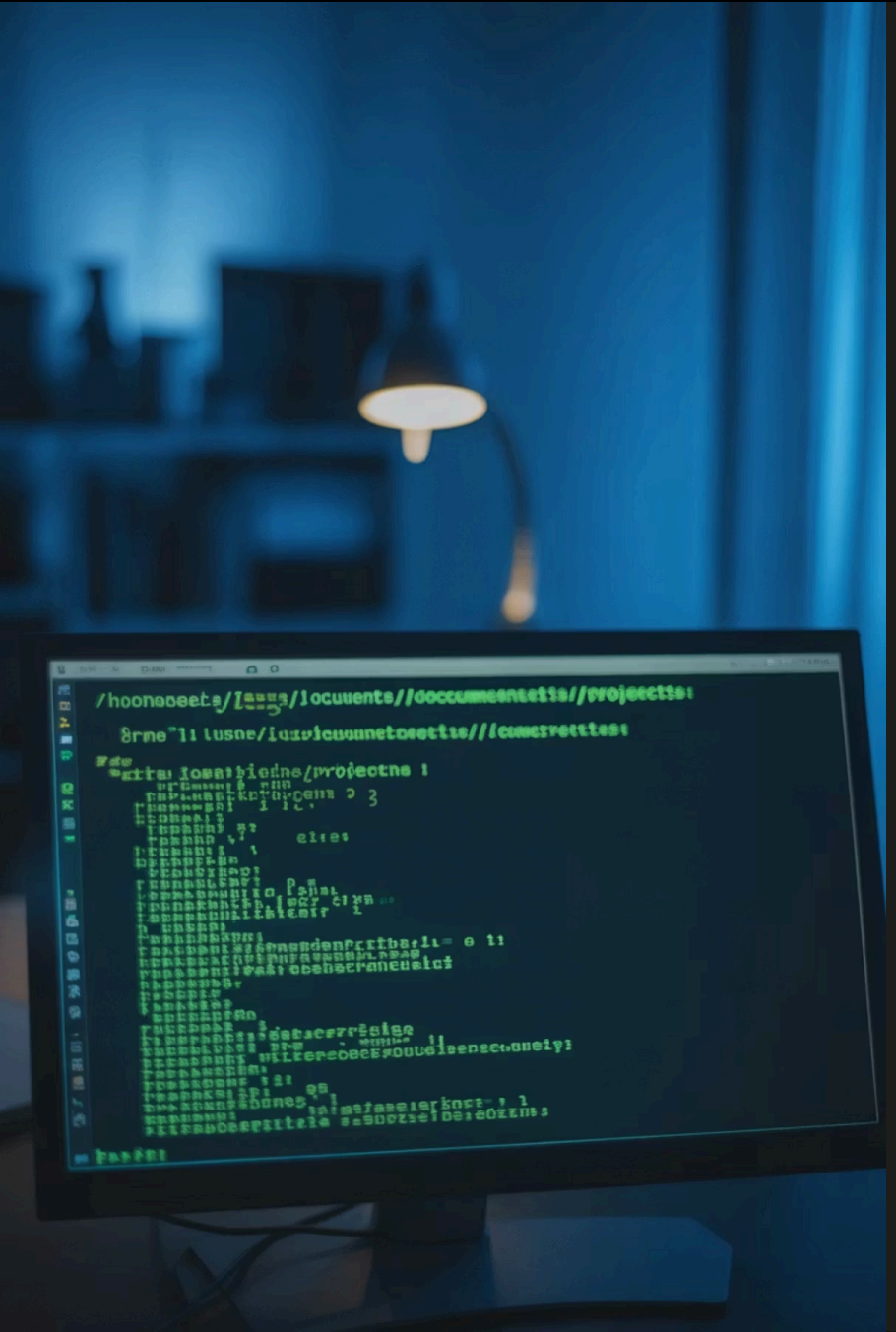


Linux File Systems and Permissions

A comprehensive guide to understanding file system hierarchies, permissions, and essential commands in Linux environments.

Linux File System Hierarchy

The Linux file system follows a standardised structure defined by the Filesystem Hierarchy Standard (FHS), creating consistency across distributions.

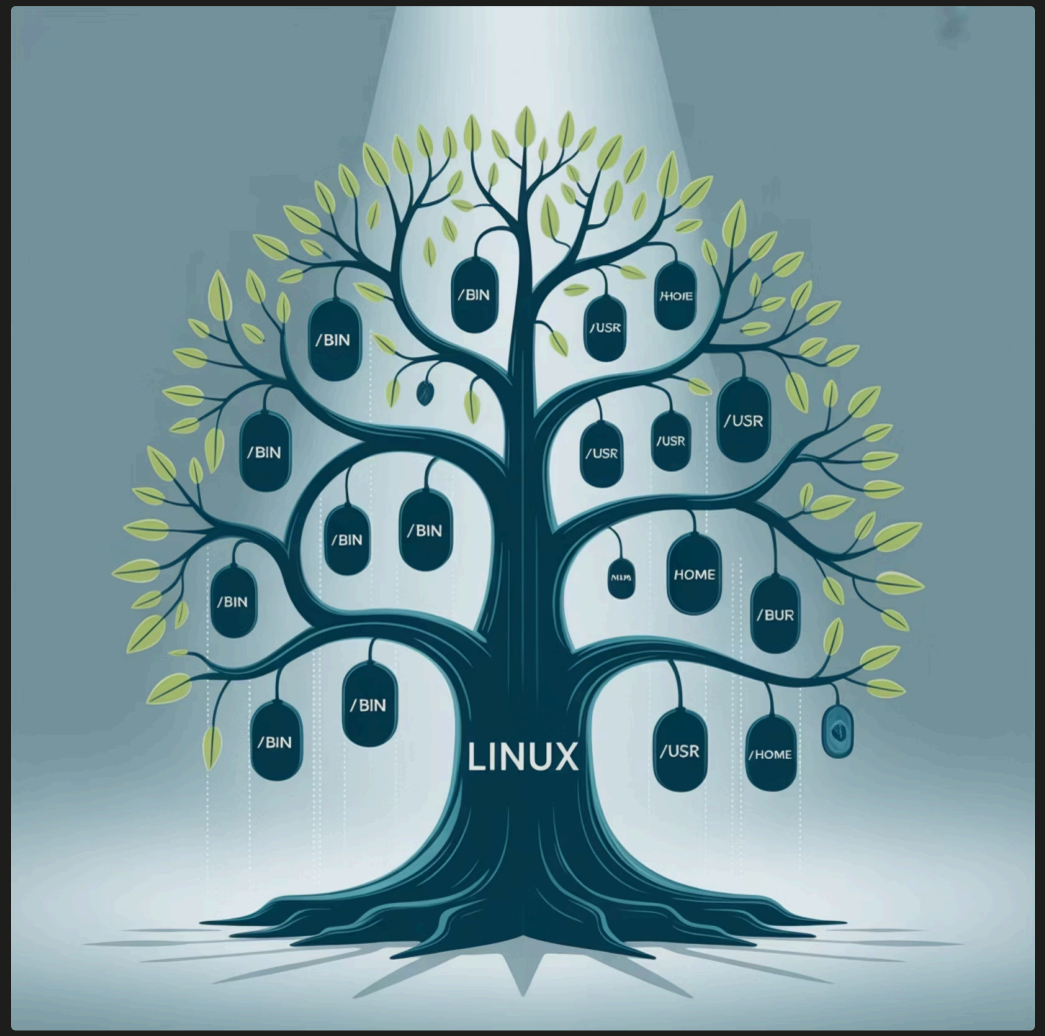
Root-Based Structure

All files and directories originate from the root directory, represented by a forward slash "/"

System Directories

Critical system directories include /etc for configuration files, /bin for essential binaries, /usr for user applications, and /home for user data

Root Directory and System Branches



The Starting Point

The "/" directory forms the base of every Linux file system, representing the absolute beginning of the directory hierarchy.

System Architecture

Critical system directories branch directly from root, creating a logical structure for different system components.

Root Access Control

Only the root user (superuser) has permissions to modify the contents of the root directory, ensuring system stability and security.

Home Directories

/home

Contains individual user home directories where personal files and configurations are stored

Example: /home/sarah contains all files owned by user "sarah"

Environment variables like $HOME point to the user's home directory

/root

The home directory for the superuser (root)

Separate from the system root directory "/"

Contains root user's personal files and configurations

User home directories are central to the Linux experience, providing dedicated spaces for each user's data, configurations, and personalised settings.

Commonly Used Linux File Systems

ext4

- Default on most distributions
- Fast, reliable, and mature
- Excellent backward compatibility
- Journaling prevents corruption during crashes

Btrfs

- Advanced features including snapshots
- Subvolumes for flexible organisation
- Integrated RAID capabilities
- Built-in checksums for data integrity

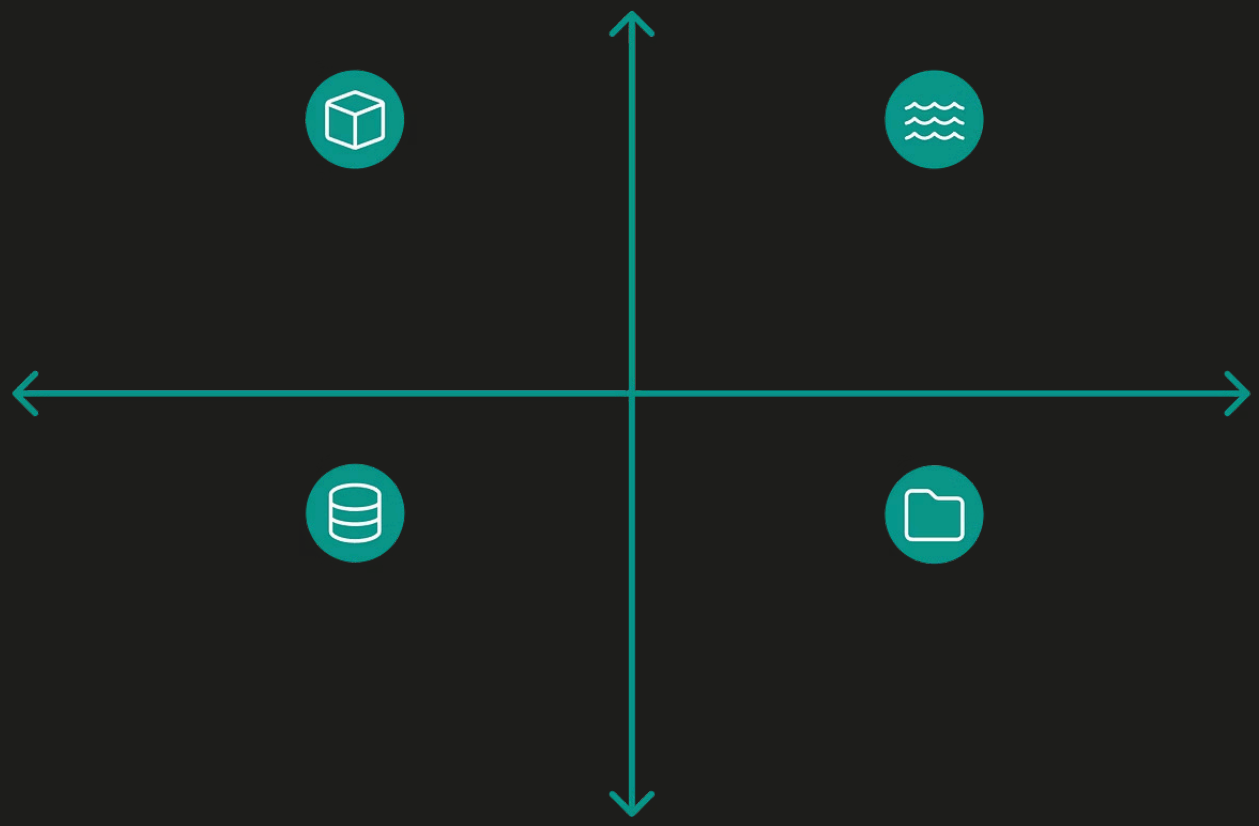
XFS

- High throughput performance
- Excellent with large files
- Support for parallel I/O operations
- Created by Silicon Graphics

ZFS

- Superior data integrity protection
- Massive capacity support
- Data deduplication capabilities
- Originally developed by Sun Microsystems

File System Features and Limitations



High Stability

Low Feature Richness

Low Stability

High Feature Richness

Btrfs

Feature rich, less stable

ZFS

Advanced features, stable but complex

XFS

Stable, fewer features, limited snapshots

ext4

Very stable, basic feature set

ext4: Volumes up to 1 EiB, files up to 16 TiB

XFS: Excellent for large files, weak with many small files

Mounting and Accessing File Systems

Unified Directory Tree

File systems can span multiple physical devices but appear under a single unified root "/"

Cross-Platform Compatibility

External drives often use exFAT, NTFS, or vfat for compatibility with Windows and macOS

Key Mount Commands

- mount /dev/sdb1 /mnt/external
- umount /mnt/external
- mount -a (mounts all in /etc/fstab)

Inodes and Metadata

Inode Storage

Every file and directory has a unique inode number that stores its metadata (not the actual content)

Metadata Contents

Inodes contain permissions, ownership information, file size, access/modification timestamps, and data block pointers

Name Mapping

Inodes do not store filenames; directory entries map human-readable names to corresponding inode numbers

Understanding inodes is crucial for troubleshooting file system issues, especially when dealing with hard links or inode exhaustion problems.

File Types in Linux

Regular Files (-)

Standard files containing data, text, or programs

Example: documents, scripts, binaries

Directories (d)

Special files that contain references to other files

Act as containers for organizing the file system

Symbolic Links (l)

Pointers to other files or directories

Similar to shortcuts in Windows

Device Files (b,c)

Interface to hardware devices

Block devices (b) and character devices (c)

Special Files (p,s)

FIFO pipes (p) for inter-process communication

Sockets (s) for network communication

Each file type is encoded in the inode and is visible as the first character in the output of `ls -l` command.

Linux Permissions: Overview

Owner (u)

The user who created or owns the file

Has primary control over the file

Group (g)

Users belonging to the file's assigned group

Share the same group permissions

Others (o)

All other users on the system

Typically have the most restricted access

Viewing Permissions

```
$ ls -l /home/user/documents
-rw-r--r-- 1 sarah developers 4096 Jul 10 14:23 report.txt
drwxr-x--- 2 sarah developers 4096 Jul 11 09:15 private/
lrwxrwxrwx 1 sarah developers   15 Jul 12 10:42 link.txt -> /etc/hosts
```

Permission String

First character indicates file type:

- '-' for regular file
- 'd' for directory
- 'l' for symbolic link

Permission Blocks

Three permission blocks (rwx):

- First block: Owner permissions
- Second block: Group permissions
- Third block: Others permissions

Additional Info

The output also shows:

- Number of links
- Owner and group names
- File size in bytes
- Last modification date

Changing Permissions: chmod

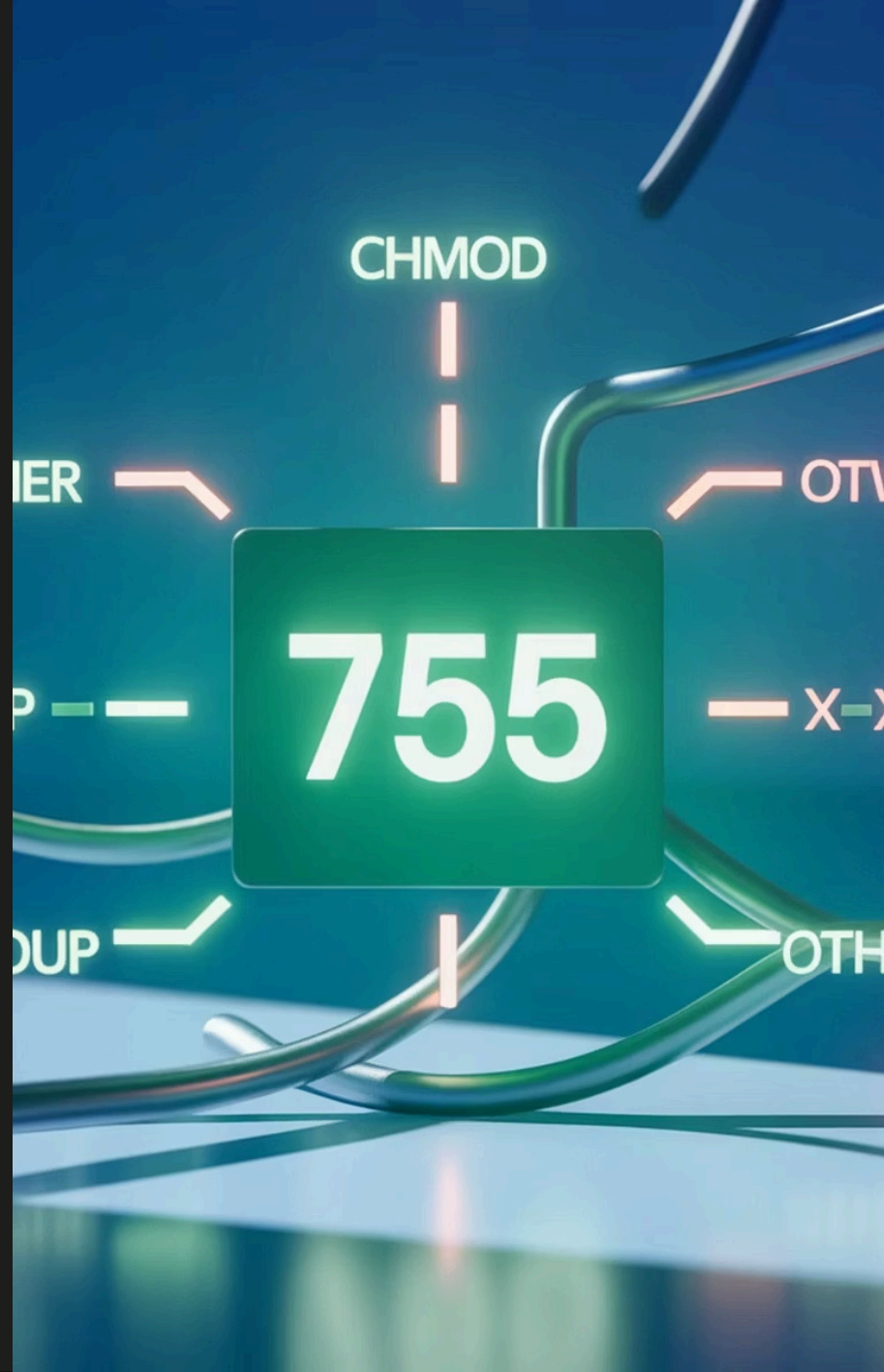
Symbolic Mode

```
# Add execute for owner
chmod u+x file.sh
# Remove write for group
and others
chmod go-w file.txt
# Set read and write for all
chmod a=rw document.pdf
```

Uses symbols to represent user classes (u,g,o,a) and permissions (r,w,x) with operators (+,-,=)

Octal Mode

```
# Set rwxr-xr-x (755)
chmod 755 script.sh
# Set rw-r----- (640)
chmod 640 sensitive.txt
# Set rwxrwxrwx (777)
chmod 777 public.sh
```



Managing Ownership: chown & chgrp

chown Command

Changes file/directory user and group ownership

```
# Change owner to sarah
chown sarah file.txt
# Change owner and group
chown sarah:developers file.txt
# Recursive ownership change
chown -R sarah:developers /projects
```

chgrp Command

Changes only the group ownership of a file/directory

```
# Change group to developers
chgrp developers file.txt
# Recursive group change
chgrp -R developers /projects
```

Only the root user or the file owner can change ownership. This prevents users from transferring ownership of sensitive files to unauthorized users.

Practical Examples and Tips

1. Creating Files and Directories

```
# Create empty file
touch report.txt

# Create directory
mkdir -p projects/website

# Create file with content
echo "Hello World" > greeting.txt
```

2. Finding Files by Permission

```
# Find world-writable files (security risk)
find / -type f -perm -o=w -ls 2>/dev/null

# Find setuid files
find / -perm -4000 -ls 2>/dev/null

# Find executable scripts in home directory
find /home -name "*.sh" -perm /u=x
```

3. Recursive Permission Changes

```
# Set directory permissions
chmod -R 755 /var/www/html

# Fix permissions for web files
find /var/www -type d -exec chmod 755 {} \;
find /var/www -type f -exec chmod 644 {} \;
```

Summary and Best Practices

File System Selection

- Choose based on workload requirements
- ext4 for general use and stability
- XFS for large files and high-performance needs
- Btrfs/ZFS for advanced features like snapshots

Security Considerations

- Be cautious with setuid/setgid binaries
- Monitor and restrict sudo access
- Keep root-owned directories secure
- Use ACLs for complex permission scenarios

Permission Management

- Apply principle of least privilege
- Regularly audit system permissions
- Avoid world-writable files and directories
- Use group permissions instead of "other" permissions

Maintenance Tasks

- Regularly check file system health
- Monitor disk space and inode usage
- Back up critical data to different file systems
- Document custom permission schemes