Linux System Administration Guide

Table of Contents

1. Terminal & Shell

Definition

Terminal: A terminal is a program that provides a text-based interface for interacting with the operating system. It's essentially a window where you can type commands and see their output. The terminal emulates the old physical terminals (teletypewriters) that were connected to mainframe computers.

Shell: A shell is a command-line interpreter that processes commands entered by the user. It acts as an interface between the user and the operating system kernel. The shell reads commands, interprets them, executes them, and returns the results.

Why These Names?

- **Terminal**: Named after the physical terminals (endpoints) that were connected to mainframe computers in the early days of computing. These were literally the “terminals” or endpoints of the connection.
- **Shell**: Named because it’s the outermost layer of the operating system that users interact with, like a shell around the kernel. It “shells” or encases the kernel, providing a protective and interactive layer.

Benefits Over GUI

1. **Speed and Efficiency**: Commands can be executed much faster than navigating through multiple GUI menus
2. **Resource Usage**: Command line uses significantly fewer system resources than graphical interfaces
3. **Automation**: Commands can be scripted and automated for repetitive tasks
4. **Remote Administration**: You can manage systems remotely over SSH connections efficiently
5. **Precision**: More precise control over system operations and configurations
6. **Batch Operations**: Perform operations on multiple files or systems simultaneously
7. **Accessibility**: Works on systems with minimal hardware or over slow network connections

How They Are Used

The terminal and shell work together in this process:

1. **User Input**: User types a command in the terminal

2. **Shell Interpretation**: Shell parses and interprets the command
3. **Execution**: Shell executes the command or calls appropriate system programs
4. **Output**: Results are displayed back in the terminal

Example Workflow:

```
$ ls -la /home
# User types command → Shell interprets → System executes → Output displays
total 12
drwxr-xr-x  3 root root 4096 Jan 15 10:30 .
drwxr-xr-x 19 root root 4096 Jan 15 09:45 ..
drwxr-xr-x  5 john john 4096 Jan 15 10:30 john
```

---

2. Everything in Linux is a File

Core Philosophy

Linux follows the principle that “everything is a file.” This means that all system resources, devices, processes, and data are represented as files in the filesystem. This unified approach simplifies system administration and programming.

What This Means in Practice

Regular Files

- Text files, executables, images, documents
- Example: `/home/user/document.txt`

Directories

- Special files that contain other files
- Example: `/home/user/Documents/`

Device Files

- **Block devices**: Storage devices like hard drives
  - Example: `/dev/sda1` (first partition of first SATA drive)
- **Character devices**: Input/output devices like keyboards, mice
  - Example: `/dev/tty1` (terminal device)

Process Files

- Information about running processes
- Example: `/proc/1234/` (information about process ID 1234)

System Information Files

- Kernel and system information
- Example: `/proc/cpuinfo` (CPU information)
- Example: `/sys/class/net/eth0/` (network interface information)

Special Files

- **Pipes**: For inter-process communication
- **Sockets**: For network and local communication

Practical Examples

```
# View CPU information as a file
$ cat /proc/cpuinfo

# Check memory usage as a file
$ cat /proc/meminfo

# View network interfaces
$ ls /sys/class/net/

# Read from a device file
$ cat /dev/random | head -c 10 | base64

# Everything has file permissions, even devices
$ ls -l /dev/sda1
brw-rw---- 1 root disk 8, 1 Jan 15 10:30 /dev/sda1
```

---

3. Linux Commands (Basics)

Essential Navigation Commands

`pwd` - Print Working Directory

Shows your current location in the filesystem.

```
$ pwd
/home/john/Documents
```

`ls` - List Directory Contents

Displays files and directories in the current or specified location.

```
$ ls
file1.txt  file2.txt  folder1/  folder2/

$ ls /etc
hosts  passwd  group  fstab  ...
```

`cd` - Change Directory

Navigates to different directories.

```
$ cd /home/john          # Absolute path
$ cd Documents           # Relative path
$ cd ..                  # Go up one level
$ cd ~                   # Go to home directory
$ cd -                   # Go to previous directory
```

File and Directory Operations

`mkdir` - Make Directory

Creates new directories.

```
$ mkdir newFolder
$ mkdir -p path/to/nested/folders    # Create nested directories
```

`rmdir` - Remove Directory

Removes empty directories.

```
$ rmdir emptyFolder
```

`rm` - Remove Files and Directories

Deletes files and directories.

```
$ rm file.txt                    # Remove file
$ rm -r folder/                  # Remove directory recursively
$ rm -rf folder/                # Force remove (be careful!)
```

`cp` - Copy Files and Directories

```
$ cp file1.txt file2.txt        # Copy file
$ cp file1.txt /path/to/dest/   # Copy to different location
$ cp -r folder1/ folder2/       # Copy directory recursively
```

`mv` - Move/Rename Files and Directories

```
$ mv oldname.txt newname.txt    # Rename file
$ mv file.txt /path/to/dest/    # Move file
$ mv folder1/ /path/to/dest/    # Move directory
```

File Content Operations

`cat` - Display File Content

```
$ cat file.txt                  # Display entire file
$ cat file1.txt file2.txt       # Display multiple files
```

`less` and `more` - Paginated File Viewing

```
$ less largefile.txt                # Navigate with arrow keys, q to quit
$ more largefile.txt                # Similar to less, space to continue
```

`head` and `tail` - Display File Portions

```
$ head -n 10 file.txt               # Show first 10 lines
$ tail -n 20 file.txt               # Show last 20 lines
$ tail -f logfile.log               # Follow file changes (useful for logs)
```

`grep` - Search Text Patterns

```
$ grep "pattern" file.txt           # Search for pattern in file
$ grep -i "pattern" file.txt        # Case-insensitive search
$ grep -r "pattern" /path/          # Recursive search in directory
```

System Information Commands

`ps` - Process Status

```
$ ps aux                            # Show all running processes
$ ps -ef                            # Alternative format
```

`top` - Dynamic Process Viewer

```
$ top                           # Real-time process monitoring
```

`df` - Disk Free Space

```
$ df -h                         # Human-readable disk usage
```

`du` - Directory Usage

```
$ du -sh /path/to/directory     # Show directory size
```

`free` - Memory Usage

```
$ free -h                       # Human-readable memory information
```

---

4. Commands and Their Flags (Arguments)

Understanding Command Structure

Linux commands follow this general structure:

```
command [options] [arguments]
```

Types of Options/Flags

Short Options (Single Character)

- Prefix with single dash: `-`
- Can be combined: `-la` is equivalent to `-l -a`

Long Options (Full Words)

- Prefix with double dash: `--`
- More descriptive: `--help`, `--version`

Common Flags Across Commands

Help and Information

```
$ command --help                # Show help information
$ man command                   # Show manual page
$ command --version             # Show version information
```

Verbose and Quiet

```
$ command -v                      # Verbose output (more details)
$ command -q                      # Quiet output (less details)
```

Command-Specific Examples

`ls` Flags

```
$ ls -l                           # Long format (detailed information)
$ ls -a                           # Show all files (including hidden)
$ ls -la                          # Combine long format and show all
$ ls -lh                          # Human-readable file sizes
$ ls -lt                          # Sort by modification time
$ ls -lr                          # Reverse order
$ ls --color=auto                 # Colorize output
```

`cp` Flags

```
$ cp -r source/ dest/             # Recursive copy (for directories)
$ cp -p file1 file2               # Preserve permissions and timestamps
$ cp -i file1 file2               # Interactive (ask before overwrite)
$ cp -v file1 file2               # Verbose (show what's being copied)
$ cp -u source/ dest/             # Update (copy only newer files)
```

`rm` Flags

```
$ rm -r directory/                # Recursive removal
$ rm -f file                      # Force removal (no confirmation)
$ rm -rf directory/               # Force recursive removal (DANGEROUS!)
$ rm -i file                      # Interactive (ask confirmation)
$ rm -v file                      # Verbose (show what's being removed)
```

`grep` Flags

```
$ grep -i "pattern" file          # Case-insensitive search
$ grep -v "pattern" file          # Invert match (show non-matching lines)
$ grep -n "pattern" file          # Show line numbers
$ grep -r "pattern" directory/    # Recursive search
$ grep -l "pattern" *.txt         # Show only filenames with matches
$ grep -c "pattern" file          # Count matching lines
```

`find` Flags

```
$ find /path -name "*.txt"        # Find files by name pattern
$ find /path -type f              # Find only files
$ find /path -type d              # Find only directories
$ find /path -size +1M            # Find files larger than 1MB
$ find /path -mtime -7            # Find files modified in last 7 days
$ find /path -perm 755            # Find files with specific permissions
```

Practical Examples with Multiple Flags

```
# Create a directory with parent directories and show verbose output
$ mkdir -pv /path/to/new/directory

# Copy files recursively, preserving attributes, with verbose output
$ cp -rpv source_directory/ destination_directory/

# List files in long format, showing all files, with human-readable sizes
$ ls -lah /var/log/

# Search for a pattern recursively, ignoring case, showing line numbers
$ grep -rin "error" /var/log/
```

---

5. File System (FHS)

Filesystem Hierarchy Standard (FHS)

The FHS defines the directory structure and contents in Unix-like operating systems, including Linux. It ensures consistency across different Linux distributions.

Root Directory Structure

`/` - Root Directory

The top-level directory of the filesystem hierarchy. All other directories branch from here.

`/bin` - Essential Binaries

Contains essential command binaries needed for system boot and single-user mode.

```
$ ls /bin
bash  cat  chmod  cp  date  echo  grep  ls  mkdir  mv  rm  sh
```

`/boot` - Boot Files

Contains static files required for system boot, including kernel images and bootloader files.

```
$ ls /boot
grub/  initrd.img  vmlinuz  config-5.4.0
```

`/dev` - Device Files

Contains device files representing hardware devices and virtual devices.

```
$ ls /dev
sda  sda1  sda2  tty1  null  zero  random  urandom
```

`/etc` - System Configuration

Contains system-wide configuration files and directories.

```
$ ls /etc
passwd  group  hosts  fstab  ssh/  apache2/  network/
```

Key files:

- `/etc/passwd` - User account information
- `/etc/group` - Group information
- `/etc/hosts` - Host name resolution
- `/etc/fstab` - Filesystem mount information

`/home` - User Home Directories

Contains home directories for regular users.

```
$ ls /home
john/  jane/  admin/
```

`/lib` - Essential Libraries

Contains shared library files needed by programs in `/bin` and `/sbin`.

```
$ ls /lib
libc.so.6  libm.so.6  modules/
```

`/media` - Removable Media

Mount point for removable media devices (USB drives, CDs, etc.).

```
$ ls /media
cdrom/  usb/
```

`/mnt` - Temporary Mount Points

Temporary mount point for mounting filesystems.

```
$ mount /dev/sdb1 /mnt/external_drive
```

`/opt` - Optional Software

Contains optional software packages, typically third-party software.

```
$ ls /opt
google/  oracle/  custom_software/
```

`/proc` - Process Information

Virtual filesystem containing information about running processes and kernel.

```
$ cat /proc/cpuinfo     # CPU information
$ cat /proc/meminfo     # Memory information
$ ls /proc/1234/        # Information about process ID 1234
```

`/root` - Root User Home

Home directory for the root user (system administrator).

`/run` - Runtime Data

Contains runtime data created since last boot.

```
$ ls /run
systemd/  NetworkManager/  user/
```

`/sbin` - System Binaries

Contains essential system binaries, typically used by administrators.

```
$ ls /sbin
fdisk  fsck  mount  umount  iptables  systemctl
```

`/srv` - Service Data

Contains data served by system services.

```
$ ls /srv
www/  ftp/  git/
```

`/sys` - System Information

Virtual filesystem providing information about devices and kernel.

```
$ ls /sys
block/  class/  devices/  fs/  kernel/
```

`/tmp` - Temporary Files

Temporary files that are usually cleared on boot.

```
$ ls /tmp
temp_file123  user_session_data
```

`/usr` - User Utilities

Contains user utilities and applications (secondary hierarchy).

Subdirectories:

- `/usr/bin` - User command binaries
- `/usr/lib` - User libraries
- `/usr/local` - Local software installations
- `/usr/share` - Shared data (documentation, icons)

`/var` - Variable Data

Contains files that change frequently during system operation.

Subdirectories:

- `/var/log` - Log files
- `/var/spool` - Print and mail queues
- `/var/tmp` - Temporary files preserved between boots
- `/var/www` - Web server files

Practical Navigation Examples

```
# View system logs
$ ls /var/log/
syslog  auth.log  kern.log  apache2/

# Check user accounts
$ cat /etc/passwd

# View mounted filesystems
```

```
$ cat /proc/mounts

# Find configuration files
$ find /etc -name "*.conf"

# Check system information
$ cat /proc/version
$ cat /sys/class/net/eth0/address
```

---

6. File Permissions

Permission System Overview

Linux uses a permission system to control access to files and directories. Every file and directory has associated permissions that determine who can read, write, or execute them.

Permission Types

Read ®

- **Files**: Permission to read/view file contents
- **Directories**: Permission to list directory contents

Write (w)

- **Files**: Permission to modify file contents
- **Directories**: Permission to create, delete, or rename files within the directory

Execute (x)

- **Files**: Permission to execute the file as a program
- **Directories**: Permission to enter (cd into) the directory

User Categories

Owner (u)

The user who owns the file or directory.

Group (g)

Users who belong to the file's group.

Others (o)

All other users on the system.

Permission Representation

Symbolic Notation

```
$ ls -l file.txt
-rw-r--r-- 1 john users 1234 Jan 15 10:30 file.txt
```

Breaking down `-rw-r--r--`:

- First character: File type (`-` for regular file, `d` for directory, `l` for link)
- Next 3 characters: Owner permissions (`rw-`)
- Next 3 characters: Group permissions (`r--`)
- Last 3 characters: Others permissions (`r--`)

Octal (Numeric) Notation

Each permission type has a numeric value:

- Read ® = 4
- Write (w) = 2
- Execute (x) = 1

Common permission combinations:

- `7` (rwx) = 4 + 2 + 1 = Full permissions
- `6` (rw-) = 4 + 2 = Read and write
- `5` (r-x) = 4 + 1 = Read and execute
- `4` (r–) = 4 = Read only
- `0` (—) = 0 = No permissions

Examples:

- `755` = rwxr-xr-x (Owner: full, Group/Others: read and execute)
- `644` = rw-r–r-- (Owner: read/write, Group/Others: read only)
- `600` = rw------- (Owner: read/write, Group/Others: no access)

Changing Permissions

`chmod` - Change Mode

```
# Using symbolic notation
$ chmod u+x file.txt          # Add execute permission for owner
$ chmod g-w file.txt          # Remove write permission for group
$ chmod o+r file.txt          # Add read permission for others
$ chmod ugo+r file.txt        # Add read permission for all
$ chmod a+r file.txt          # Add read permission for all (alternative)

# Using octal notation
$ chmod 755 file.txt          # Set permissions to rwxr-xr-x
$ chmod 644 file.txt          # Set permissions to rw-r--r--
$ chmod 600 file.txt          # Set permissions to rw-------

# Recursive permission change
$ chmod -R 755 directory/     # Apply permissions recursively
```

`chown` - Change Owner

```
$ chown john file.txt          # Change owner to john
$ chown john:users file.txt    # Change owner to john and group to users
$ chown :admin file.txt        # Change only the group to admin
$ chown -R john:users dir/     # Change ownership recursively
```

`chgrp` - Change Group

```
$ chgrp admin file.txt         # Change group to admin
$ chgrp -R developers dir/     # Change group recursively
```

Special Permissions

Setuid (Set User ID) - 4

When executed, the file runs with the privileges of the file owner.

```
$ chmod 4755 program           # Set setuid bit
$ ls -l program
-rwsr-xr-x 1 root root 12345 Jan 15 10:30 program
```

Setgid (Set Group ID) - 2

- **Files**: Run with the privileges of the file's group
- **Directories**: New files inherit the directory's group

```
$ chmod 2755 directory        # Set setgid bit on directory
$ ls -ld directory
drwxr-sr-x 2 john admin 4096 Jan 15 10:30 directory
```

Sticky Bit - 1

Applied to directories, prevents users from deleting files they don't own.

```
$ chmod 1755 /tmp             # Set sticky bit (common on /tmp)
$ ls -ld /tmp
drwxrwxrwt 10 root root 4096 Jan 15 10:30 /tmp
```

Practical Examples

```
# Create a script and make it executable
$ echo '#!/bin/bash\necho "Hello World"' > script.sh
$ chmod +x script.sh
$ ./script.sh

# Set up a shared directory
$ mkdir shared_folder
$ chmod 775 shared_folder
$ chgrp developers shared_folder
$ chmod g+s shared_folder      # Set setgid for group inheritance

# Secure a private file
```

```
$ chmod 600 private_file.txt

# View file permissions with details
$ ls -la
total 20
drwxr-xr-x  2 john users 4096 Jan 15 10:30 .
drwxr-xr-x 25 john users 4096 Jan 15 10:29 ..
-rw-r--r--  1 john users  220 Jan 15 10:30 .bashrc
-rwxr-xr-x  1 john users   45 Jan 15 10:30 script.sh
-rw-------  1 john users  123 Jan 15 10:30 private_file.txt
```

---

7. User Management

User Account Fundamentals

Linux is a multi-user system where each user has a unique identity, home directory, and specific permissions. User management is crucial for system security and organization.

User Types

System Users (UID 0-999)

- Root user (UID 0): System administrator with unlimited privileges
- System service users: Run system services and daemons

Regular Users (UID 1000+)

- Interactive users who can log in and use the system
- Limited privileges for security

User Account Information

`/etc/passwd` - User Database

Contains basic user account information.

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
john:x:1001:1001:John Doe,,,:/home/john:/bin/bash
```

Format: `username:password:UID:GID:GECOS:home_directory:shell`

- `username`: Login name
- `password`: Usually 'x' (actual passwords in /etc/shadow)
- `UID`: User ID number
- `GID`: Primary group ID
- `GECOS`: Full name and additional info
- `home_directory`: Path to user's home directory
- `shell`: Default shell program

`/etc/shadow` - Password Database

Contains encrypted passwords and password policies.

```
$ sudo cat /etc/shadow
root:$6$randomsalt$hashedpassword:18000:0:99999:7:::
john:$6$randomsalt$hashedpassword:18500:0:99999:7:::
```

User Management Commands

`useradd` - Add User Account

```
# Basic user creation
$ sudo useradd john

# Create user with home directory and shell
$ sudo useradd -m -s /bin/bash jane

# Create user with specific UID and groups
$ sudo useradd -u 1500 -g users -G sudo,developers -m bob

# Create user with full options
$ sudo useradd -c "John Doe" -d /home/john -m -s /bin/bash -G sudo john
```

Common options:

- `-m` : Create home directory
- `-s` : Specify shell

- `-c` : Add comment (full name)
- `-d` : Specify home directory path
- `-g` : Specify primary group
- `-G` : Specify additional groups
- `-u` : Specify UID

`usermod` - Modify User Account

```
# Change user's shell
$ sudo usermod -s /bin/zsh john

# Add user to additional groups
$ sudo usermod -aG sudo,developers john

# Change user's home directory
$ sudo usermod -d /new/home/path -m john

# Lock/unlock user account
$ sudo usermod -L john          # Lock account
$ sudo usermod -U john          # Unlock account

# Change user's full name
$ sudo usermod -c "John Smith" john
```

`userdel` - Delete User Account

```
# Delete user (keep home directory)
$ sudo userdel john

# Delete user and home directory
$ sudo userdel -r john

# Force deletion even if user is logged in
$ sudo userdel -f john
```

`passwd` - Password Management

```
# Change your own password
$ passwd

# Change another user's password (as root)
$ sudo passwd john

# Set password expiry
$ sudo passwd -e john          # Force password change on next login

# Lock/unlock password
$ sudo passwd -l john          # Lock password
$ sudo passwd -u john          # Unlock password

# Display password status
$ sudo passwd -S john
```

User Information Commands

`id` - Display User and Group IDs

```
$ id
uid=1001(john) gid=1001(john) groups=1001(john),27(sudo),1002(developers)

$ id john
uid=1001(john) gid=1001(john) groups=1001(john),27(sudo)
```

`who` and `w` - Display Logged-in Users

```
$ who
john     pts/0        2024-01-15 10:30 (192.168.1.100)
jane     tty2         2024-01-15 09:15

$ w
 10:45:23 up 2 days,  4:32,  2 users,  load average: 0.15, 0.23, 0.18
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
john     pts/0    192.168.1.100    10:30    0.00s  0.12s  0.01s w
jane     tty2     -                09:15   34:23   0.05s  0.05s -bash
```

`su` - Switch User

```
# Switch to root user
$ su -
```

```
# Switch to another user
$ su - john

# Execute single command as another user
$ su -c "ls /root" -
```

`sudo` - Execute Commands as Another User

```
# Run command as root
$ sudo apt update

# Run command as specific user
$ sudo -u john ls /home/john

# Switch to root shell
$ sudo -i

# Edit files safely
$ sudo nano /etc/hosts
```

User Environment

Home Directory Setup

```
# Default files created in new user home
$ ls -la /home/john
-rw-r--r-- 1 john john  220 Jan 15 10:30 .bash_logout
-rw-r--r-- 1 john john 3771 Jan 15 10:30 .bashrc
-rw-r--r-- 1 john john  807 Jan 15 10:30 .profile
```

Shell Configuration

```
# View available shells
$ cat /etc/shells
/bin/bash
/bin/dash
/bin/zsh
/bin/sh

# Change default shell
$ chsh -s /bin/zsh john
```

Practical User Management Examples

```
# Create a developer user account
$ sudo useradd -m -s /bin/bash -c "Developer User" -G developers dev1

# Set initial password
$ sudo passwd dev1
```

```
# Create multiple users from a script
#!/bin/bash
for user in alice bob carol; do
    sudo useradd -m -s /bin/bash -G users "$user"
    echo "Created user: $user"
done

# Find all users with UID >= 1000 (regular users)
$ awk -F: '$3>=1000 {print $1}' /etc/passwd

# Display user account details
$ getent passwd john
john:x:1001:1001:John Doe,,,:/home/john:/bin/bash
```

---

8. Group Management

Group System Overview

Groups in Linux are collections of users that share common permissions and access rights. They provide an efficient way to manage permissions for multiple users simultaneously.

Group Types

Primary Group

- Every user must belong to exactly one primary group
- Specified in `/etc/passwd` (GID field)
- New files created by the user belong to this group by default

Secondary Groups

- Additional groups a user can belong to
- Listed in `/etc/group` file
- User inherits permissions from all secondary groups

Group Information Files

`/etc/group` - Group Database

Contains group information and membership.

```
$ cat /etc/group
root:x:0:
sudo:x:27:john,jane
users:x:100:john,bob
developers:x:1001:alice,bob,carol
```

Format: `group_name:password:GID:user_list`

- `group_name`: Name of the group
- `password`: Usually 'x' (group passwords rarely used)

- `GID`: Group ID number
- `user_list`: Comma-separated list of group members

`/etc/gshadow` - Group Shadow File

Contains group passwords and administrative information.

```
$ sudo cat /etc/gshadow
sudo:*::john,jane
developers:!:admin:alice,bob,carol
```

Group Management Commands

`groupadd` - Create Group

```
# Create a basic group
$ sudo groupadd developers

# Create group with specific GID
$ sudo groupadd -g 1500 marketing

# Create system group (GID < 1000)
$ sudo groupadd -r system_service
```

`groupmod` - Modify Group

```
# Change group name
$ sudo groupmod -n newname oldname

# Change group GID
$ sudo groupmod -g 1600 developers
```

`groupdel` - Delete Group

```
# Delete group (only if no users have it as primary group)
$ sudo groupdel developers

# Check if group is safe to delete
$ grep developers /etc/passwd
```

`gpasswd` - Group Password and Membership Management

```
# Add user to group
$ sudo gpasswd -a john developers

# Remove user from group
$ sudo gpasswd -d john developers

# Set group administrators
$ sudo gpasswd -A admin1,admin2 developers

# Set group members (replaces existing list)
$ sudo gpasswd -M user1,user2,user3 developers
```

```
# Remove group password
$ sudo gpasswd -r developers
```

`usermod` - Add/Remove Users to/from Groups

```
# Add user to secondary groups (append)
$ sudo usermod -aG sudo,developers john

# Set user's secondary groups (replace)
$ sudo usermod -G users,developers john

# Change user's primary group
$ sudo usermod -g developers john
```

Group Information Commands

`groups` - Display User’s Groups

```
# Show current user's groups
$ groups
john sudo developers users

# Show specific user's groups
$ groups alice
```

```
alice : alice developers marketing
```

`getent` - Query Group Database

```
# Show all groups
$ getent group

# Show specific group
$ getent group developers
developers:x:1001:alice,bob,carol

# Show groups for a user
$ getent group | grep john
sudo:x:27:john,jane
users:x:100:john,bob
developers:x:1001:john,alice,bob
```

Practical Group Management

Creating Project-Based Groups

```
# Create groups for different projects
$ sudo groupadd web_team
$ sudo groupadd database_team
$ sudo groupadd security_team
```

```
# Add users to appropriate groups
$ sudo usermod -aG web_team alice,bob
$ sudo usermod -aG database_team carol,david
$ sudo usermod -aG security_team eve,frank

# Create shared directories with group ownership
$ sudo mkdir /projects/{web,database,security}
$ sudo chgrp web_team /projects/web
$ sudo chgrp database_team /projects/database
$ sudo chgrp security_team /projects/security

# Set appropriate permissions
$ sudo chmod 775 /projects/web
$ sudo chmod g+s /projects/web  # Set setgid for group inheritance
```

Managing System Administration Groups

`sudo` Group

Users in the sudo group can execute commands with elevated privileges.

```
# Add user to sudo group
$ sudo usermod -aG sudo john

# Check sudo group membership
$ getent group sudo
sudo:x:27:john,jane,admin

# Test sudo access
```

```
$ sudo -l -U john
User john may run the following commands:
    (ALL : ALL) ALL
```

`wheel` Group (on some systems)

Similar to sudo group, provides administrative privileges.

```
# Add user to wheel group
$ sudo usermod -aG wheel admin
```

Group-Based File Permissions

Setting Up Shared Directories

```
# Create a shared directory for developers
$ sudo mkdir /shared/development
$ sudo chgrp developers /shared/development
$ sudo chmod 775 /shared/development
$ sudo chmod g+s /shared/development

# Verify permissions
$ ls -ld /shared/development
drwxrwsr-x 2 root developers 4096 Jan 15 10:30 /shared/development

# Test group access
```

```
$ sudo -u alice touch /shared/development/test_file
$ ls -l /shared/development/test_file
-rw-rw-r-- 1 alice developers 0 Jan 15 10:30 test_file
```

Advanced Group Permissions with ACLs

```
# Install ACL utilities (if not already installed)
$ sudo apt install acl

# Set default ACL for group
$ sudo setfacl -d -m g:developers:rwx /shared/development

# View ACL settings
$ getfacl /shared/development
# file: shared/development
# owner: root
# group: developers
# flags: -s-
user::rwx
group::rwx
other::r-x
default:user::rwx
default:group::rwx
default:group:developers:rwx
default:mask::rwx
default:other::r-x
```

Group Administration Best Practices

Naming Conventions

```
# Use descriptive group names
developers          # Good
marketing_team      # Good
proj_alpha         # Good
grp1               # Poor - not descriptive
```

Regular Group Audits

```
#!/bin/bash
# Script to audit group memberships

echo "=== Group Membership Audit ==="
echo "Date: $(date)"
echo

# Show all groups and their members
for group in $(cut -d: -f1 /etc/group); do
    members=$(getent group $group | cut -d: -f4)
    if [ -n "$members" ]; then
        echo "Group: $group"
        echo "Members: $members"
        echo "---"
    fi
done
```

```
# Find users with sudo access
echo "=== Users with sudo access ==="
getent group sudo | cut -d: -f4 | tr ',' '\n'
```

Removing Unused Groups

```
# Find groups with no members
$ for group in $(cut -d: -f1 /etc/group); do
    members=$(getent group $group | cut -d: -f4)
    if [ -z "$members" ]; then
        gid=$(getent group $group | cut -d: -f3)
        if [ $gid -gt 999 ]; then  # Only check non-system groups
            echo "Empty group: $group (GID: $gid)"
        fi
    fi
  done

# Check if group is safe to delete (not a primary group)
$ awk -F: -v group="old_group" '$4==group {print "User " $1 " has " group " as primary group"}' /etc/passwd
```

Troubleshooting Group Issues

Common Group Problems and Solutions

User Can't Access Group Files

```
# Check user's group membership
$ groups username

# Check if user needs to log out/in for group changes to take effect
$ su - username  # or ask user to log out/in

# Verify file/directory group ownership and permissions
$ ls -la /path/to/file
```

Group Permissions Not Working

```
# Check if setgid is set on directory
$ ls -ld /shared/directory
drwxrwsr-x 2 root groupname 4096 Jan 15 10:30 /shared/directory

# Set setgid if missing
$ sudo chmod g+s /shared/directory
```

Finding All Files Owned by a Group

```
# Find all files owned by specific group
$ find / -group developers -type f 2>/dev/null

# Find files and directories owned by group
$ find /home -group developers 2>/dev/null
```

Group Management Scripts

Bulk Group Operations

```bash
#!/bin/bash
# Script to create multiple groups and assign users

# Create project groups
groups=(web_frontend web_backend mobile_ios mobile_android qa_team)

for group in "${groups[@]}"; do
    if ! getent group $group > /dev/null 2>&1; then
        sudo groupadd $group
        echo "Created group: $group"
    else
        echo "Group $group already exists"
    fi
done

# Assign users to groups based on roles
sudo usermod -aG web_frontend,web_backend alice
sudo usermod -aG mobile_ios,mobile_android bob
sudo usermod -aG qa_team carol

echo "Group assignments completed"
```

Group Cleanup Script

```bash
#!/bin/bash
# Script to clean up unused groups

echo "Finding potentially unused groups..."

for group in $(awk -F: '$3 > 999 {print $1}' /etc/group); do
    # Check if any user has this as primary group
    primary_users=$(awk -F: -v gid=$(getent group $group | cut -d: -f3) '$4==gid {print $1}' /etc/passwd)

    # Check if group has secondary members
    secondary_members=$(getent group $group | cut -d: -f4)

    if [ -z "$primary_users" ] && [ -z "$secondary_members" ]; then
        echo "Unused group found: $group"
        echo "  No users have this as primary or secondary group"
        echo "  Use 'sudo groupdel $group' to remove"
    fi
done
```

This concludes the comprehensive Linux System Administration Guide covering all the topics you requested. The guide provides detailed explanations, practical examples, and real-world scenarios for each concept, making it suitable for both learning and reference during your lecture.