Given a scenario, deploy and execute basic Bash scripts.

There are three components that we need to understand in any structured programming methodology:

1. Sequential execution
2. Conditional execution
3. Looping constructs

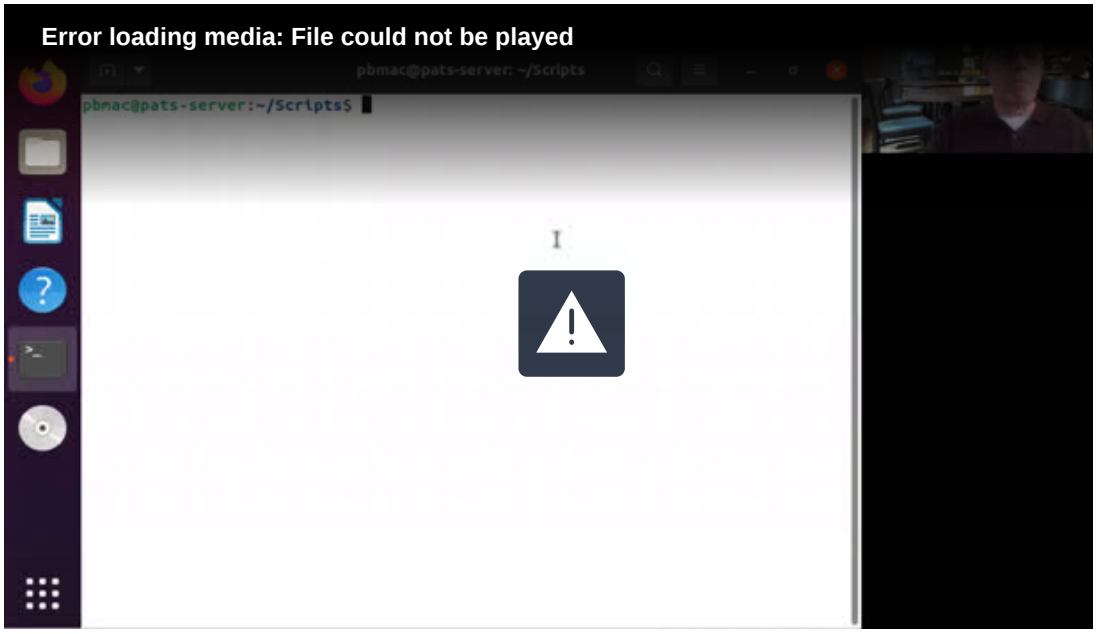
Each of these will be discussed as we learn how to implement control structures in a shell script.

Sequential Execution

Sequential execution means that each command in a program script executes in the order in which it is listed in the program. The first command in the sequence executes first and when it is complete, the second command executes, and so on. The presence of functions in the code does not negate sequential execution; we can still follow the sequential flow of the instructions. As technology advances there is more parallel programming being run on smaller computers, but how that impacts sequential execution is beyond the scope of this course.

Conditional Execution

The following video gives an introduction to the concept of conditional execution. More in depth discussion follows the video.



Conditional Statements: There are a total of five conditional statements which can be used in bash programming.

1. if statement
2. if-else statement
3. if..elif..else..fi statement (Else If ladder)
4. if..then..else..if..then..fi..fi..(Nested if)
5. switch statement

Their description with syntax is as follows:

if statement

This block will process if specified condition is true.
Syntax:

```
if [ expression ]
then
```

```
   statement
fi
```

An example of the code:

```
#Initializing two variables
a=20
b=20

if [ $a == $b ]
then
    #If they are equal then print this
    echo "a is equal to b"
else
    #else print this
    echo "a is not equal to b"
fi
```

if-else statement

If specified condition is not true in if part then else part will be executed.
Syntax:

```
if [ expression ]
then
   statement1
else
   statement2
fi
```

if..elif..else..fi statement (Else If ladder)

To use multiple conditions in one if-else block, then the elif keyword is used in shell. If expression1 is true then it executes statement 1 and 2, and this process continues. If none of the conditions is true then it processes else part.
Syntax:

```
if [ expression1 ]
then
   statement1
   statement2
   .
   .
elif [ expression2 ]
then
   statement3
   statement4
   .
   .
else
   statement5
fi
```

Switch Statement

A case statement works as a switch statement, if specified values match with the pattern then it will execute a block of that particular pattern. When a match is found all of the associated statements until the double semicolon (;;) are executed. A case will be terminated when the last command is

executed. If there is no match, the exit status of the case is zero.

Syntax:

```
case in
Pattern 1) Statement 1;;
Pattern n) Statement n;;
esac
```

An example of a switch statement is shown below. Lines beginning with a hash tag are comments.

```
CARS="bmw"

#Pass the variable in string
case "$CARS" in
    #case 1
    "mercedes") echo "Headquarters - Affalterbach, Germany" ;;

    #case 2
    "audi") echo "Headquarters - Ingolstadt, Germany" ;;

    #case 3
    "bmw") echo "Headquarters - Chennai, Tamil Nadu, India" ;;
esac
```

Test Statement

These operators are used to test a particular property of a file.

- **-b operator**: This operator checks whether a file is a block special file or not. It returns true if the file is a block special file otherwise returns false.
- **-c operator**: This operator checks whether a file is a character special file or not. It returns true if it is a character special file otherwise returns false.
- **-d operator**: This operator checks if the given directory exists or not. If it exists then operators returns true otherwise returns false.
- **-e operator**: This operator checks whether the given file exists or not. If it exists this operator returns true otherwise returns false.
- **-r operator**: This operator checks whether the given file has read access or not. If it has read access then it returns true otherwise returns false.
- **-w operator**: This operator checks whether the given file has write access or not. If it has write then it returns true otherwise returns false.
- **-x operator**: This operator checks whether the given file has execute access or not. If it has execute access then it returns true otherwise returns false.
- **-s operator**: This operator checks the size of the given file. If the size of given file is greater than 0 then it returns true otherwise it returns false.

```
#!/bin/bash

#reading data from the user
FileName=/home/pbmac

if [ -d $FileName ]
then
    echo "The directory $FileName exists"
else
    echo "The directory $FileName does NOT exists"
fi
```

Adapted from:
"Conditional Statements | Shell Script" by bilal-hungund, Geeks for Geeks is licensed under CC BY-SA 4.0
"Basic Operators in Shell Scripting" by DrRoot_, Geeks for Geeks is licensed under CC BY-SA 4.0

13-D.10: Shell Control Statements is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

13-D.11: Script Loops

Looping Shell Scripts

There are a total of three looping statements which can be used in bash programming:

1. while statement
2. for statement
3. until statement

While Statement

Here the command is evaluated and based on the result loop will be executed; if command is raised to false then loop will be terminated.

```
while command
do
   Statement to be executed
done
```

Here is an example of a while loop. It looks very similar to most programming language while loops.

```
#!/bin/bash

count=1

# Continue in the loop until the variable 'count' is less than 6
# -lt is less than operator
while [ $count -lt 6 ]
do
    # Print the values
    echo $count

    # increment the value
    # this is an odd looking increment of count
    # it could also be done as: ((++count))
    count=`expr $count + 1`
done
```

The output of the while loop would be:

```
1
2
3
4
5
```

For Statement

The for loop operates on lists of items. It repeats a set of commands for every item in a list.

Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the value of the variable var is set to the next word in the list of words, word1 to wordN.

```
for var in word1 word2 ...wordn
do
    Statement to be executed
done
```

In the following example the variable count is set to the next numerical value from the list (1 2 3 4 5). The echo statement prints the output.

```
#!/bin/bash

#Start of for loop
for count in 1 2 3 4 5
do
    # Print the value
    echo "Loop number $count"
done
```

The output is:

```
Loop number 1
Loop number 2
Loop number 3
Loop number 4
Loop number 5
```

Until Statement

The until loop is executed as many times as the condition/command evaluates to false. The loop terminates when the condition/command becomes true.

```
#!/bin/bash

until command
do
   Statement to be executed until command is true
done
```

The example of an until loop:

```
#!/bin/bash

count=1
# -gt is greater than operator

#Iterate the loop until count is greater than 5
until [ $count -gt 5 ]
```

```
do
    # Print the values
    echo $count

    # increment the value
    count=`expr $count + 1`
done
```

Running this script produces the following output:

```
1
2
3
4
5
```

Adapted from:
"Looping Statements | Shell Script" by bilal-hungund, Geeks for Geeks is licensed under CC BY-SA 4.0

13-D.11: Script Loops is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 14: Task Automation

14-A.1: Job Scheduling - at Command

14-A.2: Cron

14-B.3: Git: Version Control

14-C.4: Orchestration

Module 14: Task Automation is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by LibreTexts.

14-A.1: Job Scheduling - at Command

EXAM OBJECTIVES COVERED

2.6 Given a scenario, automate and schedule jobs.

Objectives of this Module

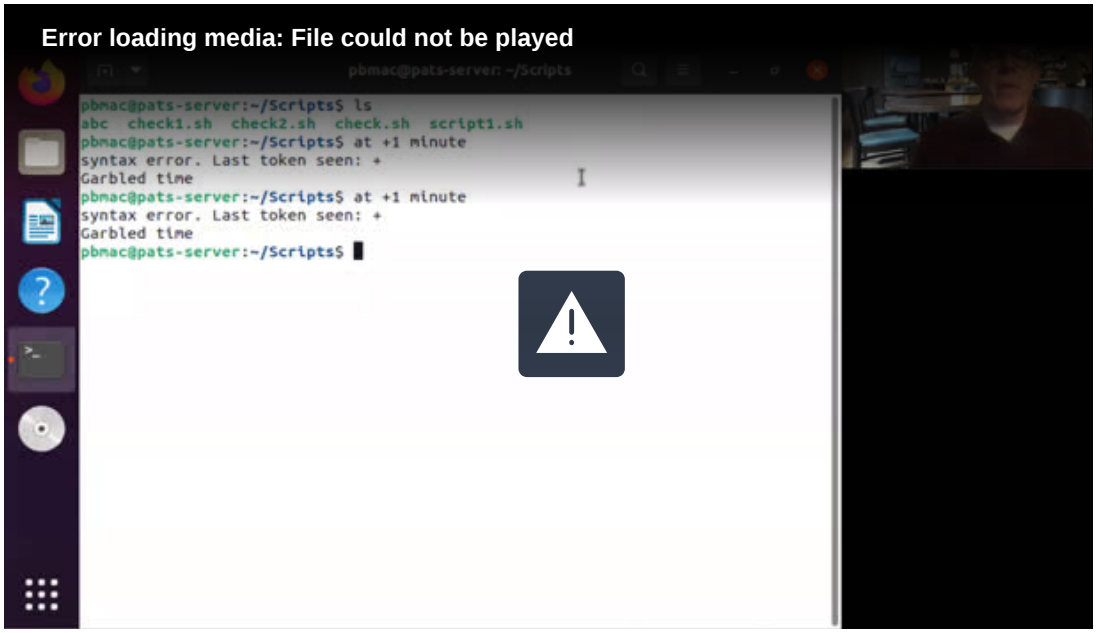
In this lesson, you will:

- Learn to run jobs on a set schedule
- Configure and run version control for files using Git
- Describe concepts fundamental to orchestration

The `at` Command

The **`at`** command is a command-line utility that is used to schedule a command to be executed at a particular time in the future. Jobs created with **`at`** command are executed only once. The **`at`** command can be used to execute any program or mail at any time in the future. It executes commands at a particular time and accepts times of the form HH:MM to run a job at a specific time of day.

A bit of a video overview of using the **`at`** command to show some examples before we get into the details.



Syntax:

```
at [ OPTION ] runtime
```

Options:

Options	Option Meaning
-m	Sends mail to the user when the job has completed even if there was no output.
-f file	Reads the job from file rather than standard input.
-l	Is an alias for atq.
-d	Is an alias for atrm.
-v	Shows the time the job will be executed before reading the job.
-c	Cats the jobs listed on the command line to standard output.
-t time_arg	Submits the job to be run at the time specified by the time_arg option argument, which must have the same format as specified for the touch(1) utility's -t time option argument ([[CC]YY]MMDDhhmm).

The runtime argument has a rather vague syntax. The following expressions like noon, midnight, teatime, tomorrow, next week, next Monday, etc. could be used with at command to schedule a job. This table shows numerous examples of this syntax. This assumes the current time is Wed Nov 18 08:00:00 PST 2020.

Time Expression	Time of Command Execution
noon	12:00 PM November 18 2020
midnight	12:00 AM November 19 2020
tomorrow	8:00 AM November 19 2020
noon tomorrow	12:00 PM November 19 2020
next week	8:00 AM November 25 2020
next Monday	8:00 AM November 23 2020
fri	8:00 AM November 20 2020
6:00 AM	6:00 AM November 19 2020
2:30 PM	2:30 PM November 18 2020
1430	2:30 PM November 18 2020
2:30 PM tomorrow	2:30 PM November 19 2020
2:30 PM next month	2:30 PM December 18 2020
2:30 PM 11/21	2:30 PM November 21 2020
2:30 PM Nov 21	2:30 PM November 21 2020
2:30 PM 11/21/2020	2:30 PM November 21 2020
now + 30 minutes	8:30 AM November 18 2020
now + 1 hour	9:00 AM November 18 2020
now + 2 days	8:00 AM November 20 2020
4 PM + 2 days	4:00 PM November 20 2020
now + 3 weeks	8:00 AM December 8 2020
now + 4 months	8:00 AM March 18 2021

There are four commands that are part of the **`at`** command set:

Command	**Explanation**
at	Executes commands at a specified time.
atq	Lists the user's pending jobs, unless the user is the superuser; in that case, everybody's jobs are listed. The format of the output lines (one for each job) is: Job number, date, hour, queue, and username.
atrm	Deletes jobs, identified by their job number.
batch	Executes commands when system load levels permit; in other words, when the load average drops below 0.8, or the value specified in the invocation of **atd**.

The following example illustrates some of the **`at`** command: 1) set the execution to 30 minutes from the current time; 2) the "at>" prompt we type in commands; 3) the <EOT> happens when we enter a ctrl-D (control-d keys pressed simultaneously); 4) **`atq`** show all **`at`** commands in our queue; 5) **`atrm`** removes the command sitting in the number 2 slot in our queue - which we obtained from the output of **`atq`**; 6) the second **`atq`** shows nothing in the queue.

```
pbmac@pbmac-server $ at now + 30 minutes
warning: commands will be executed using /bin/sh
at> echo "Hello World"
at> <EOT>
```

```
job 2 at Wed Nov 25 13:33:00 2020
pbmac@pbmac-server $ atq
2    Wed Nov 25 13:33:00 2020 a pbmac
pbmac@pbmac-server $ atrm 2
pbmac@pbmac-server $ atq
pbmac@pbmac-server $
```

Adapted from:
"at Command in Linux with Examples" by vizvasya, Geeks for Geeks is licensed under CC BY-SA 4.0

14-A.1: Job Scheduling - at Command is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

14-A.2: Cron

Cron

Common (And Uncommon) Cron Uses

Administrators use the **cron** service to schedule obvious things, such as regular backups that occur daily at 2 a.m. It can also be used for less obvious things.

- The system times (i.e., the operating system time) on my many computers are set using the Network Time Protocol (NTP). While NTP sets the system time, it does not set the hardware time, which can drift. I use **cron** to set the hardware time based on the system time.
- I also have a Bash program I run early every morning that creates a new "message of the day" (MOTD) on each computer. It contains information, such as disk usage, that should be current in order to be useful.
- Many system processes and services, like Logwatch, logrotate, and Rootkit Hunter, use the **cron** service to schedule tasks and run programs every day.

The crond daemon is the background service that enables **cron** functionality.

The **cron** service checks for files in the /var/spool/cron and /etc/cron.d directories and the /etc/anacrontab file. The contents of these files define **cron** jobs that are to be run at various intervals. The individual user **cron** files are located in /var/spool/cron, and system services and applications generally add **cron** job files in the /etc/cron.d directory. The /etc/anacrontab is a special case that will be covered later in this article.

Using Crontab

The **cron** utility runs based on commands specified in a **cron** table (crontab). Each user, including root, can have a **cron** file. These files don't exist by default, but can be created in the /var/spool/cron directory using the crontab -e command that's also used to edit a **cron** file (see the script below). It is strongly recommended that you do not use any of the standard editors (such as Vi, Vim, Emacs, Nano, or any of the many other editors that are available). Using the crontab command not only allows you to edit the command, it also restarts the crond daemon when you save and exit the editor. The **crontab** command uses Vi as its underlying editor, because Vi is always present (on even the most basic of installations).

New **cron** files are empty, so commands must be added from scratch.The job definition example below is a quick reference, to help remember what the various parts of a command mean.

```
# crontab -e
SHELL=/bin/bash
MAILTO=root@example.com
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed

# backup using the rsbu program to the internal 4TB HDD and then 4TB external
```

```
01 01 * * * /usr/local/bin/rsbu -vbd1 ; /usr/local/bin/rsbu -vbd2

# Set the hardware clock to keep it in sync with the more accurate system clock
03 05 * * * /sbin/hwclock --systohc

# Perform monthly updates on the first of the month
# 25 04 1 * * /usr/bin/dnf -y update
```

The first three lines in the code above set up a default environment. The environment must be set to whatever is necessary for a given user because **cron** does not provide an environment of any kind. The SHELL variable specifies the shell to use when commands are executed. This example specifies the Bash shell. The MAILTO variable sets the email address where **cron** job results will be sent. These emails can provide the status of the **cron** job (backups, updates, etc.) and consist of the output you would see if you ran the program manually from the command line. The third line sets up the PATH for the environment. Even though the path is set here, I always prepend the fully qualified path to each executable.

There are several comment lines in the example above that detail the syntax required to define a **cron** job. We'll break those commands down, and add a few more to show some more advanced capabilities of crontab files.

```
01 01 * * * /usr/local/bin/rsbu -vbd1 ; /usr/local/bin/rsbu -vbd2
```

This line runs a user-written Bash shell script, rsbu, that backs up all the user's systems. This job kicks off at 1:01 a.m. (01 01) every day. The asterisks (*) in positions three, four, and five of the time specification are like file globs, or wildcards, for other time divisions; they specify "every day of the month," "every month," and "every day of the week." This line runs the backups twice; one backs up to an internal dedicated backup hard drive, and the other backs up to an external USB drive that can be taken to the safe deposit box.

The following line sets the hardware clock on the computer using the system clock as the source of an accurate time. This line is set to run at 5:03 a.m. (03 05) every day.

```
03 05 * * * /sbin/hwclock --systohc
```

The third and final **cron** job (commented out) is to perform a **dnf** or **yum** update at 04:25 a.m. on the first day of each month, but it is currently commented out so it no longer runs.

```
# 25 04 1 * * /usr/bin/dnf -y update
```

Other Scheduling Tricks

Now let's do some things that are a little more interesting than these basics. Suppose you want to run a particular job every Thursday at 3 p.m.:

```
00 15 * * Thu /usr/local/bin/mycronjob.sh
```

Or, maybe you need to run quarterly reports after the end of each quarter. The **cron** service has no option for "The last day of the month," so instead you can use the first day of the following month, as shown below. (This assumes that the data needed for the reports will be ready when the job is set to run.)

```
02 03 1 1,4,7,10 * /usr/local/bin/reports.sh
```

This **cron** job runs quarterly reports on the first day of the month after a quarter ends.

The following shows a job that runs one minute past every hour between 9:01 a.m. and 5:01 p.m.

```
01 09-17 * * * /usr/local/bin/hourlyreminder.sh
```

There are many other situations where a job needs to run every two, three, or four hours. That can be accomplished by dividing the hours by the desired interval, such as ***/3** for every three hours, or **6-18/3** to run every three hours between 6 a.m. and 6 p.m. Other intervals can be divided similarly; for example, the expression ***/15** in the minutes position means "run the job every 15 minutes."

```
*/5 08-18/2 * * * /usr/local/bin/mycronjob.sh
```

One thing to note: The division expressions must result in a remainder of zero for the job to run. That's why, in this example, the job is set to run every five minutes (08:05, 08:10, 08:15, etc.) during even-numbered hours from 8 a.m. to 6 p.m., but not during any odd-numbered hours. For example, the job will not run at all from 9 p.m. to 9:59 a.m.

Limiting **cron** access

Regular users with **cron** access could make mistakes that, for example, might cause system resources (such as memory and CPU time) to be swamped. To prevent possible misuse, the sysadmin can limit user access by creating a /etc/cron.allow file that contains a list of all users with permission to create **cron** jobs. The root user cannot be prevented from using cron.

By preventing non-root users from creating their own **cron** jobs, it may be necessary for root to add their **cron** jobs to the root crontab. "But wait!" you say. "Doesn't that run those jobs as root?" Not necessarily. The username field shown in the comments can be used to specify the user ID a job is to have when it runs. This prevents the specified non-root user's jobs from running as root. The following example shows a job definition that runs a job as the user "student":

```
04 07 * * * student /usr/local/bin/mycronjob.sh
```

If no user is specified, the job is run as the user that owns the crontab file, root in this case.

cron.d

The directory /etc/cron.d is where some applications, such as SpamAssassin and sysstat, install **cron** files. Because there is no spamassassin or sysstat user, these programs need a place to locate **cron** files, so they are placed in /etc/cron.d.

The /etc/cron.d/sysstat file below contains **cron** jobs that relate to system activity reporting (SAR). These **cron** files have the same format as a user **cron** file.

```
# Run system activity accounting tool every 10 minutes
*/10 * * * * root /usr/lib64/sa/sa1 1 1
# Generate a daily summary of process accounting at 23:53
53 23 * * * root /usr/lib64/sa/sa2 -A
```

The sysstat package installs the /etc/cron.d/sysstat cron file to run programs for SAR.

The sysstat **cron** file has two lines that perform tasks. The first line runs the sa1 program every 10 minutes to collect data stored in special binary files in the /var/log/sa directory. Then, every night at 23:53, the sa2 program runs to create a daily summary.

Adapted from:
"How to use cron in Linux" by David Both, OpenSource.com is licensed under CC BY-SA 4.0

14-A.2: Cron is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

14-B.3: Git: Version Control

Git

Nowadays software development takes place in a distributive way. This article focuses on one such technology that supports distributed software development i.e., Git.

What is Git about?

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Git relies on the basis of distributed development of a software, where more than one developer may have access to the source code of a specific application and can modify changes to it, which may be seen by other developers.
- Initially designed and developed by Linus Torvalds for Linux kernel development in 2005.
- Every git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server.
- Git allows a team of people to work together, all using the same files, and it helps the team cope with the confusion that tends to happen when multiple people are editing the same files.

Why Use Version Control Software?

- Version control software allows the user to have “versions” of a project, which show the changes that were made to the code over time, and allows the user to backtrack if necessary and undo those changes.
- This ability alone, of being able to compare two versions or reverse changes, makes it fairly invaluable when working on larger projects.
- In a version control system, the changes would be saved just in time – a patch file that could be applied to one version, in order to make it the same as the next version.
- All versions are stored on a central server, and individual developers checkout and upload changes back to this server.

Characteristics of Git

1. Strong support for non-linear development
    - Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history.
    - A major assumption in Git is that a change will be merged more often than it is written.
    - Branches in Git are very lightweight.
2. Distributed development
    - Git provides each developer a local copy of the entire development history, and changes are copied from one such repository to another.
    - The changes can be merged in the same way as a locally developed branch very efficiently and effectively.
3. Compatibility with existing systems/protocol
    - Git has a CVS server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories.
4. Efficient handling of large projects
    - Git is very fast and scalable compared to other version control systems.
    - The fetching power from a local repository is much faster than what is possible with a remote server.
5. Data assurance
    - The Git history is stored in such a way that the ID of a particular version depends upon the complete development history leading up to that commit.
    - Once published, it is not possible to change the old versions without it being noticed.
6. Automatic garbage collection
    - Git automatically performs garbage collection when enough loose objects have been created in the repository

- Git automatically performs garbage collection when enough loose objects have been created in the repository.
- Garbage collection can be called explicitly using git gc –prune.

7. Periodic explicit object packing
    - Git stores each newly created object as a separate file. It uses packs that store a large number of objects in a single file (or network byte stream) called packfile, which are delta-compressed among themselves.
    - A corresponding index file is created for each pack file, specifying the offset of each object in the packfile.
    - The process of packing can be very expensive computationally.
    - Git allows the expensive pack operation to be deferred until later when time does not matter.
    - Git does periodic repacking automatically but manual repacking can be done with the git gc command.

How Git Works

1. A Git repository is a key-value object store where all objects are indexed by their SHA-1 hash value.
2. All commits, files, tags and filesystem tree nodes are different types of objects living in this repository.
3. A Git repository is a large hash table with no provision made for hash collisions.
4. Git specifically works by taking “snapshots” of files.

Git Repository Structure

The structure consists of four parts:

1. Working Directory: This is your local directory where you make the project (write code) and make changes to it.
2. Staging Area (or index): This is an area where you first need to put your project before committing. This is used for code review by other team members.
3. Local Repository: This is your local repository where you commit changes to the project before pushing them to central repository on Github. This is what is provided by distributed version control system. This corresponds to the .git folder in our directory.
4. Central Repository: This is the main project on the central server, a copy of which is with every team member as local repository.

All the repository structure is internal to Git and transparent to developer.

Git Command Examples

To add a file Readme.txt to the staging area to track its changes:

```
pbmac@pbmac-server $ git add Readme.txt
```

To create a new branch named as Testing:

```
pbmac@pbmac-server $ git branch Testing
```

To switch to branch Testing from master branch:

```
pbmac@pbmac-server $ git checkout Testing
```

To clone or make a local copy of the global repository in your system:

```
pbmac@pbmac-server $ git clone https://github.com/pbmac/TestFile.pdf
```

To commit our changes (take a snapshot) and provide a message to remember for future reference:

```
pbmac@pbmac-server $ git commit -m “Created a Readme.txt”
```

To set the basic configurations on github like your name and email:

```
pbmac@pbmac-server $ git config
```

To create a local git repository for us in our store folder (this will help to manage the git commands for that particular repository):

```
pbmac@pbmac-server $ git init
```

To check the history of commits for our reference:

```
pbmac@pbmac-server $ git log
```

To merge Testing branch with master branch:

```
pbmac@pbmac-server $ git merge Testing
```

To push all the contents of our local repository that belong to master branch to the server (Global repository):

```
pbmac@pbmac-server $ git push -u origin master
```

To see what's changed since last commit; shows all the files that have been added and modified and ready to be committed and files which are untracked:

```
pbmac@pbmac-server $ git status
```

Setting Up Repositories

In the following example, the user enters their user.name and their user.email - DO NOT simply type "user.name" or "user.email" - that is incorrect.

```
# Configure global settings
pbmac@pbmac-server $ git config --global user.name "Your name here"
pbmac@pbmac-server $ git config --global user.email "your_email@example.com"

# then we make a directory for the git project
pbmac@pbmac-server $ mkdir newProject
pbmac@pbmac-server $ cd newProject/

# Need to initialize the directory
pbmac@pbmac-server $ git init
Initialized empty Git repository in /home/pbmac/Development/newProject/.git/
pbmac@pbmac-server $ git add *.cpp

create a code file
pbmac@pbmac-server $ edit file1.cpp

# add it to the repository
pbmac@pbmac-server $ git add *.cpp
```

```
# Commit the changes to the repository server
pbmac@pbmac-server $ git commit -m 'test initial versions'
[master (root-commit) c9ed3f6] test initial versions
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.cpp
```

Branching and Merging

In particular, pay attention to branching: Create a separate branch to develop a feature (or work on a bug) without disturbing the master branch. If it works out, you can merge it back into the master; if it doesn't, you can trash it. Branching is super easy, so for big projects, you should probably do it more often than not.

To create a branch called new_feature:

```
pbmac@pbmac-server $ git branch new_feature
```

Then "check it out":

```
pbmac@pbmac-server $ git checkout new_feature
```

Make various modifications, and then add and commit. To go back to the master branch, check it out:

```
pbmac@pbmac-server $ git checkout master
```

To push the branch to github, use this:

```
pbmac@pbmac-server $ git push origin new_feature
```

If you make changes to the master branch, you'll want to merge them into your exploratory one:

```
pbmac@pbmac-server $ git checkout new_feature
pbmac@pbmac-server $ git merge master
```

If you're satisfied with your changes in the exploratory branch, merge them into the master:

```
pbmac@pbmac-server $ git checkout master
pbmac@pbmac-server $ git merge new_feature
```

If you're done with the branch and want to delete it:

```
pbmac@pbmac-server $ git branch -d new_feature
```

Git Collaboration

One of git's strengths is the ability for multiple people to collaborate on a project. There are different methods to make collaboration work; one of them is described below. In this example we will have the currentMaster code, and we will have the newFeature code.

1. Update the local repository with the currentMaster, which is the known and working branch of the project:

```
git pull currentMaster
```

2. Create a new branch where the new feature code will be added and tested:

```
git checkout -b newFeature
```

3. Once the changes have been made on this branch, they need to be staged for commit:

```
git add
```

4. Now the changes need to be committed on the local repository:

```
git commit -m "Added the new feature as requested by the client"
```

5. Upload the changes (including the newFeature branch) to GitHub:

```
git push currentMaster newFeature
```

6. Go to the main repository on GitHub - the newFeature branch should now be visible there.
7. Click on the newFeature branch.
8. Click on “Pull Request” button - this asks for the code to be merged into the main (currentMaster) branch on GitHub.
9. Click on “Send Pull Request” which sends the pull request to the individual on the team in charge of merging changes.

What is .gitignore

Files in your working Git repository can be:

1. Untracked: Changes that have not been staged or committed
2. Tracked: Changes that have been staged or committed
3. Ignored: Files you tell Git to ignore

There are some files you want Git to ignore and not track in your repository. These include many that are auto-generated or platform-specific, as well as other local configuration files such as:

1. Files with sensitive information
2. Compiled code, such as .dll or .class
3. System files like .DS_Store or Thumbs.db
4. Files with temporary information such as logs, caches, etc.
5. Generated files such as dist folders

If you don't want Git to track certain files in your repository, there is no Git command you can use. (Although you can stop tracking a file with the git rm command, such as git rm --cached .) Instead, you need to use a .gitignore file, a text file that tells Git which files not to track.

It's easy to create a .gitignore file; just create a text file and name it .gitignore. Remember to add a single dot (.) at the beginning of the file name. That's it!

Adapted from:
"git/github guide a minimal tutorial" by Karl W. Broman is licensed under CC BY 4.0
"Version Control with Git - Collaborating" by The Carpentries is licensed under CC BY 4.0
"Don't ignore .gitignore" by Rajeev Bera, OpenSource.com is licensed under CC BY-SA 4.0

14-B.3: Git: Version Control is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

14-C.4: Orchestration

Orchestration and Automation: What is the Difference?

For the longest time, it seemed the only thing any sysadmin cared about was automation. Recently, though, the mantra seems to have changed from automation to orchestration, leading many puzzled admins to wonder: "What's the difference?"

The difference between automation and orchestration is primarily in intent and tooling. Technically, automation can be considered a subset of orchestration. While orchestration suggests many moving parts, automation usually refers to a singular task or a small number of strongly related tasks. Orchestration works at a higher level and is expected to make decisions based on changing conditions and requirements.

However, this view shouldn't be taken too literally because both terms—automation and orchestration—do have implications when they're used. The results of both are functionally the same: things happen without your direct intervention. But the way you get to those results, and the tools you use to make them happen, are different, or at least the terms are used differently depending on what tools you've used.

For instance, automation usually involves scripting, often in Bash or Python or similar, and it often suggests scheduling something to happen at either a precise time or upon a specific event. However, orchestration often begins with an application that's purpose-built for a set of tasks that may happen irregularly, on demand, or as a result of any number of trigger events, and the exact results may even depend on a variety of conditions.

Decision Making and IT Orchestration

Automation suggests that a sysadmin has invented a system to cause a computer to do something that would normally have to be done manually. In automation, the sysadmin has already made most of the decisions on what needs to be done, and all the computer must do is execute a "recipe" of tasks.

Orchestration suggests that a sysadmin has set up a system to do something on its own based on a set of rules, parameters, and observations. In orchestration, the sysadmin knows the desired end result but leaves it up to the computer to decide what to do.

Consider Ansible and Bash. Bash is a popular shell and scripting language used by sysadmins to accomplish practically everything they do during a given workday. Automating with Bash is straightforward: Instead of typing commands into an interactive session, you type them into a text document and save the file as a shell script. Bash runs the shell script, executing each command in succession. There's room for some conditional decision-making, but usually, it's no more complex than simple if-then statements, each of which must be coded into the script.

Ansible, on the other hand, uses playbooks in which a sysadmin describes the desired state of the computer. It lists requirements that must be met before Ansible can consider the job done. When Ansible runs, it takes action based on the current state of the computer compared to the desired state, based on the computer's operating system, and so on. A playbook doesn't contain specific commands, instead leaving those decisions up to Ansible itself.

Of course, it's particularly revealing that Ansible is referred to as an automation—not an orchestration—tool. The difference can be subtle, and the terms definitely overlap.

Orchestration and the Cloud

Say you need to convert a file type that's regularly uploaded to your server by your users.

The manual solution would be to check a directory for uploaded content every morning, open the file, and then save it in a different format. This solution is slow, inefficient, and probably could happen only once every 24 hours because you're a busy person.

You could automate the task. Were you to do that, you might write a PHP or a Node.js script to detect when a file has been uploaded. The script would perform the conversion and send an alert or make a log entry to confirm the conversion was successful. You could improve the script over time to allow users to interact with the upload and conversion process.

Were you to orchestrate the process, you might instead start with an application. Your custom app would be designed to accept and convert files. You might run the application in a container on your cloud, and using OpenShift, you could launch additional instances of your app when the traffic or workload increases beyond a certain threshold.

Learning Automation and Orchestration

There isn't just one discipline for automation or orchestration. These are broad practices that are applied to many different tasks across many different industries. The first step to learning, though, is to become proficient with the technology you're meant to orchestrate and automate. It's difficult to orchestrate (safely) the scaling a series of web servers if you don't understand how a web server works, or what ports need to be open or closed, or what a port is. In practice, you may not be the person opening ports or configuring the server; you could be tasked with administrating OpenShift without really knowing or caring what's inside a container. But basic concepts are important because they broadly apply to usability, troubleshooting, and security.

Infrastructure as Code

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools. The IT infrastructure managed by this process comprises both physical equipment, such as bare-metal servers, as well as virtual machines, and associated configuration resources. The definitions may be in a version control system. It can use either scripts or declarative definitions, rather than manual processes, but the term is more often used to promote declarative approaches.

Orchestration Tools

Tool	Description
Ansible	Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs. Designed for multi-tier deployments since day one, Ansible models your IT infrastructure by describing how all of your systems inter-relate, rather than just managing one system at a time.
Chef	Chef operates in a client-server architecture. An agent, the Chef Client, runs on each managed node, periodically connecting to a Chef Server to download and evaluate configuration code, known as recipes. If no changes are necessary, Chef does not modify the system.
Kubernetes	Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available. The name Kubernetes originates from Greek, meaning helmsman or pilot. Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.
Nomad	Nomad is a simple, flexible, and easy to use workload orchestrator to deploy and manage containers and non-containerized applications across on-prem and clouds at scale. Nomad runs as a single binary with a small resource footprint (35MB) and is supported on macOS, Windows, Linux. Developers use declarative infrastructure-as-code (IaC) for deploying their applications and defining how an application should be deployed. Nomad automatically recovers applications from failures.
OpenStack	OpenStack is an open source platform that uses pooled virtual resources to build and manage private and public clouds. The tools that comprise the OpenStack platform, called "projects," handle the core cloud-computing services of compute, networking, storage, identity, and image services. More than a dozen optional projects can also be bundled together to create unique, deployable clouds.
OpenShift	Redhat offers OpenShift Container Platform as a Service (PaaS). It helps in the automation of applications on secure and scalable resources in hybrid cloud environments. It provides enterprise-grade platforms for building, deployment, and managing containerized applications. Built on Redhat enterprise Linux and Kubernetes engine, Openshift has various functionalities to manage clusters via UI and CLI.
Rancher	Rancher is an open-source platform that uses container

Tool	Description
	orchestration known as cattle. It allows you to leverage orchestration services like Kubernetes, Swarm, or Mesos. Rancher provides the software required to manage containers so that organizations don’t need to build container services platforms from scratch using a distinct set of open source technologies.

Orchestration Configuration

With such a wide variety of tools there is not an easy description of how to configure orchestration. Each tool has its own syntax for the config files, and the various attributes that need to be configured. The best approach would be to take some time to research a short list of potential tools, and investigate those specific tools. Once a platform is selected, spend some time gaining training and/or experience with that tool before proceeding with an implementation.

Adapted from:
"What's the difference between orchestration and automation?" by Seth Kenlon, **OpenSource.com** is licensed under CC BY-SA 4.0
"Infrastructure as code" by Multiple Contributors, Wikipedia is licensed under CC BY-SA 3.0

14-C.4: Orchestration is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

Module 15: Linux Installation

15-A.1: Linux Installation
15-A.2: Installation: Methods and Preparation

This page titled Module 15: Linux Installation is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

15-A.1: Linux Installation

Objectives of this Module

In this lesson, you will:

- Learn the steps to prepare for installing the Linux operating system.
- Perform the actual installation of Linux.

First Things

Even though Linux is a single operating system, the various distributions have slight variations that a user needs to consider when determining which distro to add to their data center. Most of the distributions have a page on their web site that lists the hardware requirements for all active versions. It is a necessity to take the time and verify that the potential hardware platform is compatible with the potential distribution of Linux.

As the popularity of Linux grows, it is more and more likely that that each distribution will support a wider range of hardware, but still, it is a good idea to verify all the pieces. For instance, CentOS has a list of specifications for the currently supported CentOS versions.

CentOS Product Specifications

This page contains an overview on the capabilities and limits of CentOS.

End of Lifetime (EOL) Dates			
	CentOS-6	**CentOS-7**	**CentOS-8**
Full Updates[1]	May 10th, 2017	Q4 2020	May 2024
Maintenance Updates[2]	November 30th, 2020	June 30th, 2024	May 31st, 2029
Architecture			
	CentOS-6	**CentOS-7**	**CentOS-8**
Latest Release	6.10	7 (2009)	8 (2004)
Supports i386 Systems (32-bit Pentium, AMD, Via)	Yes[3]	Yes (1503)	No
Supports x86_64 systems (AMD64/EM64T)	Yes	Yes	Yes
Supports PowerPC8le systems (IBM Power, Mac)	No	Yes (AltArch)	Yes (AltArch)
Supports IA-32	No	Yes (AltArch)	Yes (AltArch)
Supports ARMv7hl systems	No	Yes (AltArch)	Yes (AltArch)
Supports AArch64 (arm64) systems	No	Yes (AltArch)	Yes (AltArch)
CPU / Memory / Filesystem limits (Tested/possible)			
	CentOS-6	**CentOS-7**	**CentOS-8**
Maximum logical CPUs[4]			
x86	32	n/a	n/a
x86_64	448/4096	768/5120	768/8192
Maximum memory			
x86	16GB	n/a	n/a
x86_64	12TB/64TB	12TB/64TB	24TB/64TB
ARM	n/a	n/a	1.5TB/254TB

Figure 15 – A.1.1: "CentOS Product Specifications" by Christoph Galuschka is in the Public Domain, CC0

Notice that the newer version, CentOS 8, does NOT support the older i386 processors. The supported memory is also different for the newer versions. All of these are things that need to be verified before a decision is made about the distribution and the version.

Most Linux distributions have pages like this on their web sites - use them to verify your needs.

Partition: How Should it Look?

Disk partitioning or disk slicing is the creation of one or more regions on secondary storage, so that each region can be managed separately.
An entire disk may be allocated to a single partition, or multiple ones may be allocated for cases such as dual-booting, maintaining a swap partition, or to logically separate data such as audio and video files. The partitioning scheme is stored in a partition table such as Master Boot Record (MBR) or GUID Partition Table (GPT).

Partition tables are created and modified using one of many partitioning tools. The tools available for Arch Linux are listed in the #Partitioning tools section.

Partitions usually contain a file system directly which is accomplished by creating a file system on (a.k.a. formatting) the partition. Alternatively, partitions can contain LVM, block device encryption or RAID, which ultimately provide device files on which a file system can be placed (or the devices can be stacked further).

Any block device (e.g., disk, partition, LUKS device, LVM logical volume or RAID array) that directly contains a mountable file system is called a volume.

Disk partitioning in the past was much more critical. The disk drives were not that big, and as an administrator there was a desire to use the disk space as efficiently as possible. A great deal of planning went into what partitions were going to be created, and how much space to allocate to each partition.

With the larger drives now, there is less concern about squeezing every last usable bit out of the partitioning scheme. However, as an administrator, you should still be aware of the efficient partitioning of the disk drives.

Depending on the distribution the size may be different. The following is a sample from Fedora Linux.

/boot - 500 MB

> This partition contains the operating system kernel, which allows Fedora to boot. It also contains other files used during the bootstrap process. Due to the limitations of most firmware, creating a separate, small standard partition for this directory is recommended. In most scenarios, a 500 MB **/boot** partition is adequate.

/ (root) - 10 GB

> This is where the root directory is located. The root directory is the top level of the directory structure. By default, all files are written to this partition unless a different partition is mounted in the path being written to (for example, **/boot** or **/home**). If you follow the recommended scheme described in this section, this will be the partition where most software packages will be installed.
>
> For a minimal installation, a 5 GB root partition will be sufficient. However, for most common installations which include extra packages and a graphical user interface, the root partition should be at least 10 GB; with 20 GB being sufficient for most common use cases.

/home - at least 10 GB

> To store user data separately from system data, create a dedicated mount point for the **/home** directory. This partition should be sized based on the amount of data that will be stored locally, number of users, and so on. This will allow you to upgrade or reinstall Fedora without erasing user data files. During the installation, a separate **/home** partition will be created if there are 50 GB or more of free space for your Fedora installation.
>
> When using Fedora as a workstation for normal use with a graphical environment, this mount point should have the most disk space assigned to it, as it will likely hold the most data (user settings, images, videos, etc).

Other partitions are allowed. In some scenarios you might want to put /usr and /var in their own partitions as well. In the above example - where / is 10 GB, that is because it includes /usr and /var, which are both full of executable files, libraries, and log files, all of which take up a tremendous amount of space.

One example system where this is done you can see: /, /usr, /home and /var are all on different partitions. As the system administrator you decide how to divide up the disk space.

```
/dev/sda1        8940016   5706604   2756228  68% /
/dev/sda2       80503616  11995828  64395404  16% /usr
/dev/sda6      310238080 127966488 166489348  44% /home
/dev/sda5       80503616  24267152  52124080  32% /var
```

SWAP Partition

Swap partitions support virtual memory; data is written to them when there is not enough RAM to store the data your system is processing. This partition's size is a function of system memory workload, not total system memory, and therefore is not equal to the total system memory size. Therefore, it is important to analyze what applications a system will be running and the load those applications will serve in order to determine the system memory workload. Application providers and developers should be able to provide some guidance.

When the system runs out of swap space, the kernel terminates processes as the system RAM memory is exhausted. Configuring too much swap space results in storage devices being allocated but idle, and is a poor use of resources. Too much swap space can also hide memory leaks. The maximum size for a swap partition and other additional information can be found in the **mkswap(8)** man page.

The table below provides the recommended size of a swap partition depending on the amount of RAM in your system and whether you want sufficient memory for your system to hibernate. If you let the installation program partition your system automatically, the swap partition size will be established using these guidelines. Automatic partitioning setup assumes hibernation is not in use, and the maximum size of the swap partition is limited to 10% of the total size of the hard drive. If you want to set up enough swap space to allow for hibernation, or if you want to set the swap partition size to more than 10% of the system's storage space, you must edit the partitioning layout manually.

Amount of RAM in the system	Recommended swap space	Recommended swap space if allowing for hibernation
less than 2 GB	2 times the amount of RAM	3 times the amount of RAM
2 GB - 8 GB	Equal to the amount of RAM	2 times the amount of RAM
8 GB - 64 GB	0.5 times the amount of RAM	1.5 times the amount of RAM
more than 64 GB	workload dependent	hibernation not recommended

At the border between each range listed above (for example, a system with 2 GB, 8 GB, or 64 GB of system RAM), discretion can be exercised with regard to chosen swap space and hibernation support. If your system resources allow for it, increasing the swap space may lead to better performance.

Adapted from:

"CentOS Product Specifications" by Christoph Galuschka is in the Public Domain, CC0

"5.4.10.6. Recommended Partitioning Scheme" by Multiple Contributors, Fedora Documentation is licensed under CC BY-SA 3.0

This page titled 15-A.1: Linux Installation is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

15-A.2: Installation: Methods and Preparation

Installation: Options

In order to get Linux installed on a system there are a couple of different options. Again, most distributions have some sort of method to accomplish these options, and so it would be necessary to consult the distro's web site.

Type of Boot	Description
PXE boot	The Preboot eXecution Environment (PXE - *pixie*) specification describes a standardized client-server environment that boots a software assembly, retrieved from a network, on PXE-enabled clients. On the client side it requires only a PXE-capable network interface controller (NIC), and uses a small set of industry-standard network protocols such as DHCP and TFTP. The concept behind the PXE originated in the early days of protocols like BOOTP/DHCP/TFTP, and as of 2015 it forms part of the Unified Extensible Firmware Interface (UEFI) standard. In modern data centers, PXE is the most frequent choice for operating system booting, installation and deployment.
Boot from ISO	It is possible to download an ISO image of a Linux distribution onto an empty partition on the local disk, and configure GRUB2 to list that ISO image on the GRUB2 menu. This requires the editing of the GRUB files to ensure that the option shows on the boot menu.
Boot from removable device	Linux can be installed from a USB flash drive. This may be necessary for most new portable computers without DVD drives and is handy for others because a USB flash drive is so convenient. Also, you can configure Linux on the USB flash drive to save changes you make, unlike a read-only CD/DVD disk.

So, there are a lot of pieces to consider PRIOR to actually installing Linux. Do not skip or short circuit any of the steps. Work through all the questions and make sure that you know what it is you are doing and that you have all the parts you need. Especially the first time you attempt an installation, pick an easy straight forward installation. As you gain experience you will gain confidence to take on more of the complex options.

Here's a road map for the steps you will take during the installation process:

1. Back up all existing data on the hard disk where the installation is planned.
2. Make sure that you have all the information/documentation about your computer before starting the installation.
3. Locate and/or download the installer software along with any drivers or firmware files the machine requires.
4. Set up boot media I.E. CDs/DVDs/USB sticks or provide a network boot infrastructure from which the installer can be booted.
5. Boot the installer software.
6. Select the system's language setting.

7. Activate the Ethernet network connection, if performing a network installation.

8. If required, resize existing partitions on the target disk drive as determined prior to the install.

9. Create and mount the partitions on which Linux will be installed.

10. Monitor the automatic download/install/setup of the base system.

11. Install a boot loader which can start up Linux on the system.

12. Load the newly installed Linux system.

This page titled 15-A.2: Installation: Methods and Preparation is shared under a CC BY-NC 4.0 license and was authored, remixed, and/or curated by Patrick McClanahan.

CHAPTER OVERVIEW

Ancillary Materials

Lab Exercises

Powerpoint Presentations

Ancillary Materials is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

Lab Exercises

This page is a draft and is under active development.

Template:HideTOC

Lab Exercises is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

Powerpoint Presentations

This page is a draft and is under active development.

Template:HideTOC

Powerpoint Presentations is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

Index

B

Bash
- 01-B.14: Doing bash

C

command line interface
- 01-B.14: Doing bash

S

sticky bit
- 03-C.4.2: Special Permission Types - The Sticky Bit

Glossary

Sample Word 1 | Sample Definition 1

Detailed Licensing

Overview

Title: Linux - The Penguin Marches On (McClanahan)

Webpages: 255

Applicable Restrictions: Noncommercial

All licenses found:

- CC BY-SA 4.0: 70.2% (179 pages)
- CC BY-NC 4.0: 23.1% (59 pages)
- Undeclared: 5.9% (15 pages)
- CC BY-NC-SA 4.0: 0.8% (2 pages)

By Page

- Linux - The Penguin Marches On (McClanahan) - *CC BY-NC 4.0*
  - Front Matter - *Undeclared*
    - Title Page - *Undeclared*
    - InfoPage - *Undeclared*
    - Table of Contents - *Undeclared*
    - Licensing - *Undeclared*
    - ASCCC OERI - *Undeclared*
  - Module 01: Back to the Basics - *CC BY-NC 4.0*
    - 01-A.1: Introduction to Linux - *CC BY-NC 4.0*
    - 01-A.2: Open Source Software - *CC BY-SA 4.0*
    - 01-A.3: Free Software vs Open Source - *CC BY-NC 4.0*
    - 01-A.4: Different FOSS licensing... - *CC BY-NC 4.0*
    - 01-A.5: The GNU Project - *CC BY-NC 4.0*
    - 01-A.6: Unix - What's the Connection? - *CC BY-NC 4.0*
    - 01-A.7: Finally - Linux - *CC BY-NC 4.0*
    - 01-A.8: Advantages and Disadvantages of Linux - *CC BY-NC 4.0*
    - 01-A.9: Distro List - *CC BY-NC 4.0*
    - 01-A.10: How is Linux Used? - *CC BY-SA 4.0*
    - 01-B.11: How the Shell and CLI Work - *CC BY-SA 4.0*
    - 01-B.12: CLI - The Good and the Bad - *CC BY-NC 4.0*
    - 01-B.13: Shells: What and Why - *CC BY-SA 4.0*
    - 01-B.14: Doing bash - *CC BY-NC 4.0*
    - 01-B.15: Bash Command How To - *CC BY-SA 4.0*
    - 01-B.16: Let's Look at the Basics - *CC BY-SA 4.0*
    - 01-B.17: File Basics: More Later - *CC BY-NC 4.0*
    - 01-B.18: Who is superuser? - *CC BY-NC 4.0*
    - 01-B.19: Leftovers, What Else Do We Need? - *CC BY-SA 4.0*
    - 01-C.20: Help! Where can I find help in Linux? - *CC BY-NC 4.0*
    - 01-C.21: Linux man pages - *CC BY-SA 4.0*
    - 01-C.22: Man Page Format - *CC BY-SA 4.0*
    - 01-C.23: Additional Linux System Help Choices - *CC BY-NC 4.0*
    - 01-C.24: Sources of Online Linux Documentation - *CC BY-NC 4.0*
    - 01-C.25: Asynchronous Linux Help Resources - *CC BY-SA 4.0*
  - Module 02: User & Group Administration - *CC BY-NC 4.0*
    - 02-A.1: Users, and How to Acquire Superuser Power - *CC BY-NC 4.0*
    - 02-A.2: Who Is the Superuser? - *CC BY-SA 4.0*
    - 02-A.3: su: The Command - *CC BY-NC 4.0*
    - 02-A.4: Other Options for Administrative Privileges - *CC BY-SA 4.0*
    - 02-B.5: Users: Create, Modify, and Delete - *CC BY-NC 4.0*
    - 02-B.6: Users: Create, Modify, and Delete (cont'd) - *CC BY-SA 4.0*
    - 02-B.7: Users: Create, Modify and Delete (continued) - *CC BY-SA 4.0*
    - 02-B.8: Users: Create, Modify, and Delete - passwd Command - *CC BY-NC 4.0*
    - 02-B.9: Users: Create, Modify, and Delete - etc passwd and etc shadow - *CC BY-NC 4.0*
    - 02-B.10: Users: Create, Modify, and Delete - Edit /etc/passwd File - *Undeclared*
    - 02-C.11: Groups: Create, Modify, and Delete - *CC BY-NC 4.0*

- 02-C.12: Groups: Create, Modify, and Delete - Modify Group - *Undeclared*
- 02-D.13: User and Group Queries - *CC BY-SA 4.0*
- 02-E.14: User Account Profile Configuration - *CC BY-NC 4.0*
- 02-E.15: Example: Bash RC file - *CC BY-NC 4.0*
- 02-E.16: System Wide User Profiles - *CC BY-NC 4.0*
  - 02-E.16.1: System Wide User Profiles: /etc/profile - *CC BY-NC-SA 4.0*
  - 02-E.16.2: System Wide User Profiles: The /etc/profile.d Directory - *CC BY-NC-SA 4.0*
  - 02-E.16.3: System Wide User Profiles: The /etc/bashrc File - *CC BY-NC 4.0*

- Module 03: Permission and Ownership Management - *CC BY-NC 4.0*
  - 03-A.1: Modify Permissions: File and Directory - *CC BY-NC 4.0*
  - 03-A.2: Modify Permissions: File and Directory (continued) - *CC BY-SA 4.0*
  - 03-B.3: Modify Ownership of Files and Directories - *Undeclared*
  - 03-C.4: Special Permission Types - The setuid Bit - *CC BY-SA 4.0*
    - 03-C.4.1: Special Permission Types - The setgid Bit - *CC BY-SA 4.0*
    - 03-C.4.2: Special Permission Types - The Sticky Bit - *CC BY-SA 4.0*
  - 03-C.5: File Attributes - *CC BY-SA 4.0*
  - 03-D.6: Permission Issues and How to Troubleshoot - *CC BY-SA 4.0*
- Module 04: Managing Linux Storage - *CC BY-NC 4.0*
  - 04-A.1: Storage Devices - *CC BY-SA 4.0*
  - 04-A.2: Linux File Systems - *CC BY-SA 4.0*
  - 04-A.3: Inodes - *CC BY-SA 4.0*
  - 04-A.4 Journaling File Systems - *CC BY-SA 4.0*
  - 04-A.5: Linux Virtual File System - *CC BY-SA 4.0*
  - 04-A.6: Partitions and Labels - *CC BY-SA 4.0*
    - 04-A.6.1: Partitions and Labels continued - *CC BY-SA 4.0*
  - 04-A.7: Disk Setup Process - fdisk - *CC BY-SA 4.0*
    - 04-A.7.1: Disk Setup Process - parted - *CC BY-SA 4.0*
    - 04-A.7.2: Disk Setup Process - mkfs & mount - *CC BY-SA 4.0*
  - 04-A.8: The dev Directory - *CC BY-NC 4.0*
  - 04-B.9: Logical Volume Management - *CC BY-SA 4.0*
    - 04-B.9.1: LVM and other stuff - *CC BY-SA 4.0*
    - 04-B.9.2: LVM Tools - *CC BY-SA 4.0*
  - 04-C.10: Mounting Linux File Systems - *CC BY-SA 4.0*
  - 04-D.11: File System Management - *CC BY-NC 4.0*
    - 04-D.11.1: File System Management - ext Filesystem Tools - *CC BY-SA 4.0*
    - 04-D.11.2: File System Management - xfs Tools - *CC BY-NC 4.0*
  - 04-E.12: Linux Directory Structure - *CC BY-SA 4.0*
    - 04-E.12.1: Linux Directory Structure - Hierarchy - *CC BY-SA 4.0*
    - 04-E.12.2: Linux Directory Structure - Home Directory - *CC BY-SA 4.0*
    - 04-E.12.3: Linux Directory Structure - Paths - *CC BY-SA 4.0*
  - 04-F.13: Storage Issues - Find and Fix - *CC BY-SA 4.0*
    - 04-F.13.1: Storage Issues - IO Scheduling - *CC BY-SA 4.0*
    - 04-F.13.2: Storage Issues - More IO Scheduling - *CC BY-SA 4.0*
    - 04-F.13.3: Storage Issues - iostat, ioping - *CC BY-SA 4.0*
    - 04-F.13.4: Storage Issues - Quota Management - *CC BY-SA 4.0*
- Module 05: File and Directory Management - *CC BY-NC 4.0*
  - 05-A.1: Creating and Editing Text files - *CC BY-NC 4.0*
  - 05-A.2: Creating and Editing Text files - Vim Editor - *CC BY-SA 4.0*
    - 05-A.2.1: Vim - Insert Mode - *CC BY-SA 4.0*
    - 05-A.2.2: Vim - Visual Mode - *CC BY-SA 4.0*
    - 05-A.2.3: Vim - Command-Line Mode - *CC BY-SA 4.0*
  - 05-A.3: Nano Editor - *CC BY-NC 4.0*
  - 05-A.4: Gedit and More - *CC BY-SA 4.0*
  - 05-B.5: Searching for Files on Linux - *CC BY-SA 4.0*
    - 05-B.5.1: Searching for Files on Linux - find Command - *CC BY-SA 4.0*
    - 05-B.5.2: Searching for Files on Linux - which Command - *CC BY-SA 4.0*
    - 05-B.5.3: Searching for Files on Linux - whereis Command - *CC BY-SA 4.0*
  - 05-C.6: Working with Files and Directories - cat Command - *CC BY-NC 4.0*
    - 05-C.6.1: Working with Files and Directories - head/tail Commands - *CC BY-SA 4.0*

        - 05-C.6.2: Working with Files and Directories - less/more Command - *CC BY-SA 4.0*
        - 05-C.6.3: Working with Files and Directories - cp/mv Command - *CC BY-SA 4.0*
        - 05-C.6.4: Working with Files and Directories - rm/touch Command - *CC BY-SA 4.0*
        - 05-C.6.5: Working with Files and Directories - ls/rmidir/mkdir Command - *CC BY-SA 4.0*
    - 05-D.7: Handling Text Files - echo/printf - *CC BY-SA 4.0*
        - 05-D.7.1: Handling Text Files - tr/wc Command - *CC BY-SA 4.0*
        - 05-D.7.2: Handling Text Files - sort/diff Commands - *CC BY-SA 4.0*
        - 05-D.7.3: Handling Text Files - cut/paste - *CC BY-SA 4.0*
        - 05-D.7.4: Handling Text Files - grep Command - *CC BY-SA 4.0*
        - 05-D.7.5: Handling Text Files - awk Command - *CC BY-SA 4.0*
        - 05-D.7.6: Handling Text Files - sed Command - *CC BY-SA 4.0*
        - 05-D.7.7: Handling Text Files - ln Command - *CC BY-SA 4.0*
    - 05-E.8: File Output Manipulation - *CC BY-SA 4.0*
        - 05-E.8.1: File Output Manipulation - pipe - *CC BY-SA 4.0*
        - 05-E.8.2: File Output Manipulation - xargs - *CC BY-SA 4.0*
        - 05-E.8.3: File Output Manipulation - tee & /dev/null - *CC BY-SA 4.0*
- Module 06: Kernel Module Management - *CC BY-NC 4.0*
    - 06-A.1: Examine the Linux Kernel - *CC BY-SA 4.0*
    - 06-A.2: Linux Kernel - *CC BY-SA 4.0*
    - 06-A.3: Linux Kernel Subsystem - *CC BY-SA 4.0*
    - 06-B.4: Kernel Modules - *CC BY-SA 4.0*
    - 06-B.5: Kernel Module Management - lsmod/modinfo Command - *CC BY-SA 4.0*
        - 06-B.5.1: Kernel Module Management - insmod/rmmod Command - *CC BY-SA 4.0*
        - 06-B.5.2: Kernel Module Management - modprobe/depmod Command - *CC BY-SA 4.0*
    - 06-B.6: Kernel Modules - Configuration - *CC BY-SA 4.0*
    - 06-B.7: Kernel Modules - Kernel Parameters - *CC BY-SA 4.0*
    - 06-C.8: Kernel Module Monitoring - *CC BY-SA 4.0*
    - 06-C.9: The dmesg Command - *CC BY-SA 4.0*
- Module 07: Linux Boot Process - *CC BY-NC 4.0*
    - 07-A.1: Configuring Boot Components - *CC BY-SA 4.0*
    - 07-A.2: Configuring Boot Components (continued) - *CC BY-SA 4.0*
    - 07-A.3: Configuring initrd - *CC BY-SA 4.0*
    - 07-A.4: Boot Process - *CC BY-SA 4.0*
    - 07-B.5: GRUB2 - Installation - *CC BY-SA 4.0*
    - 07-B.6: GRUB2 - Configuration - *CC BY-SA 4.0*
- Module 08: How to Manage System Components - *CC BY-NC 4.0*
    - 08-A.1: Localization Options - Configuration - *CC BY-SA 4.0*
    - 08-A.2: Localization - date/timdatectl Command - *CC BY-SA 4.0*
    - 08-A.3: Localization - localectl Command - *CC BY-SA 4.0*
    - 08-B.4: GUI Configuration - *CC BY-SA 4.0*
    - 08-B.5: GUI Desktop Environments - *CC BY-SA 4.0*
    - 08-B.6: Remote Desktop Access - *CC BY-SA 4.0*
    - 08-B.7: Remote Console - SSH/X Forwarding - *CC BY-SA 4.0*
    - 08-C.8: Managing Linux Services - *CC BY-SA 4.0*
        - 08-C.8.1: Managing Linux Services - systemctl - *CC BY-SA 4.0*
        - 08-C.8.2: Managing Linux Services - hostnamectl - *CC BY-SA 4.0*
        - 08-C.8.3: Managing Linux Services - SysVinit - *CC BY-SA 4.0*
        - 08-C.8.4: Managing Linux Services - inittab & init.d - *CC BY-SA 4.0*
    - 08-D.9: Process Troubleshooting - *CC BY-SA 4.0*
        - 08-D.9.1: Process Troubleshooting - ps command - *CC BY-SA 4.0*
        - 08-D.9.2: Process Troubleshooting - top command - *CC BY-SA 4.0*
        - 08-D.9.3: Process Troubleshooting - lsof Command - *CC BY-SA 4.0*
        - 08-D.9.4: Process Troubleshooting - Process Priorities - *CC BY-SA 4.0*
        - 08-D.9.5: Process Troubleshooting - Background / Foreground Processes - *CC BY-SA 4.0*
        - 08-D.9.6: Process Troubleshooting - nohup and kill Commands - *CC BY-SA 4.0*
    - 08-E.10: CPU and Memory Troubleshooting - *CC BY-SA 4.0*
        - 08-E.10.1: CPU and Memory Troubleshooting - sar & Memory Issues - *CC BY-SA 4.0*

        - 08-E.10.2: CPU and Memory Troubleshooting - free / vmstat Commands - *CC BY-SA 4.0*
        - 08-E.10.3: CPU and Memory Troubleshooting - OOM & swap - *CC BY-SA 4.0*
    - Module 09: Managing Linux Devices - *CC BY-NC 4.0*
        - 09-A.1: Supported Linux Devices - *CC BY-SA 4.0*
            - 09-A.1.1: Supported Linux Devices (cont'd) - *CC BY-SA 4.0*
            - 09-A.1.2: Supported Linux Devices (continued) - *CC BY-SA 4.0*
        - 09-B.2: Configuring Linux Devices - *CC BY-SA 4.0*
            - 09-B.2.1: Configuring Linux Devices - Printing - *CC BY-SA 4.0*
        - 09-C.3: Monitoring Devices in Linux - *CC BY-SA 4.0*
        - 09-D.4: Troubleshooting: Hardware Issues - *CC BY-SA 4.0*
    - Module 10: Network Management - *CC BY-NC 4.0*
        - 10-A.1: TCP/IP Fundamentals - *Undeclared*
            - 10-A.1.1: TCP/IP Fundamentals - Identities - *CC BY-SA 4.0*
            - 10-A.1.2: TCP/IP Fundamentals - Addressing - *CC BY-SA 4.0*
            - 10-A.1.3: TCP/IP Fundamentals - Ports and Segments - *CC BY-SA 4.0*
        - 10-B.2: Server Roles in Linux - *CC BY-SA 4.0*
            - 10-B.2-3: Server Roles in Linux (continued) - *CC BY-SA 4.0*
            - 10-B.2.1: Server Roles in Linux - Continued - *CC BY-SA 4.0*
            - 10-B.2.4: Server Roles in Linux (continued) - *CC BY-SA 4.0*
        - 10-C.3: Network Connection - *CC BY-SA 4.0*
        - 10-D.4: Static VS Dynamic IP Addresses - *CC BY-SA 4.0*
        - 10-D.5: DHCP Configuration - *CC BY-SA 4.0*
        - 10-D.6: Name Resolution - *CC BY-SA 4.0*
        - 10-E.7: Cloud Configuration - *CC BY-SA 4.0*
        - 10-E.8: The Cloud and Virtualization - *CC BY-SA 4.0*
        - 10-E.9: Cloud Bootstrapping and Storage - *CC BY-SA 4.0*
        - 10-E.10: Configuring Networks and Virtualization Tools - *CC BY-NC 4.0*
        - 10-F.11: How to Troubleshoot Networking Problems - *CC BY-SA 4.0*
            - 10-F.11.1: Network Troubleshooting Commands - netstat / ss - *CC BY-SA 4.0*
            - 10-F.11.2: Network Troubleshooting Commands - dig / nslookup / host - *CC BY-SA 4.0*
            - 10-F.11.3: Network Troubleshooting Commands - ip / route / nmap - *CC BY-SA 4.0*
            - 10-F.11.4: Network Troubleshooting Commands - wireshark - *CC BY-SA 4.0*
            - 10-F.11.5: Network Troubleshooting Commands - iftop / iperf / mtr - *CC BY-SA 4.0*
            - 10-F.11.6: Network Troubleshooting Commands - arp / whois - *CC BY-SA 4.0*
    - Module 11: Package Management - *CC BY-NC 4.0*
        - 11-A.1: Who are the Package Managers - *CC BY-SA 4.0*
        - 11-B.2: rpm / yum How To - *CC BY-SA 4.0*
        - 11-C.3: dpkg / apt How To - *CC BY-SA 4.0*
        - 11-D.4: Repository Configuration - *CC BY-SA 4.0*
        - 11-E.5: Other Software Sources - *CC BY-SA 4.0*
        - 11-F.6: Software Builds from Source - *CC BY-SA 4.0*
        - 11-G.7: Software Dependency Troubleshooting - *CC BY-NC 4.0*
    - Module 12: Linux Systems Security - *CC BY-NC 4.0*
        - 12-A.1: Best Cybersecurity Practices - *CC BY-SA 4.0*
            - 12-A.1.1: Best Cybersecurity Practices - Continued - *CC BY-SA 4.0*
        - 12-A.2: chroot Jail - *CC BY-SA 4.0*
        - 12-A.3: Encryption in Linux - *CC BY-SA 4.0*
        - 12-A.4: Security Best Practices - *CC BY-NC 4.0*
        - 12-B.5: Identity and Access Management - *CC BY-SA 4.0*
        - 12-B.6: Pluggable Authentication Modules - *CC BY-NC 4.0*
        - 12-B.7: TTY Security & PKI - *CC BY-SA 4.0*
        - 12-B.8: OpenSSL / VPN - *CC BY-SA 4.0*
        - 12-B.9: Authentication Issues - *CC BY-NC 4.0*
        - 12-C.10: SELinux Configuration - *CC BY-SA 4.0*
        - 12-C.11: AppArmor - *CC BY-SA 4.0*
        - 12-D.12: Firewall Configuration in Linux - *CC BY-SA 4.0*
        - 12-D.13: Access Control Lists / iptables - *CC BY-SA 4.0*
        - 12-D.14: UFW & firewalld - *CC BY-SA 4.0*
        - 12-D.15: Netfilter / IP Forwarding - *CC BY-SA 4.0*
        - 12-E.16: Linux Logging - *CC BY-SA 4.0*
        - 12-E.17: journalctl & last Commands - *CC BY-NC 4.0*
        - 12-F.18: Backup and Restore in Linux - *CC BY-SA 4.0*
        - 12-F.19: tar / cpio / dd - *CC BY-SA 4.0*
        - 12-F.20: mirrorvg / rsync / scp - *CC BY-SA 4.0*
        - 12-F.21: File Compression - *CC BY-SA 4.0*

- 12-F.22: Integrity Check and Checksums - *CC BY-SA 4.0*

- Module 13: Working with Bash Scripts - *CC BY-NC 4.0*
  - 13-A.1: How to Customize the Bash Shell - *CC BY-SA 4.0*
  - 13-A.2: env / alias / PATH / time - *CC BY-SA 4.0*
  - 13-B.3: Fundamentals of Linux Scripts - *CC BY-SA 4.0*
  - 13-B.4: Shell Operators - *CC BY-SA 4.0*
  - 13-B.5: Strings / Arrays / Functions - *CC BY-SA 4.0*
  - 13-C.6: Writing and Running Linux Scripts - *CC BY-SA 4.0*
  - 13-C.7: Shell Expansion / Variable Substitution / Command Substitution - *CC BY-SA 4.0*
  - 13-C.8: Positional Parameters / exec Command / source Command - *CC BY-SA 4.0*
  - 13-C.9: Script Permissions / Script Extensions - *CC BY-NC 4.0*
  - 13-D.10: Shell Control Statements - *CC BY-SA 4.0*
  - 13-D.11: Script Loops - *CC BY-SA 4.0*
- Module 14: Task Automation - *CC BY-NC 4.0*
  - 14-A.1: Job Scheduling - at Command - *CC BY-SA 4.0*
  - 14-A.2: Cron - *CC BY-SA 4.0*
  - 14-B.3: Git: Version Control - *CC BY-SA 4.0*
  - 14-C.4: Orchestration - *CC BY-SA 4.0*
- Module 15: Linux Installation - *CC BY-NC 4.0*
  - 15-A.1: Linux Installation - *CC BY-SA 4.0*
  - 15-A.2: Installation: Methods and Preparation - *CC BY-NC 4.0*
- Ancillary Materials - *Undeclared*
  - Lab Exercises - *Undeclared*
  - Powerpoint Presentations - *Undeclared*
  - Test Banks - *Undeclared*
- Back Matter - *CC BY-NC 4.0*
  - Index - *CC BY-NC 4.0*
  - Glossary - *CC BY-NC 4.0*
  - Detailed Licensing - *Undeclared*