Complete Linux Reference Guide

Table of Contents

Piping

What is Piping?

Piping is a mechanism in Linux that allows you to connect the output of one command directly to the input of another command. The pipe operator `|` is used to create this connection, enabling powerful command combinations and data processing workflows.

How Piping Works

When you use a pipe, the shell creates a temporary buffer (pipe) in memory. The first command writes its output to this buffer, and the second command reads from it. This happens simultaneously, making it very efficient for processing large amounts of data.

Basic Syntax

```bash
command1 | command2 | command3
```

Common Piping Examples

1. List files and count them

```bash
ls -la | wc -l
```

Explanation: Lists all files in detail format and counts the number of lines (files).

2. Find processes and filter

```bash
ps aux | grep firefox
```

Explanation: Shows all running processes and filters only those containing "firefox".

3. Sort and remove duplicates

```bash
cat file.txt | sort | uniq
```

Explanation: Displays file content, sorts it alphabetically, and removes duplicate lines.

4. Chain multiple operations

```bash
cat /var/log/syslog | grep "error" | sort | uniq -c | sort -nr
```

Explanation:

- Reads system log
- Filters lines containing "error"
- Sorts them
- Counts unique occurrences
- Sorts by count in descending order

Advanced Piping Techniques

Tee Command (Split output)

```bash
ls -la | tee file_list.txt | grep "^d"
```

Explanation: Saves the output to a file AND passes it to the next command.

Named Pipes (FIFOs)

```bash
mkfifo mypipe
command1 > mypipe &
command2 < mypipe
```

Find Command

What is Find?

The `find` command is a powerful utility for searching files and directories in the Linux filesystem. It can search based on various criteria like name, size, type, permissions, and modification time.

How Find Works

Find traverses the directory tree starting from a specified path and evaluates each file/directory against the given criteria. It performs a depth-first search and can execute actions on matching files.

Basic Syntax

bash

```
find [path] [options] [expression] [action]
```

Common Find Options

1. Search by Name

bash

```
find /home -name "*.txt"
find . -name "config*"
find / -iname "*.PDF"  # Case insensitive
```

2. Search by Type

bash

```
find /var -type f       # Files only
find /home -type d      # Directories only
find /dev -type l       # Symbolic links
find /tmp -type s       # Sockets
```

3. Search by Size

bash

```
find /home -size +100M   # Files larger than 100MB
find /tmp -size -1k      # Files smaller than 1KB
find . -size 50c         # Files exactly 50 bytes
```

4. Search by Time

bash

```
find /var/log -mtime -7     # Modified in last 7 days
find /home -atime +30       # Accessed more than 30 days ago
find /tmp -ctime -1         # Changed in last 24 hours
```

5. Search by Permissions

bash

```
find /home -perm 755        # Exact permissions
find /var -perm -644        # At least these permissions
find /tmp -perm /644        # Any of these permissions
```

Advanced Find Examples

1. Find and Execute Commands

bash

```
find /home -name "*.log" -exec rm {} \;
find . -type f -name "*.c" -exec gcc {} -o {}.out \;
```

2. Find with Multiple Conditions

bash

```
find /var -name "*.log" -size +10M -mtime -7
find /home -type f \( -name "*.jpg" -o -name "*.png" \)
```

3. Find and List Details

bash

```
find /etc -name "*.conf" -ls
find /home -type f -printf "%s %p\n" | sort -n
```

Grep Command

What is Grep?

Grep (Global Regular Expression Print) is a command-line utility for searching text patterns within files. It uses regular expressions to match patterns and can perform complex text searches.

How Grep Works

Grep reads input line by line and compares each line against the specified pattern. When a match is found, it prints the matching line (or performs other specified actions). It supports various pattern matching techniques including literal strings and regular expressions.

Basic Syntax

bash

```
grep [options] pattern [file...]
```

Common Grep Options

1. Basic Search

bash

```
grep "error" /var/log/syslog
grep -i "ERROR" file.txt          # Case insensitive
grep -v "success" log.txt         # Invert match (exclude)
grep -n "function" script.py      # Show line numbers
```

2. Multiple Files

bash

```
grep -r "TODO" /home/user/projects/    # Recursive search
grep -l "main" *.c                     # List files with matches
grep -L "debug" *.log                  # List files without matches
```

3. Context Options

bash

```
grep -A 3 "error" log.txt         # 3 lines after match
grep -B 2 "warning" log.txt       # 2 lines before match
grep -C 5 "exception" log.txt     # 5 lines before and after
```

Regular Expressions with Grep

1. Basic Regex

bash

```
grep "^start" file.txt            # Lines starting with "start"
grep "end$" file.txt              # Lines ending with "end"
grep "t.st" file.txt              # Match "test", "tast", "t1st"
grep "colou?r" file.txt           # Match "color" or "colour"
```

2. Extended Regex (-E option)

bash

```
grep -E "error|warning|critical" log.txt
grep -E "^[0-9]{1,3}\.[0-9]{1,3}" file.txt    # IP addresses
grep -E "\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b" file.txt  # Email
```

3. Character Classes

bash

```
grep "[0-9]" file.txt           # Any digit
grep "[A-Z]" file.txt           # Any uppercase letter
grep "[[:alpha:]]" file.txt     # Any letter
grep "[[:space:]]" file.txt     # Any whitespace
```

Advanced Grep Examples

1. Count Matches

bash

```
grep -c "error" /var/log/syslog
grep -c "^$" file.txt           # Count empty lines
```

2. Only Show Matches

bash

```
grep -o "http://[^[:space:]]*" file.txt
grep -oE "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" log.txt
```

Users and Groups

What are Users and Groups?

Linux is a multi-user system where each user has a unique identity and specific permissions. Groups are collections of users that share common access rights. This system provides security and organization for system resources.

How User/Group System Works

- Each user has a unique User ID (UID) and belongs to at least one group
- Each group has a unique Group ID (GID)

- User information is stored in `/etc/passwd`
- Group information is stored in `/etc/group`
- Password hashes are stored in `/etc/shadow`

User Management Commands

1. Creating Users

bash

```bash
sudo useradd -m -s /bin/bash newuser       # Create user with home directory
sudo useradd -m -G sudo,developers john    # Create user and add to groups
sudo passwd newuser                        # Set password
```

2. Modifying Users

bash

```bash
sudo usermod -aG sudo username             # Add user to sudo group
sudo usermod -s /bin/zsh username          # Change shell
sudo usermod -l newname oldname            # Change username
sudo usermod -d /new/home -m username      # Change home directory
```

3. Deleting Users

bash

```bash
sudo userdel username                      # Delete user (keep home)
sudo userdel -r username                   # Delete user and home directory
```

4. User Information

bash

```bash
id username                                # Show user ID and groups
whoami                                     # Current user
who                                        # Logged in users
w                                          # Detailed user activity
finger username                            # User information
```

Group Management Commands

1. Creating Groups

bash

```
sudo groupadd developers              # Create new group
sudo groupadd -g 1500 customgroup     # Create with specific GID
```

2. Modifying Groups

bash

```
sudo gpasswd -a username groupname       # Add user to group
sudo gpasswd -d username groupname       # Remove user from group
sudo gpasswd -A admin1,admin2 groupname  # Set group administrators
```

3. Group Information

bash

```
groups username                  # Show user's groups
getent group groupname           # Show group members
cat /etc/group | grep groupname  # Group details
```

Important Files

/etc/passwd Format

```
username:x:UID:GID:comment:home_directory:shell
```

Example: `john:x:1001:1001:John Doe:/home/john:/bin/bash`

/etc/group Format

```
groupname:x:GID:user_list
```

Example: `developers:x:1002:john,jane,bob`

File Permissions

What are File Permissions?

File permissions in Linux control who can read, write, or execute files and directories. This system ensures security by restricting access to files based on ownership and group membership.

How Permissions Work

Every file and directory has three types of permissions for three categories of users:

- **Owner (u)**: The user who owns the file
- **Group (g)**: Users who belong to the file's group
- **Others (o)**: All other users

Permission Types

- **Read (r)**: View file contents or list directory contents
- **Write (w)**: Modify file contents or create/delete files in directory
- **Execute (x)**: Run file as program or access directory

Permission Representation

1. Symbolic Notation

```bash
-rwxrw-r--
```

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

2. Octal Notation

```bash
chmod 764 file.txt
```

- 7 (111 binary) = rwx for owner
- 6 (110 binary) = rw- for group
- 4 (100 binary) = r-- for others

Common Permission Values

```
0 = --- (no permissions)
1 = --x (execute only)
2 = -w- (write only)
3 = -wx (write and execute)
4 = r-- (read only)
5 = r-x (read and execute)
6 = rw- (read and write)
7 = rwx (read, write, and execute)
```

Changing Permissions

1. Using chmod with Octal

bash

```
chmod 755 script.sh              # rwxr-xr-x
chmod 644 document.txt           # rw-r--r--
chmod 600 private.txt            # rw-------
chmod 777 public_dir             # rwxrwxrwx
```

2. Using chmod with Symbolic

bash

```
chmod u+x script.sh              # Add execute for owner
chmod g-w file.txt               # Remove write for group
chmod o=r file.txt               # Set others to read only
chmod a+r file.txt               # Add read for all
chmod u+x,g+w,o-r file.txt       # Multiple changes
```

Changing Ownership

1. Change Owner

bash

```
sudo chown newowner file.txt
sudo chown newowner:newgroup file.txt
sudo chown -R newowner directory/     # Recursive
```

2. Change Group

bash

```
sudo chgrp newgroup file.txt
sudo chgrp -R developers project/     # Recursive
```

Special Permissions

1. Setuid (SUID) - 4000

bash

```
chmod 4755 /usr/bin/passwd          # Run as file owner
ls -l /usr/bin/passwd               # -rwsr-xr-x
```

2. Setgid (SGID) - 2000

```bash
chmod 2755 /shared/directory        # Inherit group ownership
ls -ld /shared/directory            # drwxr-sr-x
```

3. Sticky Bit - 1000

```bash
chmod 1755 /tmp                     # Only owner can delete
ls -ld /tmp                         # drwxrwxrwt
```

Default Permissions (umask)

```bash
umask                               # Show current umask
umask 022                           # Set new umask
umask -S                            # Show in symbolic format
```

Disk Partitions

What are Disk Partitions?

Disk partitions are logical divisions of a physical storage device. Each partition acts as a separate storage unit with its own filesystem, allowing for better organization, security, and management of data.

How Partitioning Works

- **Primary Partitions**: Up to 4 per disk (MBR) or 128 (GPT)
- **Extended Partition**: Container for logical partitions (MBR only)
- **Logical Partitions**: Created within extended partitions
- **Partition Table**: Stores partition information (MBR or GPT)

Partition Types

- **MBR (Master Boot Record)**: Legacy, 2TB limit, 4 primary partitions
- **GPT (GUID Partition Table)**: Modern, larger disks, 128 partitions

Viewing Disk and Partition Information

1. List Block Devices

bash

```
lsblk                    # Tree view of devices
lsblk -f                 # Show filesystem info
lsblk -p                 # Show full device paths
```

2. Disk Usage Information

bash

```
df -h                    # Human readable disk usage
df -i                    # Inode usage
du -sh /home/*           # Directory sizes
du -h --max-depth=1 /var # Subdirectory sizes
```

3. Detailed Disk Information

bash

```
sudo fdisk -l            # List all disks and partitions
sudo parted -l           # Parted partition information
sudo blkid               # Block device attributes
```

Partition Management Tools

1. fdisk (MBR and GPT)

bash

```
sudo fdisk /dev/sda      # Interactive partitioning
# Commands within fdisk:
# p - print partition table
# n - new partition
# d - delete partition
# t - change partition type
# w - write changes
# q - quit without saving
```

2. parted (More Advanced)

bash

```
sudo parted /dev/sda     # Interactive mode
sudo parted /dev/sda print  # Print partition table
sudo parted /dev/sda mklabel gpt  # Create GPT partition table
sudo parted /dev/sda mkpart primary ext4 1MiB 100GiB
```

3. gdisk (GPT Specialist)

```bash
sudo gdisk /dev/sda              # GPT partitioning tool
```

Creating Filesystems

1. Common Filesystem Types

```bash
sudo mkfs.ext4 /dev/sda1          # Create ext4 filesystem
sudo mkfs.xfs /dev/sda2           # Create XFS filesystem
sudo mkfs.ntfs /dev/sda3          # Create NTFS filesystem
sudo mkfs.fat -F32 /dev/sda4      # Create FAT32 filesystem
```

2. Filesystem Options

```bash
sudo mkfs.ext4 -L "MyData" /dev/sda1  # Add label
sudo mkfs.ext4 -m 1 /dev/sda1         # Reserve 1% for root
sudo mkfs.ext4 -T largefile /dev/sda1 # Optimize for large files
```

Mounting and Unmounting

1. Manual Mounting

```bash
sudo mkdir /mnt/mydisk            # Create mount point
sudo mount /dev/sda1 /mnt/mydisk      # Mount partition
sudo mount -t ext4 /dev/sda1 /mnt/mydisk  # Specify filesystem
```

2. Unmounting

```bash
sudo umount /mnt/mydisk           # Unmount by mount point
sudo umount /dev/sda1             # Unmount by device
sudo umount -l /mnt/mydisk        # Lazy unmount
```

3. Permanent Mounting (/etc/fstab)

```bash
# Edit /etc/fstab
sudo nano /etc/fstab

# Add line:
/dev/sda1 /mnt/mydisk ext4 defaults 0 2

# Test fstab entry:
sudo mount -a                    # Mount all fstab entries
```

Advanced Partition Operations

1. Resizing Partitions

```bash
# Resize ext4 filesystem
sudo resize2fs /dev/sda1         # Expand to partition size
sudo resize2fs /dev/sda1 50G     # Resize to specific size

# Resize XFS filesystem
sudo xfs_growfs /mnt/mydisk      # Expand XFS filesystem
```

2. Checking Filesystems

```bash
sudo fsck /dev/sda1              # Check filesystem
sudo fsck.ext4 -f /dev/sda1      # Force check ext4
sudo e2fsck -p /dev/sda1         # Automatic repair
```

3. Partition Information

```bash
sudo tune2fs -l /dev/sda1        # ext4 filesystem info
sudo xfs_info /dev/sda1          # XFS filesystem info
sudo file -s /dev/sda1           # Filesystem type
```

LVM (Logical Volume Management)

1. LVM Concepts

- **Physical Volume (PV)**: Physical disk or partition
- **Volume Group (VG)**: Collection of physical volumes

- **Logical Volume (LV)**: Virtual partition from volume group

2. LVM Commands

bash

```
# Create physical volume
sudo pvcreate /dev/sda1

# Create volume group
sudo vgcreate myvg /dev/sda1

# Create logical volume
sudo lvcreate -L 10G -n mylv myvg

# Display information
sudo pvdisplay          # Physical volumes
sudo vgdisplay          # Volume groups
sudo lvdisplay          # Logical volumes
```

Disk Monitoring and Maintenance

1. Monitor Disk Usage

bash

```
watch df -h             # Real-time disk usage
iostat -x 1             # I/O statistics
iotop                   # I/O usage by process
```

2. Disk Health

bash

```
sudo smartctl -a /dev/sda       # SMART disk health
sudo badblocks -v /dev/sda1     # Check for bad blocks
```

Practical Examples and Combinations

Example 1: System Log Analysis

bash

```
# Find all error messages from last week, count occurrences
find /var/log -name "*.log" -mtime -7 -exec grep -l "ERROR" {} \; | \
xargs grep "ERROR" | cut -d: -f3 | sort | uniq -c | sort -nr
```

Example 2: User Management Audit

bash

```
# Find all files owned by specific user and change ownership
find /home -user olduser -exec chown newuser:newgroup {} \;

# List users with shell access
grep -E "(/bin/bash|/bin/sh|/bin/zsh)$" /etc/passwd | cut -d: -f1
```

Example 3: Disk Cleanup Script

bash

```
# Find large files and show disk usage
find /home -type f -size +100M -exec ls -lh {} \; | \
awk '{print $5 " " $9}' | sort -hr | head -20

# Clean old log files
find /var/log -name "*.log" -mtime +30 -exec gzip {} \;
```

Best Practices and Tips

Security Best Practices

1. **Principle of Least Privilege**: Give users minimum necessary permissions
2. **Regular Audits**: Check user accounts and permissions regularly
3. **Strong Passwords**: Enforce password policies
4. **Sudo Usage**: Use sudo instead of root login
5. **File Permissions**: Set appropriate permissions on sensitive files

Performance Tips

1. **Use Piping**: Chain commands efficiently instead of temporary files
2. **Grep Optimization**: Use `-F` for fixed strings, `-E` for regex
3. **Find Optimization**: Use `-path` and `-prune` to exclude directories
4. **Disk Management**: Regular filesystem checks and cleanup

Common Mistakes to Avoid

1. **Recursive Permission Changes**: Be careful with `chmod -R`
2. **Wrong Partition Type**: Choose appropriate filesystem for use case
3. **Forgetting Backups**: Always backup before major changes

4. **Ignoring Disk Space**: Monitor disk usage regularly

This reference guide covers the essential Linux commands and concepts for system administration and daily use. Keep this handy for quick reference and continue practicing these commands to master Linux system management.