Complete Linux Administration Lesson Plan

Course Overview

This course is designed for beginners to learn essential Linux system administration skills. Topics are ordered to build upon previous knowledge, starting with fundamental concepts and progressing to advanced automation.

Duration: 7 sessions (2-3 hours each) **Prerequisites**: Basic computer literacy **Learning Objectives**: Master core Linux administration tasks including user management, permissions, networking, and automation

---

Lesson 1: System Information and Process Management

Foundation: Understanding your Linux system

Learning Objectives

- Navigate the Linux system and gather system information
- Monitor and manage running processes
- Understand system resources and performance

1.1 System Information Commands

Hardware and System Info

```
# Display system information
uname -a                         # Complete system information
hostnamectl                      # System hostname and OS info
lscpu                           # CPU information
free -h                         # Memory usage (human readable)
df -h                           # Disk space usage
lsblk                           # Block devices (disks/partitions)
lsusb                           # USB devices
lspci                           # PCI devices
```

System Status and Uptime

```
uptime                          # System uptime and load
date                            # Current date and time
whoami                          # Current username
id                              # User and group IDs
w                               # Who is logged in and what they're doing
last                            # Login history
```

1.2 Process Management

Viewing Processes

```
ps                           # Current user's processes
ps aux                       # All processes (detailed)
ps -ef                       # All processes (alternative format)
top                          # Real-time process monitor
htop                         # Enhanced process monitor (if available)
pstree                       # Process tree view
```

Process Control

```
# Background and foreground jobs
command &                    # Run command in background
jobs                         # List active jobs
fg %1                        # Bring job 1 to foreground
bg %1                        # Send job 1 to background
nohup command &              # Run command immune to hangups

# Process termination
kill PID                     # Terminate process by ID
kill -9 PID                 # Force kill process
killall process_name         # Kill all processes by name
pkill pattern               # Kill processes matching pattern
```

1.3 System Monitoring

```
# Resource monitoring
iostat                       # I/O statistics
```

```
vmstat                          # Virtual memory statistics
netstat -tuln                  # Network connections
ss -tuln                       # Modern replacement for netstat
dmesg                          # Kernel messages
journalctl                    # Systemd logs
```

Practical Exercise 1

1. Check your system's CPU, memory, and disk information
2. Start a long-running process (`sleep 300 &`)
3. Monitor it with `ps` and `top`
4. Practice killing processes safely

Lesson 2: User Management

Building on system knowledge to manage users

Learning Objectives

- Create, modify, and delete user accounts
- Understand user account files and settings
- Manage user passwords and account policies

2.1 Understanding User Accounts

User Account Files

```
# Key files for user management
/etc/passwd                # User account information
/etc/shadow               # Encrypted passwords and aging info
/etc/group                # Group information
/etc/gshadow              # Group password information
/etc/login.defs           # Default settings for user accounts
/etc/skel/                # Template directory for new users
```

Viewing User Information

```
cat /etc/passwd            # View all user accounts
getent passwd username     # Get specific user info
id username               # User and group IDs
finger username           # User information (if available)
```

2.2 User Account Management

Creating Users

```
# Basic user creation
useradd username               # Create user with defaults
useradd -m username           # Create user with home directory
useradd -m -s /bin/bash username  # Specify shell

# Advanced user creation
useradd -m -g primarygroup -G group1,group2 -s /bin/bash -c "Full Name" username
useradd -m -d /custom/home/path -u 1500 username  # Custom home and UID

# Set password
passwd username               # Set/change password
```

Modifying Users

```
usermod -c "New Full Name" username     # Change comment/full name
usermod -s /bin/zsh username           # Change shell
usermod -d /new/home -m username       # Change home directory
usermod -g newgroup username           # Change primary group
usermod -G group1,group2 username      # Set supplementary groups
usermod -a -G newgroup username        # Add to supplementary group
usermod -L username                    # Lock account
usermod -U username                    # Unlock account
```

Deleting Users

```
userdel username              # Delete user (keep home directory)
userdel -r username           # Delete user and home directory
userdel -f username           # Force deletion
```

2.3 Password Management

```
# Password policies
chage -l username            # View password aging info
chage -M 90 username         # Set max password age (90 days)
chage -m 7 username          # Set minimum password age
chage -W 7 username          # Set warning days before expiration
chage -E 2024-12-31 username # Set account expiration date

# Force password change
passwd -e username           # Expire password (force change at next login)
passwd -l username           # Lock password
passwd -u username           # Unlock password
```

Practical Exercise 2

1. Create three test users with different home directories
2. Set passwords and modify user properties
3. Practice locking/unlocking accounts
4. Set password aging policies

Lesson 3: Group Management

Extending user management with group concepts

Learning Objectives

- Create and manage groups
- Add/remove users from groups
- Understand group permissions and relationships

3.1 Understanding Groups

Group Information

```
cat /etc/group              # View all groups
getent group groupname      # Get specific group info
groups username            # Show user's groups
id -Gn username            # List group names for user
```

3.2 Group Management Commands

Creating Groups

```
groupadd groupname         # Create group
groupadd -g 1500 groupname # Create group with specific GID
groupadd -r systemgroup    # Create system group
```

Modifying Groups

```
groupmod -n newname oldname     # Rename group
groupmod -g 1600 groupname     # Change GID
gpasswd -a username groupname  # Add user to group
gpasswd -d username groupname  # Remove user from group
gpasswd -A admin1,admin2 groupname  # Set group administrators
```

Deleting Groups

```
groupdel groupname          # Delete group
```

3.3 Special Group Operations

```
# Group passwords (rarely used)
gpasswd groupname           # Set group password
newgrp groupname           # Switch to group (temporarily)
sg groupname command       # Execute command as group member

# Administrative commands
vigr                       # Edit /etc/group safely
vigr -s                    # Edit /etc/gshadow safely
```

Practical Exercise 3

1. Create department groups (IT, HR, Finance)
2. Add users to multiple groups
3. Practice group modifications
4. Verify group memberships

Lesson 4: File and Directory Permissions

Building on user/group knowledge for access control

Learning Objectives

- Understand Linux permission system
- Modify file and directory permissions
- Manage file ownership
- Use advanced permission features

4.1 Understanding Permissions

Permission Types

- **Read (r/4)**: View file contents or list directory
- **Write (w/2)**: Modify file or create/delete files in directory
- **Execute (x/1)**: Run file or access directory

Permission Levels

- **Owner (u)**: File/directory owner
- **Group (g)**: Primary group members
- **Others (o)**: Everyone else

Reading Permissions

```
ls -l filename              # Long listing with permissions
ls -ld directory           # Directory permissions
stat filename              # Detailed file information
```

4.2 Changing Permissions

Symbolic Method

```
chmod u+r filename         # Add read for owner
chmod g-w filename         # Remove write for group
chmod o+x filename         # Add execute for others
chmod a+r filename         # Add read for all (a = all)
chmod u+rwx,g+rx,o+r filename  # Multiple changes

# Common symbolic combinations
chmod +x script.sh         # Make executable for all
chmod -x filename          # Remove execute for all
chmod u=rwx,g=rx,o=r filename  # Set exact permissions
```

Numeric (Octal) Method

```
chmod 755 filename         # rwxr-xr-x
chmod 644 filename         # rw-r--r--
chmod 600 filename         # rw-------
chmod 777 filename         # rwxrwxrwx (avoid this!)
chmod 000 filename         # --------- (no permissions)

# Common permission combinations
chmod 755 directory/       # Standard directory permissions
chmod 644 file.txt         # Standard file permissions
chmod 600 private_file     # Owner read/write only
chmod 711 /home/username   # Home directory permissions
```

Recursive Permission Changes

```
chmod -R 755 directory/    # Change directory and all contents
find /path -type f -exec chmod 644 {} \;    # Files to 644
find /path -type d -exec chmod 755 {} \;    # Directories to 755
```

4.3 Ownership Management

Changing Ownership

```
chown user filename        # Change owner
chown user:group filename  # Change owner and group
chown :group filename      # Change group only
chown -R user:group directory/  # Recursive ownership change
```

Group Ownership

```
chgrp group filename        # Change group ownership
chgrp -R group directory/   # Recursive group change
```

4.4 Advanced Permissions

Special Permissions

```
# Setuid (4): Execute as file owner
chmod 4755 filename         # rwsr-xr-x
chmod u+s filename          # Add setuid

# Setgid (2): Execute as group owner / inherit group
chmod 2755 directory        # rwxr-sr-x
chmod g+s directory         # Add setgid

# Sticky bit (1): Only owner can delete files
chmod 1755 directory        # rwxr-xr-t
chmod +t directory          # Add sticky bit

# Combined special permissions
```

```
chmod 6755 filename         # setuid + setgid
```

Default Permissions (umask)

```
umask                       # View current umask
umask 022                   # Set umask (files: 644, dirs: 755)
umask 077                   # Restrictive umask (files: 600, dirs: 700)
```

Access Control Lists (ACLs)

```
# Check if ACLs are supported
getfacl filename            # View ACLs

# Set ACLs
setfacl -m u:username:rwx filename     # User permissions
setfacl -m g:groupname:rx filename     # Group permissions
setfacl -m o::r filename               # Others permissions
setfacl -x u:username filename         # Remove user ACL
setfacl -b filename                    # Remove all ACLs
```

Practical Exercise 4

1. Create files and directories with various permissions
2. Practice both symbolic and numeric permission changes

3. Set up shared directories with proper group permissions
4. Experiment with special permissions and ACLs

Lesson 5: Networking Commands

System administration networking essentials

Learning Objectives

- Understand network configuration and status
- Use network troubleshooting tools
- Manage network services and connections
- Monitor network traffic and performance

5.1 Network Configuration

Viewing Network Information

```
# Network interfaces
ip addr show                # Show all interfaces (modern)
ifconfig                    # Show interfaces (traditional)
ip link show               # Show network devices
ip route show              # Show routing table
```

```
# Hostname and DNS
hostname                      # Display hostname
hostname -f                  # Display FQDN
hostname -i                  # Display IP address
cat /etc/hostname            # Hostname file
cat /etc/hosts              # Local hostname resolution
cat /etc/resolv.conf        # DNS configuration
```

Network Interface Management

```
# Enable/disable interfaces
ip link set eth0 up          # Bring interface up
ip link set eth0 down        # Bring interface down
ifup eth0                    # Bring interface up (Debian/Ubuntu)
ifdown eth0                  # Bring interface down (Debian/Ubuntu)

# Assign IP addresses
ip addr add 192.168.1.100/24 dev eth0    # Add IP address
ip addr del 192.168.1.100/24 dev eth0    # Remove IP address
```

5.2 Network Connectivity Testing

Basic Connectivity

```
# Ping tests
ping google.com              # Test connectivity
ping -c 4 8.8.8.8         # Ping 4 times
ping6 ipv6.google.com     # IPv6 ping

# Traceroute
traceroute google.com      # Trace route to destination
tracepath google.com       # Alternative traceroute
mtr google.com            # Real-time traceroute
```

Port and Service Testing

```
# Port connectivity
telnet hostname 80         # Test port connectivity
nc -zv hostname 22         # Test SSH port with netcat
nmap -p 22,80,443 hostname    # Port scan specific ports
nmap -sP 192.168.1.0/24  # Ping scan network range

# Service testing
curl -I http://website.com     # HTTP headers
wget --spider http://site.com  # Test download without saving
```

5.3 Network Monitoring and Analysis

Connection Monitoring

```
# Active connections
netstat -tuln              # List listening ports
netstat -tun                 # Show established connections
ss -tuln                     # Modern replacement for netstat
ss -tulp                     # Show process names
lsof -i                      # List open network files
lsof -i :80                 # List processes using port 80
```

Network Traffic Analysis

```
# Traffic monitoring
iftop                        # Real-time traffic by connection
nethogs                      # Traffic by process
iotop                        # I/O monitoring
tcpdump -i eth0              # Packet capture
tcpdump -i eth0 port 80      # Capture HTTP traffic
wireshark                    # GUI packet analyzer (if available)
```

Bandwidth and Performance

```
# Bandwidth testing
iperf3 -s                    # Start iperf server
iperf3 -c server_ip          # Connect to iperf server
speedtest-cli                # Internet speed test

# Network statistics
cat /proc/net/dev            # Interface statistics
ip -s link show             # Interface statistics
```

5.4 DNS and Name Resolution

DNS Tools

```
# DNS queries
nslookup google.com       # DNS lookup (traditional)
dig google.com           # DNS lookup (modern)
dig @8.8.8.8 google.com  # Query specific DNS server
dig -x 8.8.8.8           # Reverse DNS lookup
host google.com          # Simple DNS lookup

# DNS record types
dig google.com MX        # Mail exchange records
dig google.com NS        # Name server records
dig google.com TXT       # Text records
dig google.com AAAA      # IPv6 records
```

5.5 Network Services Management

Service Control (systemd)

```
# Common network services
systemctl status NetworkManager    # Network manager status
systemctl restart networking      # Restart networking
systemctl status ssh              # SSH service status
systemctl enable ssh              # Enable SSH at boot
systemctl start firewalld         # Start firewall service
```

Firewall Management

```
# UFW (Ubuntu/Debian)
ufw status                # Show firewall status
ufw enable               # Enable firewall
ufw allow ssh            # Allow SSH
ufw allow 80/tcp         # Allow HTTP
ufw deny from 192.168.1.10    # Block specific IP

# iptables (direct)
iptables -L              # List rules
iptables -A INPUT -p tcp --dport 22 -j ACCEPT    # Allow SSH
```

Practical Exercise 5

1. Check network configuration and connectivity
2. Test connections to various services and ports
3. Monitor network traffic and connections
4. Practice DNS lookups and troubleshooting
5. Configure basic firewall rules

Lesson 6: Bash Scripting Fundamentals

Automation foundation using previous concepts

Learning Objectives

- Write basic bash scripts
- Use variables, conditionals, and loops
- Handle command-line arguments and user input
- Apply scripting to system administration tasks

6.1 Script Basics

Creating and Running Scripts

```
#!/bin/bash               # Shebang line
# This is a comment

echo "Hello World"        # Output text

# Make script executable
chmod +x script.sh
./script.sh              # Run script
bash script.sh           # Alternative execution
```

Script Structure Template

```bash
#!/bin/bash
# Script: script_name.sh
# Purpose: Brief description
# Author: Your Name
# Date: 2024-01-01

# Exit on any error
set -e

# Main script content here
echo "Script starting..."

# Exit successfully
exit 0
```

6.2 Variables and Data Types

Variable Declaration and Usage

```bash
#!/bin/bash

# Variable assignment (no spaces around =)
name="John"
age=25
current_date=$(date)
user_count=$(who | wc -l)
```

```
# Using variables
echo "Hello $name"
echo "You are $age years old"
echo "Today is $current_date"
echo "Users logged in: $user_count"

# Read-only variables
readonly PI=3.14159
declare -r SCRIPT_NAME="backup.sh"

# Environment variables
export MY_VAR="available to child processes"
```

Special Variables

```
#!/bin/bash

echo "Script name: $0"
echo "First argument: $1"
echo "Second argument: $2"
echo "All arguments: $@"
echo "Number of arguments: $#"
echo "Exit status of last command: $?"
echo "Process ID: $$"
```

6.3 User Input and Command Line Arguments

Reading User Input

```bash
#!/bin/bash

# Simple input
echo "Enter your name:"
read name
echo "Hello $name"

# Input with prompt
read -p "Enter your age: " age
echo "You are $age years old"

# Silent input (passwords)
read -s -p "Enter password: " password
echo -e "\nPassword entered"

# Input with timeout
read -t 10 -p "Enter something (10 seconds): " input
```

Processing Arguments

```bash
#!/bin/bash

# Check argument count
if [ $# -eq 0 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi
```

```
filename=$1

# Process all arguments
for arg in "$@"; do
    echo "Processing: $arg"
done

# Using getopts for options
while getopts "hvf:" option; do
    case $option in
        h) echo "Help message"; exit 0;;
        v) verbose=true;;
        f) filename="$OPTARG";;
        *) echo "Invalid option"; exit 1;;
    esac
done
```

6.4 Conditionals and Comparisons

If Statements

```
#!/bin/bash

# Basic if statement
if [ -f "/etc/passwd" ]; then
    echo "Password file exists"
fi
```

```bash
# If-else
if [ $# -gt 0 ]; then
    echo "Arguments provided: $@"
else
    echo "No arguments provided"
fi

# If-elif-else
if [ "$1" = "start" ]; then
    echo "Starting service"
elif [ "$1" = "stop" ]; then
    echo "Stopping service"
elif [ "$1" = "restart" ]; then
    echo "Restarting service"
else
    echo "Usage: $0 {start|stop|restart}"
fi
```

Test Conditions

```bash
#!/bin/bash

# File tests
[ -f file.txt ]          # File exists and is regular file
[ -d directory ]         # Directory exists
[ -r file.txt ]          # File is readable
[ -w file.txt ]          # File is writable
[ -x script.sh ]         # File is executable
[ -s file.txt ]           # File exists and is not empty

# String comparisons
[ "$str1" = "$str2" ]     # Strings are equal
[ "$str1" != "$str2" ]    # Strings are not equal
[ -z "$string" ]          # String is empty
[ -n "$string" ]          # String is not empty

# Numeric comparisons
[ $num1 -eq $num2 ]       # Equal
[ $num1 -ne $num2 ]       # Not equal
[ $num1 -lt $num2 ]       # Less than
[ $num1 -gt $num2 ]       # Greater than
[ $num1 -le $num2 ]       # Less than or equal
[ $num1 -ge $num2 ]       # Greater than or equal
```

6.5 Loops

For Loops

```
#!/bin/bash

# Loop over list
for user in alice bob charlie; do
    echo "Creating user: $user"
    # useradd "$user"
done
```

```
# Loop over files
for file in *.txt; do
    echo "Processing $file"
    # Process file
done

# Loop over command output
for user in $(cut -d: -f1 /etc/passwd); do
    echo "User: $user"
done

# C-style for loop
for ((i=1; i<=10; i++)); do
    echo "Number: $i"
done
```

While Loops

```
#!/bin/bash

# Counter-based while loop
counter=1
while [ $counter -le 5 ]; do
    echo "Count: $counter"
    counter=$((counter + 1))
done

# Reading file line by line
while IFS= read -r line; do
```

```
    echo "Line: $line"
done < input.txt

# Menu system
while true; do
    echo "1. List files"
    echo "2. Show date"
    echo "3. Exit"
    read -p "Choose option: " choice

    case $choice in
        1) ls -la;;
        2) date;;
        3) exit 0;;
        *) echo "Invalid option";;
    esac
done
```

6.6 Functions

Function Definition and Usage

```
#!/bin/bash

# Function definition
backup_file() {
    local source_file=$1
    local backup_dir="/backup"
    if [ ! -d "$backup_dir" ]; then
        mkdir -p "$backup_dir"
    fi

    cp "$source_file" "$backup_dir/"
    echo "Backed up $source_file to $backup_dir"
}

# Function with return value
is_user_exists() {
    local username=$1
    if id "$username" &>/dev/null; then
        return 0  # User exists
    else
        return 1  # User doesn't exist
    fi
}

# Using functions
backup_file "/etc/hosts"

if is_user_exists "apache"; then
    echo "Apache user exists"
else
    echo "Apache user not found"
fi
```

6.7 Practical Script Examples

System Information Script

```bash
#!/bin/bash
# System information gathering script

echo "=== System Information Report ==="
echo "Date: $(date)"
echo "Hostname: $(hostname)"
echo "Uptime: $(uptime)"
echo "Disk Usage:"
df -h
echo -e "\nMemory Usage:"
free -h
echo -e "\nTop 5 Processes:"
ps aux --sort=-%cpu | head -6
```

User Management Script

```bash
#!/bin/bash
# Simple user management script

create_user() {
    local username=$1
    local fullname=$2

    if id "$username" &>/dev/null; then
        echo "User $username already exists"
        return 1
    fi
    useradd -m -c "$fullname" -s /bin/bash "$username"
    passwd "$username"
    echo "User $username created successfully"
}

# Usage
if [ $# -ne 2 ]; then
    echo "Usage: $0 <username> <fullname>"
    exit 1
fi

create_user "$1" "$2"
```

Log Analysis Script

```
#!/bin/bash
# Simple log analysis script

logfile="/var/log/auth.log"

if [ ! -f "$logfile" ]; then
    echo "Log file $logfile not found"
    exit 1
fi

echo "=== Failed Login Attempts ==="
grep "Failed password" "$logfile" | tail -10
```

```
echo -e "\n=== Successful Logins ==="
grep "Accepted password" "$logfile" | tail -5

echo -e "\n=== Most Active IPs ==="
grep "Failed password" "$logfile" | \
    grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}' | \
    sort | uniq -c | sort -nr | head -5
```

Practical Exercise 6

1. Write a system monitoring script that checks disk space and sends alerts
2. Create a backup script that archives files older than 30 days
3. Build a user creation script with input validation
4. Write a log rotation script

Lesson 7: Scheduling Tasks with Cron

Automation using all previous knowledge

Learning Objectives

- Understand cron system and syntax
- Create and manage scheduled tasks

- Use at command for one-time scheduling
- Implement system maintenance automation

7.1 Understanding Cron

Cron System Components

```
# Cron daemon
systemctl status cron     # Check cron service (Debian/Ubuntu)
systemctl status crond    # Check cron service (RHEL/CentOS)

# Cron directories
/etc/crontab             # System-wide crontab
/etc/cron.d/             # System cron job files
/etc/cron.hourly/        # Hourly scripts
/etc/cron.daily/         # Daily scripts
/etc/cron.weekly/        # Weekly scripts
/etc/cron.monthly/       # Monthly scripts
/var/spool/cron/crontabs/  # User crontabs
```

Cron Log Files

```
# View cron logs
tail -f /var/log/cron     # RHEL/CentOS
tail -f /var/log/syslog | grep cron  # Debian/Ubuntu
journalctl -u cron        # Systemd systems
```

7.2 Crontab Syntax and Format

Crontab Fields

```
* * * * * command
| | | | |
| | | | └─── Day of week (0-7, Sunday=0 or 7)
| | | └───── Month (1-12)
| | └─────── Day of month (1-31)
| └───────── Hour (0-23)
└─────────── Minute (0-59)
```

Special Characters

```
*     # Any value (wildcard)
,     # Value list separator (1,3,5)
-     # Range of values (1-5)
/     # Step values (*/5 = every 5)
```

Special Time Strings

```
@yearly   # Once a year (0 0 1 1 *)
@monthly  # Once a month (0 0 1 * *)
@weekly   # Once a week (0 0 * * 0)
@daily    # Once a day (0 0 * * *)
@hourly   # Once an hour (0 * * * *)
```

```
@reboot   # At startup
```

7.3 Managing User Crontabs

Crontab Commands

```
crontab -l              # List current user's crontab
crontab -e              # Edit current user's crontab
crontab -r              # Remove current user's crontab
crontab -u username -l  # List another user's crontab (as root)
crontab -u username -e  # Edit another user's crontab (as root)
```

Environment Variables in Crontab

```
# Set environment variables at the top of crontab
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=admin@example.com
HOME=/home/user

# Then add your cron jobs
0 2 * * * /home/user/scripts/backup.sh
```

7.4 Common Cron Job Examples

Basic Scheduling Examples

```
# Every minute
* * * * * /path/to/script.sh

# Every 5 minutes
*/5 * * * * /path/to/script.sh

# Every hour at 30 minutes past
30 * * * * /path/to/script.sh

# Daily at 2:30 AM
30 2 * * * /path/to/script.sh

# Weekly on Sunday at 3 AM
0 3 * * 0 /path/to/script.sh

# Monthly on the 1st at midnight
0 0 1 * * /path/to/script.sh

# Weekdays only at 9 AM
0 9 * * 1-5 /path/to/script.sh

# Multiple times per day
0 6,12,18 * * * /path/to/script.sh
```

System Administration Tasks

```
# System backup (daily at 2 AM)
0 2 * * * /usr/local/bin/backup.sh > /var/log/backup.log 2>&1

# Log rotation (weekly)
0 3 * * 0 /usr/sbin/logrotate /etc/logrotate.conf

# Disk usage report (daily)
0 8 * * * df -h | mail -s "Disk Usage Report" admin@company.com

# Update system (weekly, Sunday 4 AM)
0 4 * * 0 /usr/bin/apt update && /usr/bin/apt upgrade -y

# Clear temp files (daily)
30 1 * * * /usr/bin/find /tmp -type f -mtime +7 -delete

# Database backup (daily at 3 AM)
0 3 * * * /usr/local/bin/db_backup.sh
```

7.5 Advanced Cron Techniques

Redirecting Output

```
# Redirect output and errors
0 2 * * * /path/to/script.sh > /var/log/script.log 2>&1

# Send output via email (if MAILTO is set)
0 2 * * * /path/to/script.sh
```

```
# Discard all output
0 2 * * * /path/to/script.sh > /dev/null 2>&1

# Log with timestamp
0 2 * * * echo "$(date): Running backup" >> /var/log/cron_jobs.log; /path/to/backup.sh
```

Conditional Execution

```
# Run only if file exists
0 2 * * * [ -f /path/to/flag.txt ] && /path/to/script.sh

# Run with file locking (prevent overlap)
*/5 * * * * /usr/bin/flock -n /var/lock/script.lock /path/to/script.sh

# Chain commands
0 2 * * * /path/to/backup.sh && /path/to/cleanup.sh || echo "Backup failed" | mail admin@company.com
```

7.6 System Cron Jobs and Configuration

System-wide Crontab (/etc/crontab)

```
# Example /etc/crontab structure
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
```

```
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

Creating System Cron Jobs in /etc/cron.d/

```
# Create custom system cron job file
sudo nano /etc/cron.d/custom-backup

# Content example:
# System backup job
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=admin@company.com

# Daily backup at 2 AM
0 2 * * * root /usr/local/bin/system_backup.sh

# Weekly log cleanup on Sunday at 1 AM
0 1 * * 0 root /usr/local/bin/log_cleanup.sh
```

Managing System Cron Directories

```
# Scripts in these directories run automatically
ls -la /etc/cron.hourly/
ls -la /etc/cron.daily/
ls -la /etc/cron.weekly/
ls -la /etc/cron.monthly/

# Create executable scripts
sudo nano /etc/cron.daily/backup
chmod +x /etc/cron.daily/backup

# Test run-parts command
sudo run-parts --test /etc/cron.daily/
sudo run-parts /etc/cron.daily/
```

7.7 Alternative Scheduling: at and anacron

Using ‘at’ for One-time Jobs

```
# Schedule one-time job
at 2:30 AM tomorrow
at> /usr/local/bin/maintenance.sh
at> <Ctrl+D>

# Alternative formats
at 14:30 today
at 9:00 AM Jul 31
at now + 1 hour
at midnight

# List scheduled jobs
atq                   # List jobs
```

```
at -l                  # Alternative listing

# Remove scheduled job
atrm job_number        # Remove specific job
at -r job_number       # Alternative removal

# View job details
at -c job_number       # Show job commands

# Batch processing (run when system load is low)
batch
batch> /usr/local/bin/heavy_process.sh
batch> <Ctrl+D>
```

Understanding anacron

```
# anacron configuration
cat /etc/anacrontab

# Example anacrontab format:
# period delay job-identifier command
1       5       cron.daily      nice run-parts /etc/cron.daily
7       25      cron.weekly     nice run-parts /etc/cron.weekly
30      45      cron.monthly    nice run-parts /etc/cron.monthly

# anacron commands
anacron -T             # Test configuration
anacron -n             # Run jobs now (ignore timestamps)
anacron -u              # Update timestamps without running
```

7.8 Practical Cron Job Scripts

System Backup Script

```
#!/bin/bash
# File: /usr/local/bin/system_backup.sh
# Purpose: Daily system backup
# Cron: 0 2 * * * /usr/local/bin/system_backup.sh

# Configuration
BACKUP_DIR="/backup/$(date +%Y%m%d)"
RETENTION_DAYS=30
LOG_FILE="/var/log/backup.log"

# Function to log messages
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S'): $1" | tee -a "$LOG_FILE"
}

# Create backup directory
mkdir -p "$BACKUP_DIR"

log_message "Starting system backup to $BACKUP_DIR"

# Backup important directories
tar -czf "$BACKUP_DIR/etc_backup.tar.gz" /etc/ 2>/dev/null
tar -czf "$BACKUP_DIR/home_backup.tar.gz" /home/ 2>/dev/null
tar -czf "$BACKUP_DIR/var_log_backup.tar.gz" /var/log/ 2>/dev/null

# Database backup (if applicable)
if command -v mysqldump &> /dev/null; then
    mysqldump --all-databases > "$BACKUP_DIR/mysql_backup.sql" 2>/dev/null
    log_message "MySQL databases backed up"
fi

# Cleanup old backups
find /backup/ -type d -name "????????" -mtime +$RETENTION_DAYS -exec rm -rf {} + 2>/dev/null

log_message "Backup completed successfully"

# Send notification email (if mail is configured)
if command -v mail &> /dev/null; then
    echo "System backup completed successfully on $(hostname)" | \
    mail -s "Backup Report - $(date +%Y-%m-%d)" admin@company.com
fi
```

System Monitoring Script

```
#!/bin/bash
# File: /usr/local/bin/system_monitor.sh
# Purpose: Monitor system resources and alert if thresholds exceeded
# Cron: */10 * * * * /usr/local/bin/system_monitor.sh

# Configuration
DISK_THRESHOLD=90
```

```
MEMORY_THRESHOLD=90
CPU_THRESHOLD=90
LOAD_THRESHOLD=5.0
ALERT_EMAIL="admin@company.com"

# Check disk usage
check_disk() {
    for usage in $(df -h | awk 'NR>1 {print $5}' | sed 's/%//'); do
        if [ "$usage" -gt "$DISK_THRESHOLD" ]; then
            echo "ALERT: Disk usage is ${usage}% (threshold: ${DISK_THRESHOLD}%)"
            return 1
        fi
    done
    return 0
}

# Check memory usage
check_memory() {
    memory_usage=$(free | awk '/Mem:/ {printf "%.0f", $3/$2 * 100}')
    if [ "$memory_usage" -gt "$MEMORY_THRESHOLD" ]; then
        echo "ALERT: Memory usage is ${memory_usage}% (threshold: ${MEMORY_THRESHOLD}%)"
        return 1
    fi
    return 0
}

# Check CPU load
check_load() {
    load_1min=$(uptime | awk -F'load average:' '{print $2}' | awk '{print $1}' | sed 's/,//')
    if (( $(echo "$load_1min > $LOAD_THRESHOLD" | bc -l) )); then
        echo "ALERT: Load average is $load_1min (threshold: $LOAD_THRESHOLD)"
        return 1
    fi
    return 0
}

# Main monitoring
alerts=""
check_disk || alerts="${alerts}$(check_disk)\n"
check_memory || alerts="${alerts}$(check_memory)\n"
check_load || alerts="${alerts}$(check_load)\n"

# Send alerts if any issues found
if [ ! -z "$alerts" ]; then
    echo -e "System Monitoring Alert - $(hostname)\n\n$alerts" | \
    mail -s "System Alert - $(date)" "$ALERT_EMAIL"
fi
```

Log Cleanup Script

```
#!/bin/bash
# File: /usr/local/bin/log_cleanup.sh
# Purpose: Clean up old log files to save disk space
# Cron: 0 3 * * 0 /usr/local/bin/log_cleanup.sh

# Configuration
LOG_RETENTION_DAYS=30
ARCHIVE_DIR="/var/log/archived"

# Create archive directory
```

```
mkdir -p "$ARCHIVE_DIR"

# Find and compress old logs
find /var/log -name "*.log" -mtime +7 -not -path "/var/log/archived/*" -exec gzip {} \;

# Move very old compressed logs to archive
find /var/log -name "*.gz" -mtime +$LOG_RETENTION_DAYS -not -path "/var/log/archived/*" -exec mv {} "$ARCHIVE_DIR/"

# Clean up archived logs older than retention period
find "$ARCHIVE_DIR" -name "*.gz" -mtime +$LOG_RETENTION_DAYS -delete

# Truncate large active logs if needed
for log in /var/log/messages /var/log/syslog /var/log/daemon.log; do
    if [ -f "$log" ] && [ $(stat -f%z "$log" 2>/dev/null || stat -c%s "$log") -gt 104857600 ]; then
        # If log > 100MB, truncate to last 10000 lines
        tail -10000 "$log" > "${log}.tmp" && mv "${log}.tmp" "$log"
    fi
done

echo "$(date): Log cleanup completed" >> /var/log/log_cleanup.log
```

7.9 Cron Security and Best Practices

Security Considerations

```
# Control who can use cron
echo "alloweduser" >> /etc/cron.allow     # Only listed users can use cron
echo "denieduser" >> /etc/cron.deny       # Listed users cannot use cron
```

```
# Check cron permissions
ls -la /etc/cron*
ls -la /var/spool/cron/

# Secure cron directories
chmod 700 /var/spool/cron/crontabs/
chmod 600 /var/spool/cron/crontabs/*
```

Best Practices

```
# 1. Always use full paths in cron jobs
0 2 * * * /usr/bin/rsync -av /home/ /backup/

# 2. Set environment variables explicitly
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
HOME=/root
SHELL=/bin/bash

# 3. Handle output appropriately
0 2 * * * /usr/local/bin/backup.sh >> /var/log/backup.log 2>&1

# 4. Use locking to prevent job overlap
*/5 * * * * /usr/bin/flock -n /var/lock/myjob.lock /usr/local/bin/myjob.sh

# 5. Test scripts before scheduling
# Run script manually first
/usr/local/bin/backup.sh
```

```
# 6. Monitor cron job execution
# Check logs regularly
tail -f /var/log/cron

# 7. Use meaningful names for scripts in /etc/cron.d/
# Good: /etc/cron.d/database-backup
# Bad: /etc/cron.d/job1
```

7.10 Troubleshooting Cron Jobs

Common Issues and Solutions

```
# 1. Environment differences
# Test environment in cron context
* * * * * env > /tmp/cron-env.txt

# Compare with interactive environment
env > /tmp/interactive-env.txt
diff /tmp/cron-env.txt /tmp/interactive-env.txt

# 2. Path issues - use full paths or set PATH
# Wrong:
0 2 * * * backup.sh

# Correct:
PATH=/usr/local/bin:/usr/bin:/bin
0 2 * * * /usr/local/bin/backup.sh
```

```
# 3. Permission issues
# Check script permissions
ls -la /usr/local/bin/backup.sh
chmod +x /usr/local/bin/backup.sh

# 4. Cron daemon issues
# Check if cron is running
systemctl status cron
systemctl start cron

# 5. Syntax errors in crontab
# Test crontab syntax
crontab -T  # Some systems support this

# 6. Missing mail system
# Install mail system to see cron output
apt-get install mailutils  # Debian/Ubuntu
yum install mailx          # RHEL/CentOS
```

Debugging Techniques

```
# 1. Add debugging to scripts
#!/bin/bash
set -x  # Enable debug mode
exec 2>> /var/log/script-debug.log  # Log errors

# 2. Test cron jobs frequently first
# Set to run every minute for testing
* * * * * /usr/local/bin/test-script.sh
```

```
# Then change to desired schedule
0 2 * * * /usr/local/bin/test-script.sh

# 3. Use logger command for debugging
0 2 * * * logger "Cron job started"; /usr/local/bin/script.sh; logger "Cron job finished"

# 4. Capture all output for debugging
0 2 * * * /usr/local/bin/script.sh > /tmp/cronlog 2>&1
```

Practical Exercise 7

1. Create a system monitoring script that checks disk space, memory, and CPU usage
2. Set up automated backups running daily at 2 AM
3. Implement log rotation for custom application logs
4. Create a script that emails weekly system reports
5. Set up one-time maintenance tasks using the `at` command
6. Practice troubleshooting cron jobs by intentionally creating issues and fixing them

---

Course Summary and Final Project

What You've Learned

Throughout this course, you've mastered:

- System information gathering and process management
- Complete user and group administration
- File permissions and access control
- Network configuration and troubleshooting
- Bash scripting for automation
- Task scheduling and system automation

Final Project: Complete System Administration Script

Create a comprehensive system administration script that incorporates all learned concepts:

Requirements:

1. **System Information Module**: Gather and display system stats
2. **User Management Module**: Create/modify/delete users with proper permissions
3. **Security Module**: Set file permissions and perform security checks
4. **Backup Module**: Automated backup with rotation
5. **Monitoring Module**: Check system health and send alerts
6. **Scheduling**: Set up the script to run via cron
7. **Logging**: Comprehensive logging of all operations
8. **Error Handling**: Robust error handling and recovery

Next Steps for Advanced Learning

- System service management with systemd
- Advanced networking and firewalls
- Container technologies (Docker, Podman)
- Configuration management (Ansible, Puppet)
- System performance tuning
- Security hardening and compliance
- Advanced shell scripting and automation

Additional Resources

- Linux man pages (`man command`)
- Distribution-specific documentation
- Online Linux communities and forums
- Practice environments (VirtualBox, VMware, cloud instances)
- Professional certifications (LPIC, Red Hat, CompTIA Linux+)

Remember: The best way to learn Linux is through hands-on practice. Set up a test environment and experiment with these commands and concepts regularly!