

# Linux: System Administration

A screenshot of a Linux terminal window. The prompt 'sumcu@ubuntu: ~\$' is displayed in green and blue text. The command 'sudo' is being entered in white text, followed by a vertical cursor bar. The background is black with a grid pattern.

```
sumcu@ubuntu: ~$ sudo
```

# LOGISTICS



## **Class Hours:**

- Instructor will set class start and end times.
- There will be regular breaks in class.



## **Telecommunication:**

- Turn off or set electronic devices to silent (not vibrate)
- Reading or attending to devices can be distracting to other students
- Try to delay until breaks or after class

## **Miscellaneous:**

- Courseware
- Bathroom
- Fire drills

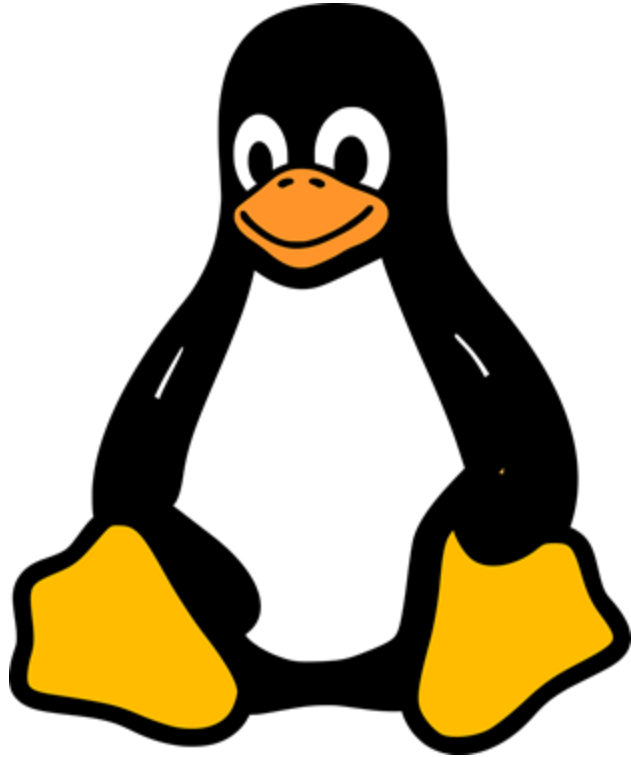
# INTRODUCE YOURSELF

Time to introduce yourself:

- Name
- What is your role in the organization
- Indicate Linux experience



# Course Objectives



- **By the end of this course, you will be able to:**
- **Understand Linux architecture and the command line** — Learn how Linux boots, how processes interact with the kernel, and how to navigate and control the system using shell commands.
- **Manage users, groups, and permissions** — Secure access to files and services by controlling who can read, write, and execute system resources.
- **Handle package management and system updates** — Install, update, and remove software using popular package managers like `yum`, `apt`, and `dnf`.
- **Manage services and troubleshoot systems** — Control background processes with `systemctl`, analyze logs with `journalctl`, and apply diagnostic commands to resolve issues effectively.

# What is Linux System Administration?



Linux Administration is the practice of managing and maintaining Linux systems — ensuring they run efficiently, securely, and reliably.

A Linux administrator is responsible for installing software, managing users and permissions, configuring services, monitoring performance, and troubleshooting issues across servers and networks.

# Linux Operating System

## What is Linux?

- An **open-source operating system** based on **UNIX**.
- Acts as the **bridge between hardware and applications**, managing processes, memory, and files.
- Used everywhere — from servers and smartphones to supercomputers and embedded systems.

## Key Concepts:

- **Kernel:** The core that manages system resources and hardware communication.
- **Shell:** The command-line interface (CLI) that lets users interact with the system.
- **Filesystem:** The structured way Linux organizes and stores data.

## Why Learn Linux?

- Powers over **90% of cloud servers** and **all major DevOps tools**.
- Essential for **system administrators, SREs, and developers**.
- Mastering Linux builds a foundation for automation, security, and cloud engineering.

# Linux Operating System

## Linux Distributions (Distros)

### What is a Distro?

A Linux distribution is a complete operating system built around the Linux kernel, bundled with tools, libraries, and a package manager.

### ◆ Major Families

### 💡 Notes

- Server admins often use **RHEL** or **Ubuntu LTS** for stability.
- Developers and hobbyists prefer **Fedora** or **Arch** for bleeding-edge updates.
- Cloud images (e.g., EC2 AMIs) are usually based on **RHEL**, **Amazon Linux**, or **Ubuntu**.

Family	Example Distros	Package Manager
Debian-based	Ubuntu, Kali, Linux Mint	apt
Red Hat-based	RHEL, CentOS, Fedora, AmazonLinux	dnf / yum
SUSE-based	openSUSE, SUSE Linux Enterprise	zypper
Independent	Arch, Gentoo, Alpine	pacman, emerge, apk

# Linux Boot Process

## 1. BIOS / UEFI Initialization

- Hardware performs **Power-On Self Test (POST)**.
- Loads the **bootloader** from disk or firmware.

## 2. Bootloader (e.g., GRUB)

- Locates and loads the **Linux kernel** into memory.
- Passes control to the kernel with system parameters.

## 3. Kernel Initialization

- Detects hardware and mounts the **root filesystem** (/).
- Starts the **init process** (PID 1) — the first user-space program.

## 4. Init / Systemd

- Initializes system services and targets (networking, logging, etc.).
- Mounts remaining file systems and enables background daemons.

## 5. Login & Shell

- Displays the **login prompt (TTY or GUI)**.
- After authentication, user enters a **shell session** ready for commands.

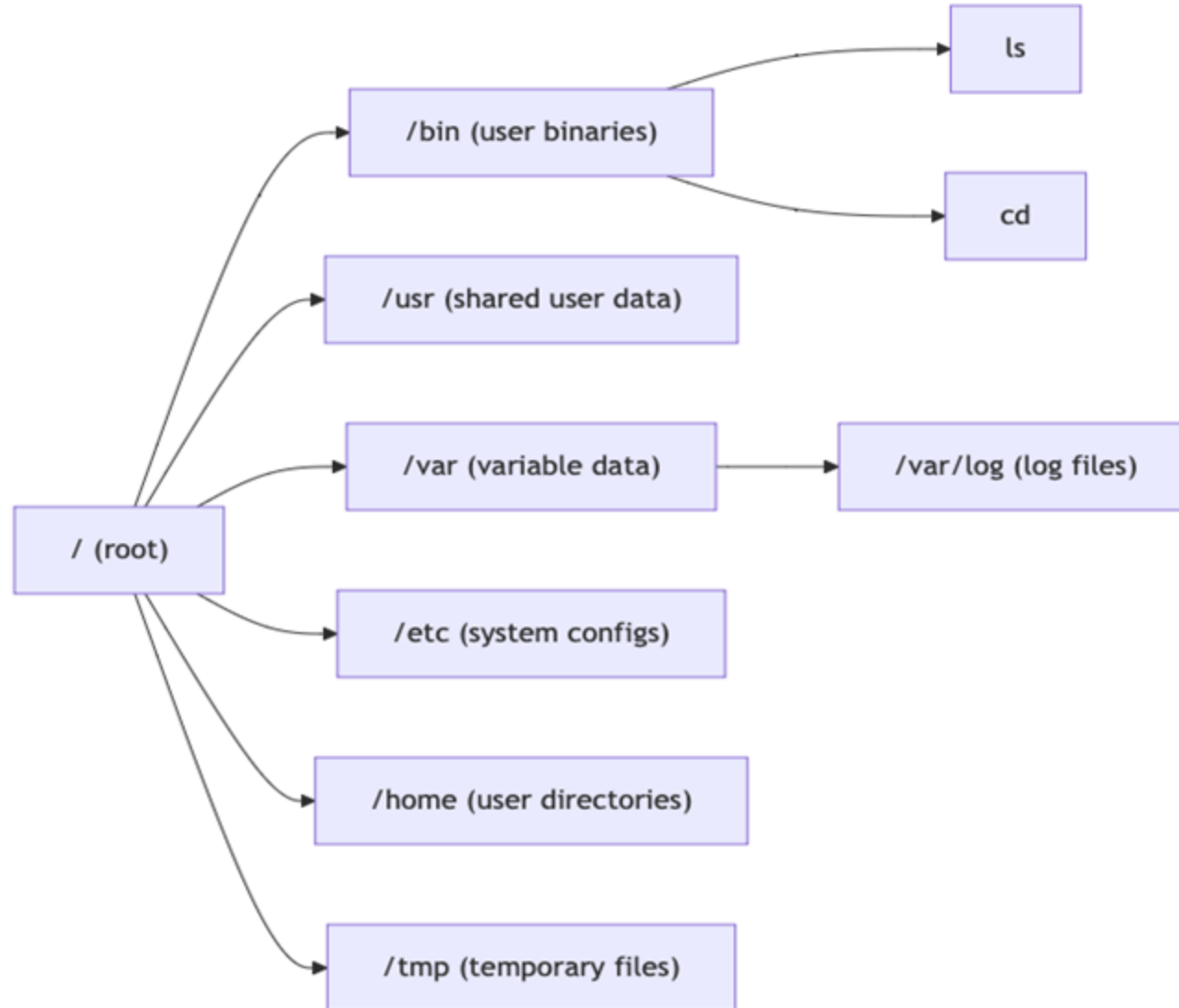


# Linux Basics: File System



- File navigation is core to Linux administration and security. Administrators must be able to locate, inspect, and manage files efficiently — both for configuration and troubleshooting.
- Linux systems typically use ext4 (or sometimes xfs) as their filesystem type.  
You can verify this using commands such as:  
`df -T` (shows filesystem type for each mount) or `mount | grep /dev`
- The `touch` command creates a file or updates its timestamp, while `ls -l` displays file metadata such as permissions, ownership, and modification time — key for security auditing.
- Mastering navigation and file awareness is the first step toward mastering the system itself.

# Linux Basics: File System



# Basic Commands



- Below are some basic commands to get started with
  - pwd, cd, ls, ls -la, history
  - mkdir, touch, cp, mv, rm, diff
  - echo, cat, history
- The **Trick** is to know how to use the manual and other shortcuts
  - man ls
  - Try using /all, enter and n and shift+n. Use g and shift+g.
  - **Always** look up the manual and options when encountering new commands and options.
  - Not all commands have a man page.
  - **Tab Key** is extremely useful for navigating files

# Lab Work



## 💡 Working on Labs Effectively

Don't just copy and paste commands. Take time to **understand each option**, **read the man pages**, and **experiment with alternate approaches**.

Curiosity and exploration builds mastery.

# Absolute vs Relative Path



Consider a house. To get to the kitchen, instructions may be **relative** to your current location (say a room), or from the front door (**absolute**).

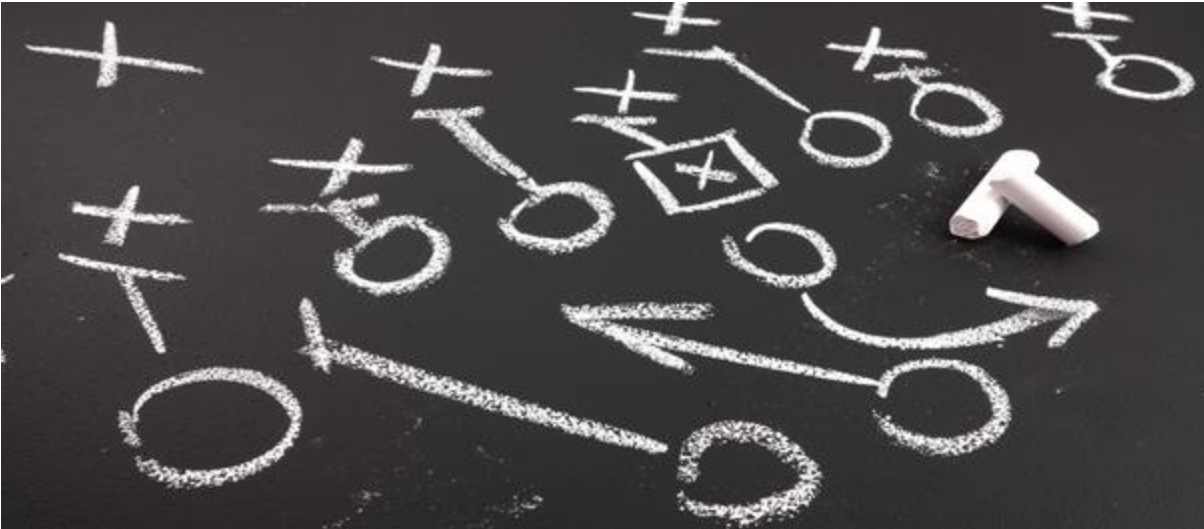
- `cat /home/ubuntu/files/myfile.txt`
  - Always works, no matter working directory.
- `cat files/myfile.txt`
  - Only works if `files/myfile.txt` is in the working directory. If you change your working directory, this command will fail.



# Safety with File Operations

Linux file systems do not come with recycle bins. As an administrator, you should be aware of backup options when running commands that may overwrite files.

- `cp -b` and `mv -b` make backups before overwriting any files
- `rm -i` will prompt a user to confirm file deletion



# Shell Basics

A **shell** is a **command-line interface** between the **user** and the **operating system**.

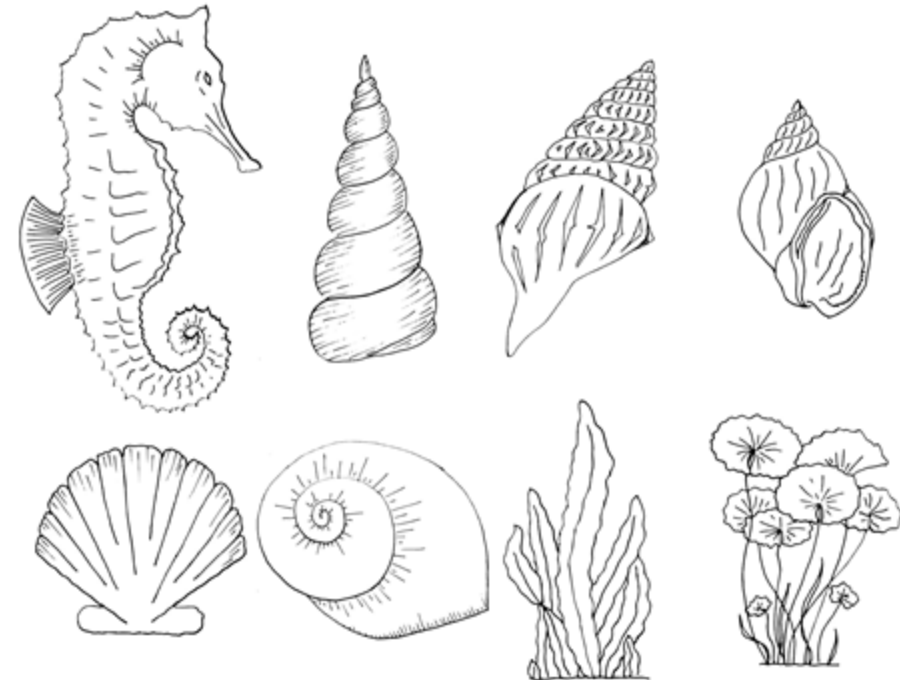
It **interprets commands** you type and **tells the OS what to do**.

Acts as a **wrapper around the kernel** — you don't talk to hardware directly.

Supports **scripting** for automation (**.sh** files).

Common types:

- **Bash** (most common on Linux)
- **Zsh**
- **Fish**
- **PowerShell** (Windows)



# Shell Basics

```
[ec2-user@ip-172-31-1-87 ~]$ cd test  
[ec2-user@ip-172-31-1-87 test]$ pwd  
/home/ec2-user/test  
[ec2-user@ip-172-31-1-87 test]$
```

- Username
- Host Address
- Working Directory
- Prompt ( # for root user ) - After the \$ you enter a command



# Basic Commands

```
[ec2-user@ip-172-31-1-87 ~]$ mkdir test
[ec2-user@ip-172-31-1-87 ~]$ cd test
[ec2-user@ip-172-31-1-87 test]$ echo "Hello World"
Hello World
[ec2-user@ip-172-31-1-87 test]$ echo "Hello World" > file.txt
[ec2-user@ip-172-31-1-87 test]$ echo "Hello Again" > file.txt
[ec2-user@ip-172-31-1-87 test]$ echo "Thank you" >> file.txt
[ec2-user@ip-172-31-1-87 test]$ cat file.txt
Hello Again
Thank you
[ec2-user@ip-172-31-1-87 test]$ cd ..
[ec2-user@ip-172-31-1-87 ~]$ ls -l
total 0
drwxr-xr-x. 2 ec2-user ec2-user 22 Oct 21 16:35 test
[ec2-user@ip-172-31-1-87 ~]$ rm -rf test
[ec2-user@ip-172-31-1-87 ~]$ ls -l
total 0
[ec2-user@ip-172-31-1-87 ~]$
```

# Hidden Files

## The Home Directory

- Your home folder may look **empty**, but `ls -a` reveals **hidden files** (those starting with `.`).
- These files aren't encrypted or protected — they're simply **tucked away** for convenience.

### Common Hidden Files:

- `.bashrc` → runs each time you open a **new shell**
- `.bash_profile` → runs when you **log in**
- `.ssh/` → stores your **SSH keys** and authorized public keys for remote access

💡 *Hidden files often store configuration and authentication details — handle them with care!*

# Lab 1.0 Connecting to Instance

Estimated Time: 15 Minutes



# Lab 1.1 Linux File System

Estimated Time: 20 Minutes



# POP QUIZ:

Which folder contains System Wide Configurations?

- A. /bin
- B. /var
- C. /etc
- D. /home



# POP QUIZ:

Which folder contains System Wide Configurations?

- A. /bin
- B. /var
- C. **/etc**
- D. /home

"What about the other  
directories?"





# POP QUIZ:

What command is used to rename a file in Linux?

- A. nm
- B. mv
- C. cp
- D. rename



# POP QUIZ:

What command is used to rename a file in Linux?

A. nm

B. mv

C. cp

D. rename

"Can this be dangerous?"





# POP QUIZ:

What option below will display long listing of all files?

- A. ls -l
- B. ls -ap
- C. echo pwd
- D. ls -la



# POP QUIZ:

What option below will display long listing of all files?

- A. ls -l
- B. ls -ap
- C. echo pwd
- D. **ls -la** "What about ls -l -a?"



# Linux: File Attributes

Files come with attributes, such as ownership, permissions, type, size, and last touch time.

**System Administrators must understand these attributes to manage access, ensure security, troubleshoot issues, and maintain proper system organization.**



# Linux: File Permissions

Symbol	Permission	Description	Numeric Value
r	Read	Can view or read file	4
w	Write	Modify or delete file	2
x	Execute	Can execute the file	1
-	No Permission	No access	0

```
[ec2-user@ip-172-31-1-87 ~]$ ls -l
total 0
drwxr-xr-x. 2 ec2-user ec2-user 6 Oct 21 16:48 test-dir
-rw-r--r--. 1 ec2-user ec2-user 0 Oct 21 16:48 test.txt
```

# Linux: File Permissions

```
[ec2-user@ip-172-31-1-87 ~]$ ls -l
total 0
drwxr-xr-x. 2 ec2-user ec2-user 6 Oct 21 16:48 test-dir
-rw-r--r--. 1 ec2-user ec2-user 0 Oct 21 16:48 test.txt
```

- First set of bits in red belong to the **user** (file owner). The file owner is also indicated in red.
- The blue set of bits is for the **group**. The file belongs to the group indicated in blue (defaults group is user who created the file)
- The third set of permissions is **other** users



# Linux: File Permissions

```
[ec2-user@ip-172-31-1-87 ~]$ ls -l
total 0
drwxr-xr-x. 2 ec2-user ec2-user 6 0ct 21 16:48 test-dir
-rw-r--r--. 1 ec2-user ec2-user 0 0ct 21 16:48 test.txt
```

The permissions for test-dir are 755

- $r(4) + w(2) + x(1) = 7$
- $r(4) + x(1) = 5$
- $r(4) + x(1) = 5$

***What are the permissions for test.txt?***

# Linux: File Permissions

```
[ec2-user@ip-172-31-1-87 ~]$ ls -l
total 0
drwxr-xr-x. 2 ec2-user ec2-user 6 Oct 21 16:48 test-dir
-rw-r--r--. 1 ec2-user ec2-user 0 Oct 21 16:48 test.txt
```

- The **first char** is the file type. Empty indicates ordinary file, d for directory
- The second **blue box** indicates the number of links to a file
- The third **orange box** represents the file size estimate
- The last **green box** is a timestamp of when the file was last touched

# Linux: File Types

- **Regular File (-)** – Text, data, or binary programs
- **Directory (d)** – Holds files or other directories
- **Symbolic Link (l)** – Shortcut pointing to another file
- **Character Device (c)** – Hardware sending data one character at a time (e.g., keyboard)
- **Block Device (b)** – Hardware that transfers data in blocks (e.g., disk drives)
- **Named Pipe (p)** – Communication between running processes
- **Socket (s)** – Network or inter-process communication endpoint





# Changing File Permissions

The **chmod** (change file mode bits) command is used to **modify file or directory permissions**.

It can be used in two ways:

- **Numeric (absolute)** — set exact permissions using numbers
  - a. `chmod 755 file.sh`
  - b. Sets permissions to **rw-r-xr-x**
- **Symbolic (relative)** — adjust permissions using letters and operators
  - a. `chmod u-x file.sh`
  - b. Would simply subtract execute to the user of the file **rw-r-xr-x**

💡 *Sysadmins use **chmod** to control who can read, write, or execute files on the system.*



# Linux: File Execution

- To execute a file, it must be executable, and you must have permissions to execute.
- You can add the executable file to a directory in `$PATH` variable if you would like make script globally executable.
- Do not be fooled by `source` command. It does not execute a script, it simply runs the commands in a file line by line in current shell.



# Linux: File Execution

```
[ec2-user@ip-172-31-1-87 ~]$ chmod +x my_script.sh
[ec2-user@ip-172-31-1-87 ~]$ ls -l my_script.sh
-rwxr-xr-x. 1 ec2-user ec2-user 17 Oct 21 17:54 my_script.sh
[ec2-user@ip-172-31-1-87 ~]$ ./my_script.sh
hello world
[ec2-user@ip-172-31-1-87 ~]$ echo $PATH
/home/ec2-user/.local/bin:/home/ec2-user/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
[ec2-user@ip-172-31-1-87 ~]$ mkdir -p bin
[ec2-user@ip-172-31-1-87 ~]$ mv my_script.sh bin/my_script.sh
[ec2-user@ip-172-31-1-87 ~]$ cd test-dir/
[ec2-user@ip-172-31-1-87 test-dir]$ my_script.sh
hello world
[ec2-user@ip-172-31-1-87 test-dir]$
```

# Linux: Links

A **link** is a reference that points to another file or directory, allowing multiple names or paths to access the same data.

## Hard vs Soft Links

- **Hard Link:** Created with `ln file linkname`
  - Points to the **same inode** (actual data).
  - If the original file is deleted, the link **still works**.
  - Essentially two names for the same file.
- **Soft (Symbolic) Link:** Created with `ln -s file linkname`
  - Points to the **file's path**, not its data.
  - If the original file is deleted, the link **breaks**.
  - Works like a **shortcut** or **alias**.

 *Hard links share data; soft links reference the path.*



# Linux: File Stat

The `stat` command displays **detailed information** about a file or directory beyond what `ls -l` shows.

Key fields explained:

- **File:** Name and type (regular file, directory, link, etc.)
- **Size:** File size in bytes
- **Inode:** Unique ID for the file's data on disk
- **Links:** Number of hard links pointing to the inode
- **Access (Permissions):** Shows mode in both symbolic and numeric form
- **UID / GID:** Owner and group IDs
- **Access / Modify / Change:**
  - **Access** → Last read time
  - **Modify** → Last content change
  - **Change** → Last metadata change

💡 `stat` gives a full snapshot of a file's identity and history.

# Linux: File Stat

```
[ec2-user@ip-172-31-1-87 ~]$ stat test.txt
  File: test.txt
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: 10301h/66305d  Inode: 8536241    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ec2-user)   Gid: ( 1000/ec2-user)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2025-10-21 16:48:32.734814761 +0000
Modify: 2025-10-21 16:48:32.734814761 +0000
Change: 2025-10-21 16:48:32.734814761 +0000
 Birth: 2025-10-21 16:48:32.734814761 +0000
[ec2-user@ip-172-31-1-87 ~]$
```

# Lab 1.2 File Details

Estimated Time: 30 Minutes





# POP QUIZ:

Which is the correct permission bits below for 701?

- A. `rwx-----x`
- B. `rwxrwxrwx`
- C. `--x---rwx`
- D. None of the above





# POP QUIZ:

Which is the correct permission bits below for 701?

- A. `rwx-----x`
- B. `rwxrwxrwx`
- C. `--x---rwx`
- D. None of the above

"Will a user who is not the owner, but in the group, be able to execute?"



# POP QUIZ:

Which statement about Linux file links is **true**?

- A. A soft link points directly to the file's inode.
- B. A hard link becomes invalid if the original file is deleted.
- C. A soft link stores the path to the original file.
- D. A hard link can reference files on different file systems.



# POP

**QUIZ:** Which statement about Linux file links is **true**?

"What are links again?"

- A. A soft link points directly to the file's inode.
- B. A hard link becomes invalid if the original file is deleted.
- C. A soft link stores the path to the original file.
- D. A hard link can reference files on different file systems.



# Linux: File Types

- **Regular File (-)** – Text, data, or binary programs
- **Directory (d)** – Holds files or other directories
- **Symbolic Link (l)** – Shortcut pointing to another file
- **Character Device (c)** – Hardware sending data one character at a time (e.g., keyboard)
- **Block Device (b)** – Hardware that transfers data in blocks (e.g., disk drives)
- **Named Pipe (p)** – Communication between running processes
- **Socket (s)** – Network or inter-process communication endpoint





# Linux: File Processing and Compression

It is not enough to manage file attributes. Effective system administrators must also be able to **process, view, and manipulate file contents** efficiently.

This includes using commands like `cat`, `tac`, `head`, `tail`, and `less` to read files, as well as tools to **redirect, filter, and compress data** for storage, analysis, or transfer.



# Linux: File Redirection

## File Redirection

- Controls **where input and output go**
- Sends command results to files instead of the screen

### Common Operators:

- `>` Write (overwrite) output to file
- `>>` Append output to file
- `<` Take input from file
- `2>` Redirect errors
- `&>` Redirect output **and** errors

 *Use redirection to save results, capture logs, or process data automatically.*





# Linux: Reading Files

## Reading Files

- View or scroll through file contents in the terminal

### Common Commands:

- `cat file.txt` → Display file contents
- `tac file.txt` → Display contents **in reverse**
- `less file.txt` → Scroll and search inside file
- `more file.txt` → View file one screen at a time

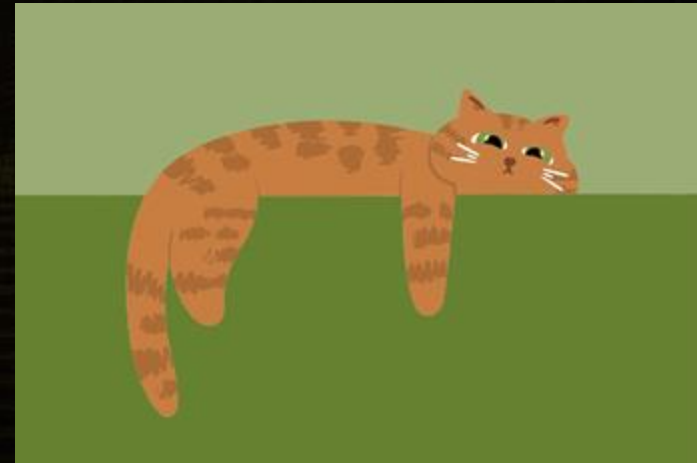
### Searching in `less`:

- Press `/` then type a word → search **forward**
- Press `?` then type a word → search **backward**
- Use `n` / `N` to move to next or previous match

💡 *`less` is best for reading large files interactively.*

# Linux: Reading Files

```
[ec2-user@ip-172-31-1-87 ~]$ echo "Header of my file" > data.txt
[ec2-user@ip-172-31-1-87 ~]$ echo "New header of my file" > data.txt
[ec2-user@ip-172-31-1-87 ~]$ echo "Linux is fun" >> data.txt
[ec2-user@ip-172-31-1-87 ~]$ cat data.txt
New header of my file
Linux is fun
[ec2-user@ip-172-31-1-87 ~]$ tac data.txt
Linux is fun
New header of my file
[ec2-user@ip-172-31-1-87 ~]$ tac data.txt > data.rev
[ec2-user@ip-172-31-1-87 ~]$ cat data.txt
New header of my file
Linux is fun
[ec2-user@ip-172-31-1-87 ~]$ cat data.rev
Linux is fun
New header of my file
[ec2-user@ip-172-31-1-87 ~]$ cat notafire.txt
cat: notafire.txt: No such file or directory
[ec2-user@ip-172-31-1-87 ~]$ cat notafire.txt 2> error.log
[ec2-user@ip-172-31-1-87 ~]$ cat notafire.txt > result.log
cat: notafire.txt: No such file or directory
[ec2-user@ip-172-31-1-87 ~]$ cat result.log
[ec2-user@ip-172-31-1-87 ~]$ cat error.log
cat: notafire.txt: No such file or directory
[ec2-user@ip-172-31-1-87 ~]$
```



# Linux: Reading Files

## Viewing File Sections

Sometimes you only need the **first** or **last** few lines of a file.

Use:

- `head -n` → shows the **first n lines**
- `tail -n` → shows the **last n lines**

Example:

```
ls -ltr | tail -1
```

 Returns the **most recently modified file** in the current directory.

 *Perfect for checking logs or large outputs quickly.*

# Linux: grep command

## The **grep** Command

The **grep** command searches files for **lines matching a pattern**.

### **Basic use:**

```
grep "error" logfile.txt
```


- → Prints lines containing the word *error*.

### **Search all files in a folder:**

```
grep "error" *
```

### **Search recursively through subfolders:**

```
grep -r "error" /var/log
```

 *Great for finding configurations, debugging errors, or locating specific text anywhere on your system.*





# Linux: grep examples

```
[ec2-user@ip-172-31-1-87 ~]$ grep "debug" /var/log/cloud-init.log -ic # counts number of lines with debug
957
[ec2-user@ip-172-31-1-87 ~]$ grep "debug" /var/log/cloud-init.log -i | tail -2 # prints last 2 lines
2025-10-20 21:19:58,137 - cc_power_state_change.py[DEBUG]: cmdline changed for 1611 [now: None]
2025-10-20 21:19:58,142 - cc_power_state_change.py[DEBUG]: check_condition command (test -f /run/cloud-init-se
linux-reboot): exited 1. condition not met.
[ec2-user@ip-172-31-1-87 ~]$ sudo grep "error" /var/log -ilr # prints every file name with error
/var/log/dnf.log
/var/log/dnf.rpm.log
/var/log/journal/ec21f4760dd031654c602cf85faff450/system.journal
/var/log/journal/ec21f4760dd031654c602cf85faff450/user-1000.journal
/var/log/chrony/tracking.log
/var/log/amazon/ssm/amazon-ssm-agent.log
/var/log/amazon/ssm/errors.log
/var/log/dnf.librepo.log.1
[ec2-user@ip-172-31-1-87 ~]$
```

# Linux: More File Commands

## Working with Text: **wc**, **sort**, **uniq**

### **wc** (Word Count)

Counts **lines**, **words**, and **characters** in a file.

```
wc file.txt
```

```
wc -l file.txt    # lines only
```

### **sort**

Arranges lines in **alphabetical or numerical order**.

```
sort names.txt
```

```
sort -r names.txt    # reverse order
```

### **uniq**

Filters out **repeated lines** (works best after **sort**).

```
sort names.txt | uniq
```

 *Combine them for quick text analysis and cleanup!*



# Linux: More File Commands

```
[ec2-user@ip-172-31-1-87 ~]$ cat data.txt
Linux 90
C 95
Java 92
Perl 44
Python 98
Python 98
C 95
[ec2-user@ip-172-31-1-87 ~]$ uniq data.txt # Are all lines outputted unique?
Linux 90
C 95
Java 92
Perl 44
Python 98
C 95
[ec2-user@ip-172-31-1-87 ~]$ sort data.txt | uniq # After sorting, they are unique
C 95
Java 92
Linux 90
Perl 44
Python 98
[ec2-user@ip-172-31-1-87 ~]$ sort -n -k2 data.txt
Perl 44
Linux 90
Java 92
C 95
C 95
Python 98
Python 98
[ec2-user@ip-172-31-1-87 ~]$ sort data.txt | uniq -c # count how many times a line is repeated
  2 C 95
  1 Java 92
  1 Linux 90
  1 Perl 44
  2 Python 98
```

# Linux: File Differences

## The **diff** Command

Compares **two files line by line** and shows what's different.

Example:

```
diff file1.txt file2.txt
```

### Symbols:

- **<** → Line only in the **first file**
- **>** → Line only in the **second file**

 Use **diff** to track changes, compare configs, or debug edits between versions.



# Linux: File Differences

```
[ec2-user@ip-172-31-1-87 ~]$ cat data.txt
Linux 90
C 95
Java 92
Perl 44
Python 98
Python 980000
C 95
[ec2-user@ip-172-31-1-87 ~]$ cat data-2.txt
Linux 90
C 10000
Java 92
Perl 44
Python 98
Python 98
C 95
[ec2-user@ip-172-31-1-87 ~]$ diff data.txt data-2.txt
2c2
< C 95
---
> C 10000
6c6
< Python 980000
---
> Python 98
[ec2-user@ip-172-31-1-87 ~]$
```

# Linux: File Differences

## The **tar** Command

Used to **archive** (and optionally compress) multiple files into one package.

### Common Uses:

Create an archive:

```
tar -cvf backup.tar folder/
```

Extract an archive:

```
tar -xvf backup.tar
```

Create a compressed archive:

```
tar -czvf backup.tar.gz folder/
```

Extract a compressed archive:

```
tar -xzvf backup.tar.gz
```

### Flags:

- **c** = create   **x** = extract   **v** = verbose   **f** = filename   **z** = gzip

 *tar bundles files for easy backup, sharing, or compression.*

# Lab 1.3 File Processing and Compression

Estimated Time: 40 Minutes



# POP QUIZ:

Which of these **tar** commands creates a compressed archive?

- A `.tar -xvf files.tar.gz`
- B `.tar -czvf files.tar.gz folder/`
- C `.tar -cvf files.tar folder/`
- D `.tar files.tar.gz folder/.`





# POP QUIZ:

Which of these `tar` commands creates a compressed archive?

- A `.tar -xvf files.tar.gz`
- B `.tar -czvf files.tar.gz folder/`
- C `.tar -cvf files.tar folder/`
- D `.tar files.tar.gz folder/.`

"What does the `-z` do?"



# POP QUIZ:

- In `diff file1 file2`, what does the `<` symbol mean?
- A. Line only in second file
  - B. Line only in first file
  - C. Line exists in both
  - D. Line is newer



# POP QUIZ:

In `diff file1 file2`, what does the `<` symbol mean?

- A. Line only in second file
- B. Line only in first file
- C. Line exists in both
- D. Line is newer

"What gets printed if line exists in both files?"





# POP QUIZ:

Which command finds lines containing “error” in all files of a directory?

- A. `grep error /var/log`
- B. `grep -r error /var/log`
- C. `find "error" /var/log`
- D. `search error /var/log`



# POP QUIZ:

Which command finds lines containing “error” in all files of a directory?

- A. `grep error /var/log`
- B. `grep -r error /var/log`
- C. `find "error" /var/log`
- D. `search error /var/log`

“What does -r do?”



# Lab 1.4 File Processing Challenge

Estimated Time: 30 Minutes






# Linux Day 1 Complete

## Today You Learned

- How to **navigate** the Linux file system
- About different file types and how to process them
- How to **redirect input and output**
- How to **archive and compress** files using tar

 *You now have the essentials to explore, manage, and process files confidently from the Linux command line!*

