SESSION 1 & SESSION 2 – LINUX FUNDAMENTALS & SYSTEM OPERATIONS

---

1. INTRODUCTION TO LINUX

1.1 Definition

Linux is an **open-source, Unix-like operating system kernel** developed by **Linus Torvalds** in 1991. Combined with GNU utilities, it forms a complete operating system.

1.2 History of Linux

- **1969** – UNIX developed at Bell Labs
- **1983** – GNU Project by Richard Stallman
- **1991** – Linux Kernel released by Linus Torvalds
- **1992** – GPL License adoption
- **Present** – Used in servers, cloud, mobile (Android), supercomputers

---

1.3 Why Linux?

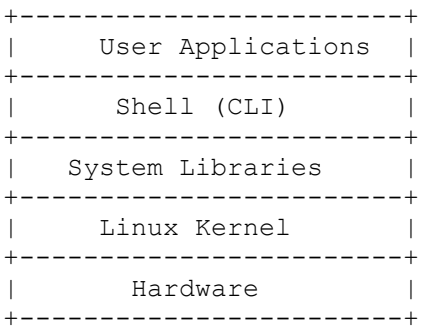
- Open Source
- Secure & Stable
- Multi-user & Multitasking
- High Performance
- Strong Networking
- Automation-friendly

1.4 Linux Distributions

Category	Examples
Desktop	Ubuntu, Fedora
Server	RHEL, CentOS, Rocky
Security	Kali Linux
Embedded	Yocto

---

2. LINUX ARCHITECTURE



Kernel Responsibilities

- Process management
- Memory management
- File system management
- Device drivers
- Network stack
- Security enforcement

---

3. LINUX FILE SYSTEM

3.1 Definition

Linux uses a **hierarchical tree structure** starting from `/` (root).

3.2 Filesystem Hierarchy (FHS)

```
/
├── bin      → Essential binaries
├── boot     → Bootloader files
├── dev      → Device files
├── etc      → Configuration files
├── home     → User home directories
├── lib      → Shared libraries
├── media    → Removable media
├── mnt      → Temporary mounts
├── opt      → Optional software
├── proc     → Process info (virtual FS)
├── root     → Root user home
├── sbin     → System binaries
├── tmp      → Temporary files
├── usr      → User programs
```

```
└── var      → Logs, spool files
```

3.3 Types of Files

- Regular files (`-`)
- Directory (`d`)
- Symbolic link (`l`)
- Block device (`b`)
- Character device (`c`)
- Socket (`s`)
- Pipe (`p`)

---

4. WORKING WITH FILES AND DIRECTORIES

4.1 Navigation Commands

pwd

```
pwd
```

➡ Shows current working directory

cd

```
cd /etc
cd ..
cd ~
```

4.2 Listing Files

ls

```
ls
ls -l
ls -a
ls -lh
```

Options explained:

- `-l` → long listing
- `-a` → hidden files
- `-h` → human readable size

4.3 File Operations

touch

```
touch file1.txt
```

cp

```
cp file1 file2
cp -r dir1 dir2
```

mv

```
mv old new
```

rm

```
rm file
rm -r dir
```

⚠ Dangerous with `-rf`

4.4 Directory Commands

mkdir

```
mkdir test
mkdir -p a/b/c
```

rmdir

```
rmdir emptydir
```

---

5. VIEWING & PROCESSING FILE CONTENT

5.1 cat / tac

```
cat file
tac file
```

5.2 more / less

```
more file
less file
```

5.3 head / tail

```
head -n 5 file
tail -f logfile
```

5.4 wc

```
wc file
wc -l file
```

5.5 sort

```
sort file
sort -r file
```

5.6 grep

```
grep "error" file
grep -i root /etc/passwd
```

5.7 diff

```
diff file1 file2
```

5.8 file

```
file filename
```

---

6. SEARCHING FILES

find

```
find / -name file.txt
find /var -size +100M
```

locate

```
locate passwd
updatedb
```

7. COMPRESSION & ARCHIVING

gzip / gunzip

```
gzip file
gunzip file.gz
```

zip / unzip

```
zip a.zip file
unzip a.zip
```

tar

```
tar -cvf backup.tar dir
tar -xvf backup.tar
tar -czvf backup.tar.gz dir
```

---

8. SYSTEM & USER COMMANDS

who / w / whoami

```
who
w
whoami
```

finger

```
finger username
```

---

date / cal

```
date
cal
```

---

bc

```
bc
bc -l
```

---

alias / unalias

```
alias ll='ls -l'
unalias ll
```

---

9. PERMISSIONS & OWNERSHIP

9.1 Permission Model

```
-rwxr-xr--
│ │ │ │ │ │
│ │ │ │ │ └─ Others
│ │ │ │ └─── Group
│ │ │ └───── Owner
│ └─────────── File type
```

---

chmod

```
chmod 755 file
chmod u+x file
```

---

chown

```
chown user file
chown user:group file
```

---

10. ACCESS CONTROL LISTS (ACL)

Purpose

Fine-grained permissions beyond chmod.

Commands

```
getfacl file
setfacl -m u:john:rwx file
setfacl -x u:john file
```

---

11. NETWORK COMMANDS

telnet

```
telnet host port
```

⚠ Insecure (plaintext)

---

ftp

```
ftp server
```

⚠ Insecure

---

ssh

```
ssh user@server
```

✔ Encrypted

---

sftp

```
sftp user@server
put file
get file
```

---

finger

```
finger user
```

---

12. SECONDARY STORAGE DEVICES

Types

- HDD
- SSD
- CD/DVD
- USB

Device Files

```
/dev/sda
```

```
/dev/sda1
/dev/cdrom
```

---

lsblk

```
lsblk
```

---

mount

```
mount /dev/sdb1 /mnt
```

umount

```
umount /mnt
```

---

13. DISK FORMATTING IN LINUX

File Systems

- ext4
- xfs
- vfat
- ntfs

---

mkfs

```
mkfs.ext4 /dev/sdb1
mkfs.vfat /dev/sdb1
```

---

fdisk

```
fdisk /dev/sdb
n → new partition
w → write
```

---

14. LAB PERSPECTIVE (MANDATORY)

Lab 1

- Use `pwd`, `ls`, `cd`
- Create directories and files

Lab 2

- Use `grep`, `sort`, `wc`
- Analyze log files

Lab 3

- Set permissions using `chmod`
- Change ownership using `chown`

Lab 4

- Mount USB device
- Format disk safely

---

15. SECURITY CONSIDERATIONS

- Avoid `chmod 777`
- Use SSH instead of telnet/ftp
- Limit root usage
- Use ACLs carefully
- Always verify device before formatting

---

16. EXAM & INTERVIEW POINTS

- Linux follows **everything is a file**
- Root (`/`) is top of filesystem
- Permissions: **r=4, w=2, x=1**
- ACL overrides traditional permissions
- `tar` is archiver, not compressor
- `find` is real-time, `locate` uses DB

---

17. MINI CASE STUDY

Scenario:
A company server requires secure file sharing.

Solution:

- Use `ssh` & `sftp`
- Set ACL for specific users
- Mount external backup drive
- Use `tar` + `gzip` for backups

SESSION 3 & SESSION 4 – LINUX INSTALLATION, BOOT PROCESS & PACKAGE MANAGEMENT

---

1. LINUX INSTALLATION

1.1 Definition

Linux installation is the process of **deploying a Linux operating system** onto hardware or virtual machines, including disk partitioning, filesystem creation, bootloader setup, and package selection.

1.2 Pre-Installation Requirements

Hardware Requirements

- CPU: x86_64 / ARM
- RAM: Minimum 1–2 GB (Server: 4+ GB)
- Disk Space: 20 GB minimum
- BIOS/UEFI support
- Network (for updates)

Installation Media

- ISO image
- DVD / USB

- PXE (network boot)

1.3 Installation Types

Type	Description
Graphical	GUI-based installer
Text Mode	CLI-based installer
Automated	Kickstart / Preseed
Network Install	PXE-based

1.4 General Linux Installation Workflow

```
Power ON
   ↓
BIOS / UEFI
   ↓
Boot Loader
   ↓
Kernel + initramfs
   ↓
Installer (Anaconda / Debian Installer)
   ↓
Disk Partitioning
   ↓
Package Selection
   ↓
Configuration (users, network)
   ↓
Bootloader Installation
   ↓
Reboot → Linux System
```

---

2. ANACONDA INSTALLER (INTERACTIVE INSTALLATION)

2.1 What is Anaconda?

Anaconda is the **default graphical and text-mode installer** for:

- RHEL

- CentOS
- Rocky Linux
- Fedora

2.2 Features

- GUI & Text mode
- LVM support
- RAID support
- Kickstart automation
- Network configuration
- SELinux configuration

2.3 Anaconda Interactive Installation Steps

Step-by-Step Workflow

```
Boot from ISO
→ Select Install Linux
→ Language Selection
→ Keyboard Layout
→ Installation Source
→ Software Selection
→ Installation Destination
→ Network & Hostname
→ Time & Date
→ Root Password
→ User Creation
→ Begin Installation
```

Disk Partitioning Options

- Automatic partitioning
- Manual partitioning
- LVM-based partitioning

Common Partitions

Mount Point	Purpose
/	Root filesystem

Mount Point	Purpose
/boot	Boot files
/home	User data
swap	Virtual memory

---

3. HANDS-FREE INSTALLATION METHODS

3.1 Need for Automated Installation

- Large-scale deployments
- Consistency
- Faster provisioning
- Reduced human error

3.2 Kickstart Installation (RedHat-based)

Definition

Kickstart is a **script-based automated installation mechanism**.

Kickstart Workflow

```
PXE Boot / ISO
→ Kickstart File (ks.cfg)
→ Automatic Disk Partitioning
→ Automatic Package Installation
→ Automatic Configuration
→ Reboot
```

Kickstart File Example

```
install
url --url=http://server/repo
lang en_US
keyboard us
rootpw redhat
timezone Asia/Kolkata
clearpart --all
```

```
autopart
%packages
@core
vim
net-tools
%end
```

3.3 Debian Preseed (Brief)

- Used in Ubuntu/Debian
- Similar to Kickstart
- Automated responses to installer questions

---

4. LINUX BOOT PROCEDURE

4.1 Boot Process Overview

Linux booting is a **multi-stage controlled process** that loads the OS into memory.

4.2 Boot Sequence (Detailed)

```
Power ON
  ↓
BIOS / UEFI
  ↓
MBR / GPT
  ↓
Boot Loader (GRUB)
  ↓
Kernel
  ↓
initramfs
  ↓
systemd (PID 1)
  ↓
Target (Runlevel)
  ↓
Login Prompt / GUI
```

4.3 BIOS vs UEFI

BIOS	UEFI
Legacy	Modern
MBR	GPT
2TB limit	>2TB
Slower	Faster

---

5. GRUB BOOT LOADER CONFIGURATION

5.1 What is GRUB?

GRUB (GRand Unified Bootloader) loads:

- Linux kernel
- initramfs
- Boot parameters

5.2 GRUB Versions

- GRUB Legacy
- GRUB2 (modern)

5.3 GRUB Files

File	Purpose
/etc/default/grub	User configuration
/etc/grub.d/	Scripts
/boot/grub2/grub.cfg	Final config (DO NOT edit)

5.4 GRUB Configuration Example

```
GRUB_TIMEOUT=5
GRUB_DEFAULT=saved
GRUB_CMDLINE_LINUX="rhgb quiet"
```

Regenerate GRUB

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

(UEFI)

```
grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

5.5 GRUB Recovery (Single User Mode)

- Edit kernel parameters
- Add `rd.break`
- Reset root password

---

6. INITIAL RAM DISK (initrd / initramfs)

6.1 Definition

initramfs is a **temporary root filesystem loaded into RAM** during boot.

6.2 Purpose

- Load drivers
- Mount real root filesystem
- Support LVM, RAID, encryption

6.3 Boot with initramfs

```
Kernel
→ initramfs
→ Load drivers
→ Mount /
→ switch_root
```

6.4 initramfs Files

```
/boot/initramfs-$(uname -r).img
```

Rebuild initramfs

```
dracut -f
```

---

7. RUN LEVELS / TARGETS

7.1 Traditional Runlevels (SysV)

Runlevel	Meaning
0	Halt
1	Single-user
3	Multi-user (CLI)
5	GUI
6	Reboot

7.2 systemd Targets

Target	Equivalent
poweroff.target	Runlevel 0
rescue.target	Runlevel 1
multi-user.target	Runlevel 3
graphical.target	Runlevel 5
reboot.target	Runlevel 6

Check Current Target

```
systemctl get-default
```

Set Default Target

```
systemctl set-default multi-user.target
```

---

8. REPOSITORY & PACKAGE MANAGEMENT

8.1 Package Management Concepts

- Binary packages
- Dependency resolution
- Version control
- Secure updates (GPG)

8.2 RPM Package Management

RPM Commands

```
rpm -ivh package.rpm
rpm -Uvh package.rpm
rpm -e package
rpm -qa
rpm -qi package
rpm -ql package
```

Limitations of RPM

- No dependency resolution

8.3 Yellowdog Updater (YUM)

Definition

YUM is a **high-level package manager** that handles dependencies automatically.

YUM Commands

```
yum install httpd
yum remove httpd
yum update
yum search nginx
yum list installed
```

8.4 DNF (Modern YUM Replacement)

```
dnf install vim
dnf remove vim
dnf upgrade
dnf repolist
```

8.5 YUM Repository Configuration

```
/etc/yum.repos.d/base.repo
```

Example:

```
[base]
name=Base Repo
baseurl=http://mirror/repo
enabled=1
gpgcheck=1
```

8.6 DEB Package Management

---

dpkg

```
dpkg -i package.deb
dpkg -r package
dpkg -l
```

APT

```
apt update
apt install apache2
apt remove apache2
apt upgrade
```

---

9. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Installation

- Install Linux using Anaconda
- Create custom partitions

Lab 2 – GRUB

- Modify GRUB timeout
- Regenerate grub.cfg

Lab 3 – Boot Troubleshooting

- Enter rescue mode
- Reset root password

Lab 4 – Package Management

- Install packages using YUM/DNF
- Configure local repository
- Remove packages safely

---

10. SECURITY CONSIDERATIONS

- Enable GPG checks
- Restrict GRUB editing with password
- Secure automated installations
- Limit root access
- Use trusted repositories only

---

11. PERFORMANCE & ADMIN TIPS

- Use minimal installation for servers
- Keep initramfs clean
- Remove unused repositories
- Automate installations using Kickstart

---

12. EXAM & INTERVIEW POINTS

- GRUB loads kernel + initramfs
- initramfs is temporary RAM filesystem
- systemd replaces SysV init
- RPM ≠ dependency resolver
- YUM/DNF handles dependencies
- Kickstart enables hands-free installation

---

13. MINI CASE STUDY

Scenario:
Deploy 100 Linux servers in a data center.

Solution:

- PXE + Kickstart
- Central YUM repository
- Standard partitioning
- systemd multi-user target
- Automated security baseline

SESSION 5 – SYSTEM CONTROL, AUTOMATED DEPLOYMENT & USER ADMINISTRATION

---

1. SHUTDOWN AND REBOOT CONCEPTS

1.1 Why Controlled Shutdown Is Important

- Prevents filesystem corruption
- Ensures safe termination of processes
- Protects hardware and data
- Maintains service integrity

1.2 System States

State	Meaning
Running	Normal operation
Halt	Power off
Reboot	Restart system
Rescue	Maintenance mode

1.3 Shutdown Commands (systemd-based)

shutdown

```
shutdown now
shutdown -h now
shutdown -r now
shutdown +10 "System maintenance"
```

Explanation (line-by-line):

- `shutdown` → Sends signal to systemd
- `-h` → Halt
- `-r` → Reboot
- `+10` → Delay (minutes)
- Message → Broadcast to logged-in users

---

poweroff

```
poweroff
```

➡ Powers off the system immediately

---

reboot

```
reboot
```

➡ Restarts the system cleanly

---

halt

```
halt
```

⏎ Stops CPU execution (may not power off)

1.4 systemctl Targets

```
systemctl poweroff
systemctl reboot
systemctl rescue
```

1.5 Traditional Commands (Backward Compatibility)

```
init 0
init 6
```

1.6 Forced Reboot (NOT recommended)

```
reboot -f
echo b > /proc/sysrq-trigger
```

⚠ Use only in emergency

---

2. KICKSTART CONFIGURATION & CUSTOMIZATION

2.1 What Is Kickstart?

Kickstart is a **Red Hat–based automated OS installation mechanism** using a predefined configuration file.

2.2 Why Kickstart?

- Hands-free installation
- Large-scale deployments
- Consistency
- Automation and DevOps pipelines

2.3 Kickstart File Structure

```
ks.cfg
├── Command Section
├── %packages
├── %pre
├── %post
```

2.4 Kickstart Commands Section

Basic Configuration

```
install
lang en_US.UTF-8
keyboard us
timezone Asia/Kolkata
rootpw redhat
reboot
```

Installation Source

```
url --url=http://server/repo
```

Disk Partitioning

```
clearpart --all --initlabel
autopart
```

OR manual:

```
part /boot --fstype ext4 --size 1024
part swap --size 2048
part / --fstype ext4 --grow
```

Network Configuration

```
network --bootproto=dhcp --device=eth0 --onboot=on
```

SELinux & Firewall

```
selinux --enforcing
```

```
firewall --enabled --ssh
```

---

3. DEPLOYMENT USING KICKSTART

3.1 Kickstart Deployment Methods

Method	Description
ISO-based	Local media
PXE-based	Network boot
HTTP/NFS	Remote config

3.2 PXE + Kickstart Workflow

```
Client Boot
↓
DHCP
↓
PXE Server
↓
TFTP (bootloader)
↓
Kernel + initramfs
↓
Kickstart File
↓
Automated Installation
↓
Reboot → Ready Server
```

3.3 Specifying Kickstart at Boot

```
linux inst.ks=http://server/ks.cfg
```

3.4 %packages Section

```
%packages
@core
vim
net-tools
httpd
%end
```

3.5 %pre Section (Before Installation)

```
%pre
echo "Pre-install script running" > /tmp/pre.log
%end
```

3.6 %post Section (After Installation)

```
%post
useradd devops
echo redhat | passwd --stdin devops
%end
```

---

4. USER ADMINISTRATION

4.1 User Concepts

- Linux is multi-user
- Each user has:
  - UID
  - GID
  - Home directory
  - Login shell

System Files

File	Purpose
/etc/passwd	User details
/etc/shadow	Encrypted passwords
/etc/group	Group info

4.2 Types of Users

Type	UID Range
Root	0
System	1–999
Normal	≥1000

4.3 User Creation

useradd

```
useradd john
useradd -m -s /bin/bash alice
```

Explanation:

- `-m` → Create home directory
- `-s` → Login shell

---

passwd

```
passwd john
```

4.4 User Modification

usermod

```
usermod -aG wheel john
usermod -l newname oldname
```

4.5 User Deletion

userdel

```
userdel john
userdel -r john
```

4.6 Group Administration

groupadd

```
groupadd dev
```

groupmod

```
groupmod -n developers dev
```

groupdel

```
groupdel developers
```

4.7 Switching Users

```
su - john
sudo command
```

4.8 Sudo Configuration

```
visudo
```

Example:

```
john ALL=(ALL) ALL
```

---

5. LAB PERSPECTIVE (MANDATORY)

Lab 1 – System Control

- Schedule shutdown
- Switch to rescue mode
- Test reboot commands

Lab 2 – Kickstart

- Create ks.cfg
- Automate user creation
- Deploy OS using PXE

Lab 3 – User Management

- Create users/groups
- Assign sudo privileges
- Verify /etc/passwd & /etc/shadow

---

6. SECURITY CONSIDERATIONS

- Restrict shutdown access
- Protect Kickstart files
- Encrypt passwords
- Minimize sudo privileges
- Disable unused users

---

7. PERFORMANCE & AUTOMATION TIPS

- Use Kickstart for CI/CD pipelines
- Use %post scripts wisely
- Automate security hardening
- Use minimal packages

---

8. EXAM & INTERVIEW POINTS

- Kickstart = unattended installation
- `%pre` runs before disk partitioning
- `%post` runs after OS install
- systemd uses targets, not runlevels
- UID 0 is root
- `useradd` ≠ `adduser`

---

9. MINI CASE STUDY

Scenario:
Automated deployment of Linux servers with standard users.

Solution:

- PXE + Kickstart
- Create users in `%post`
- Enforce SELinux
- Enable firewall
- Auto reboot

SESSION 6 – LINUX NETWORKING, REMOTE ACCESS & AUTHENTICATION

---

1. IPv4 AND IPv6 ADDRESSING

1.1 Introduction to IP Addressing

An IP address is a **logical numeric identifier** assigned to a network interface that enables **communication between devices** over a network.

1.2 IPv4 Addressing

1.2.1 Definition

IPv4 uses **32-bit addressing**, written in **dotted decimal notation**.

Example:

```
192.168.1.10
```

1.2.2 IPv4 Structure

```
| Network Part | Host Part |
```

Each IPv4 address consists of:

- Network ID
- Host ID

1.2.3 IPv4 Address Classes

Class	Range	Default Mask
A	1.0.0.0 – 126.0.0.0	255.0.0.0
B	128.0.0.0 – 191.255.0.0	255.255.0.0
C	192.0.0.0 – 223.255.255.0	255.255.255.0
D	Multicast	N/A
E	Experimental	N/A

1.2.4 Private IPv4 Addresses

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

1.2.5 Subnet Mask & CIDR

```
192.168.1.10/24
```

- `/24` → Network bits
- Enables subnetting and efficient IP allocation

1.2.6 IPv4 Configuration Files

RHEL / CentOS

```
/etc/sysconfig/network-scripts/ifcfg-eth0
```

Example:

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.1.10
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
ONBOOT=yes
```

Ubuntu / Debian

```
/etc/netplan/01-netcfg.yaml
```

1.3 IPv6 Addressing

1.3.1 Definition

IPv6 uses **128-bit addressing**, written in **hexadecimal** and separated by colons.

Example:

```
2001:db8:abcd:0012::1
```

1.3.2 IPv6 Features

- Huge address space
- No NAT required
- Built-in IPsec support
- Auto-configuration (SLAAC)

1.3.3 IPv6 Address Types

Type	Purpose
Unicast	One-to-one
Multicast	One-to-many
Anycast	Nearest node

1.3.4 IPv6 Configuration Example

```
ip -6 addr add 2001:db8::10/64 dev eth0
```

1.3.5 IPv6 Configuration Files (RHEL)

```
/etc/sysconfig/network-scripts/ifcfg-eth0
IPV6INIT=yes
IPV6ADDR=2001:db8::10/64
```

---

2. OPENSSH

2.1 What is OpenSSH?

OpenSSH is an **open-source implementation of the SSH protocol** providing **secure remote login, command execution, and file transfer**.

2.2 Why SSH?

- Encrypted communication
- Secure remote access
- Key-based authentication
- Port forwarding

2.3 SSH Architecture

```
SSH Client
   ↓ (Encrypted Channel)
SSH Server (sshd)
```

2.4 SSH Components

- `ssh` – remote login
- `sshd` – SSH daemon
- `scp` – secure copy
- `sftp` – secure FTP

2.5 SSH Configuration Files

Server Config

`/etc/ssh/sshd_config`

Key parameters:

```
Port 22
PermitRootLogin no
PasswordAuthentication no
```

```
PubkeyAuthentication yes
```

Client Config

```
~/.ssh/config
```

2.6 SSH Commands (With Explanation)

ssh

```
ssh user@server
```

- `user` → Remote user
- `server` → Host/IP

---

scp

```
scp file user@server:/path
```

---

sftp

```
sftp user@server
put file
get file
```

2.7 SSH Key-Based Authentication

Key Generation

```
ssh-keygen
```

Creates:

- `id_rsa` (private key)
- `id_rsa.pub` (public key)

Copy Key

```
ssh-copy-id user@server
```

2.8 SSH Security Best Practices

- Disable root login
- Use key-based auth
- Change default port
- Use fail2ban
- Limit users with `AllowUsers`

---

3. VNC (VIRTUAL NETWORK COMPUTING)

3.1 What is VNC?

VNC provides **graphical remote desktop access** to Linux systems using the **RFB protocol**.

3.2 VNC Architecture

```
VNC Viewer (Client)
      ↓
VNC Server (X Desktop)
```

3.3 VNC Components

- VNC Server
- VNC Viewer
- X Window System

3.4 VNC Ports

```
Display :1 → TCP 5901
Display :2 → TCP 5902
```

3.5 Installing VNC Server

```
yum install tigervnc-server
```

3.6 Configuring VNC

Set VNC Password

```
vncpasswd
```

Start VNC Server

```
vncserver :1
```

VNC Configuration File

```
~/.vnc/xstartup
```

3.7 Securing VNC

- Use SSH tunneling
- Restrict firewall ports
- Use strong passwords
- Disable VNC when not needed

---

4. NETWORK AUTHENTICATION

4.1 What is Network Authentication?

Network authentication ensures **only authorized users and systems** can access network resources.

4.2 Authentication Factors

- Something you know (password)
- Something you have (key, token)
- Something you are (biometric)

4.3 Linux Authentication Architecture (PAM)

```
Application
   ↓
PAM
   ↓
Authentication Modules
```

4.4 PAM Configuration Files

`/etc/pam.d/`

Example:

`/etc/pam.d/sshd`

4.5 Centralized Authentication

LDAP

- Central user database
- Scalable

Kerberos

- Ticket-based authentication
- Used in Active Directory

4.6 Network Authentication Tools

- `sssd`
- `realmd`
- `authconfig`

4.7 SSH + PAM Integration

- PAM controls SSH access
- Account locking
- Password policies

---

5. LAB PERSPECTIVE (MANDATORY)

Lab 1 – IP Configuration

- Assign static IPv4
- Configure IPv6
- Test using `ping`

Lab 2 – SSH

- Configure sshd
- Enable key-based login
- Disable root login

Lab 3 – VNC

- Install VNC
- Configure remote desktop
- Secure with SSH tunnel

Lab 4 – Authentication

- Configure PAM rules
- Test login restrictions
- Centralized authentication demo

---

6. SECURITY CONSIDERATIONS

- Avoid plaintext protocols
- Use encryption always
- Restrict remote access
- Monitor login logs
- Apply firewall rules

---

7. EXAM & INTERVIEW POINTS

- IPv4 = 32-bit, IPv6 = 128-bit
- SSH replaces telnet
- VNC is GUI-based remote access
- PAM controls authentication
- SSH keys are more secure than passwords

---

8. MINI CASE STUDY

Scenario:
Secure remote access for Linux servers.

Solution:

- Static IP addressing
- SSH key-based authentication
- Disable VNC direct access
- PAM-based restrictions
- Firewall enforcement

SESSION 7 – USER MANAGEMENT & GROUP MANAGEMENT IN LINUX

---

1. INTRODUCTION TO USER & GROUP MANAGEMENT

1.1 Why User & Group Management Is Critical

Linux is a **multi-user operating system**. Proper user and group management ensures:

- System security
- Controlled access to files and services
- Accountability and auditing
- Least-privilege enforcement
- Compliance with enterprise policies

1.2 Identity in Linux

Each identity consists of:

- **User ID (UID)**
- **Group ID (GID)**
- **Home directory**
- **Login shell**
- **Authentication credentials**

---

2. USER MANAGEMENT

2.1 Types of Users

User Type	UID Range	Purpose
Root	0	Superuser
System Users	1–999	Services & daemons
Normal Users	≥1000	Human users

2.2 User Account Database Files

2.2.1 /etc/passwd

Purpose: Stores basic user account information.

```
cat /etc/passwd
```

Format:

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

Example:

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

Field	Meaning
username	Login name
x	Password placeholder
UID	User ID
GID	Primary group
comment	User info
home	Home directory
shell	Login shell

⚠ **Passwords are NOT stored here**

2.2.2 /etc/shadow

Purpose: Stores encrypted passwords and aging information.

```
sudo cat /etc/shadow
```

Format:

```
username:encrypted_password:last_change:min:max:warn:inactive:expire
```

Security:

- Readable only by root
- Protects against password theft

2.2.3 /etc/login.defs

Defines:

- UID ranges
- Password aging
- Default policies

2.3 User Lifecycle Management

2.3.1 User Creation

useradd

```
useradd john
```

Common options:

```
useradd -m -s /bin/bash -c "Dev User" john
```

Option	Description
-m	Create home directory
-s	Login shell
-c	Comment
-u	Custom UID

passwd

```
passwd john
```

Sets or changes password.

2.3.2 User Login

- Authentication via PAM
- Shell initialization
- Profile loading (`.bash_profile`, `.bashrc`)

2.3.3 User Modification

usermod

```
usermod -l newname oldname
usermod -aG wheel john
usermod -s /sbin/nologin guest
```

2.3.4 User Locking & Unlocking

```
passwd -l john
passwd -u john
```

OR

```
usermod -L john
usermod -U john
```

2.3.5 User Deletion

userdel

```
userdel john
userdel -r john
```

- `-r` → removes home directory

⚠ Orphan files may remain

2.4 Special User Concepts

2.4.1 Root User

- UID 0
- Full system privileges
- Should not be used for daily tasks

2.4.2 Sudo Users

sudo

```
sudo command
```

visudo

```
visudo
```

Example:

```
john ALL=(ALL) ALL
```

2.4.3 Nologin & False Shell

```
/sbin/nologin
/bin/false
```

Used for service accounts.

---

3. GROUP MANAGEMENT

3.1 Why Groups Exist

Groups allow:

- Shared file access
- Easier permission management
- Role-based access control (RBAC)

3.2 Group Database File

/etc/group

```
cat /etc/group
```

Format:

```
groupname:x:GID:user1,user2
```

Example:

```
developers:x:1002:john,alice
```

3.3 Types of Groups

Type	Description
Primary	Assigned at user creation
Secondary	Additional group memberships
System	Used by services

3.4 Group Lifecycle Management

3.4.1 Group Creation

groupadd

```
groupadd dev
groupadd -g 2000 admins
```

3.4.2 Group Modification

groupmod

```
groupmod -n developers dev
groupmod -g 3000 developers
```

3.4.3 Group Deletion

groupdel

```
groupdel developers
```

3.5 Managing Group Membership

Add User to Group

```
usermod -aG dev john
```

Check User Groups

```
groups john
id john
```

3.6 Primary vs Secondary Groups

- **Primary Group**
  - Files created by user belong to this group
- **Secondary Groups**
  - Additional access rights

---

4. FILE OWNERSHIP & GROUP RELATION

```
ls -l file
-rw-r----- 1 john dev file
```

- Owner → john
- Group → dev

Change Ownership

```
chown john file
chown john:dev file
```

Change Group

```
chgrp dev file
```

---

5. SECURITY IMPLICATIONS

5.1 Common Security Risks

- Shared root passwords
- Excessive sudo access
- Orphan users

- Weak password policies

5.2 Best Practices

- Use sudo instead of root
- Enforce password aging
- Lock unused accounts
- Separate service accounts
- Audit `/etc/passwd` regularly

---

6. LAB PERSPECTIVE (MANDATORY)

Lab 1 – User Lifecycle

- Create user
- Set password
- Modify shell
- Lock/unlock user
- Delete user

Lab 2 – Group Management

- Create group
- Add users
- Change group ownership
- Verify permissions

Lab 3 – Security Hardening

- Configure sudo
- Disable root SSH login
- Set password expiry

---

7. PERFORMANCE & ADMIN TIPS

- Use consistent UID/GID across servers
- Centralize users (LDAP/AD)
- Automate user creation with scripts
- Monitor login logs (`/var/log/secure`)

---

8. EXAM & INTERVIEW POINTS

- `/etc/passwd` ≠ passwords
- `/etc/shadow` stores encrypted passwords
- UID 0 = root
- `useradd` is low-level command
- `usermod -aG` appends groups
- Groups simplify permission management

---

9. MINI CASE STUDY

Scenario:
Secure multi-user Linux server for development team.

Solution:

- Individual user accounts
- Shared developer group
- Sudo for admins only
- Locked unused users
- Periodic audits

SESSION 8 – DISK MANAGEMENT, PARTITIONING & LVM

---

1. DISK MANAGEMENT IN LINUX

1.1 Definition

Disk management in Linux involves **detecting, partitioning, formatting, mounting, monitoring, and managing storage devices** to ensure data availability, performance, and reliability.

1.2 Why Disk Management Is Important

- Efficient use of storage
- Data organization
- Performance optimization
- Fault isolation
- Scalability (dynamic storage growth)
- Backup & recovery planning

1.3 Types of Storage Devices

Device Type	Example
HDD	/dev/sda
SSD	/dev/nvme0n1
USB	/dev/sdb
CD/DVD	/dev/sr0
Virtual Disk	/dev/vda

1.4 Linux Disk Naming Convention

```
/dev/sda      → First disk
/dev/sda1     → First partition
/dev/sdb2     → Second disk, second partition
/dev/nvme0n1p1 → NVMe partition
```

1.5 Disk & Block Device Commands

lsblk

```
lsblk
```

➡ Displays block devices and mount points

---

blkid

```
blkid
```

➥ Shows UUID and filesystem type

---

df

```
df -h
```

➥ Shows mounted filesystem usage

---

du

```
du -sh /var
```

➥ Shows directory disk usage

1.6 Disk Partitioning Concepts

Partition

A **logical division of a physical disk**.

Partition Tables

Type	Features
MBR	Max 2 TB, 4 primary partitions
GPT	>2 TB, 128 partitions, UEFI

---

2. FDISK (MBR PARTITIONING TOOL)

2.1 What is fdisk?

`fdisk` is a **command-line utility** used to manage **MBR partition tables**.

2.2 When to Use fdisk

- Legacy BIOS systems
- Disks ≤ 2 TB
- MBR-based partitioning

2.3 fdisk Workflow

```
Disk → Partition Table (MBR)
        ↓
Primary / Extended / Logical Partitions
```

2.4 Running fdisk

```
fdisk /dev/sdb
```

2.5 fdisk Interactive Commands

Command	Meaning
p	Print partition table
n	New partition
d	Delete partition
t	Change partition type
w	Write changes
q	Quit without saving

2.6 Creating a Partition Using fdisk (Step-by-Step)

```
fdisk /dev/sdb
```

1. `n` → New partition

2. `p` → Primary
3. Partition number → 1
4. First sector → default
5. Last sector → +5G
6. `w` → Write changes

2.7 Formatting fdisk Partition

```
mkfs.ext4 /dev/sdb1
```

2.8 Limitations of fdisk

- No GPT support
- No resizing
- Static partition sizes

---

3. GDISK (GPT PARTITIONING TOOL)

3.1 What is gdisk?

`gdisk` is a **GPT-aware partitioning tool**, often called **GPT fdisk**.

3.2 Why gdisk?

- Supports disks > 2 TB
- UEFI compatibility
- More partitions
- Better reliability

3.3 gdisk Disk Layout

```
Disk
├── GPT Header
```

```
├── Partition Entry Array
├── Partitions
└── Backup GPT
```

3.4 Running gdisk

```
gdisk /dev/sdc
```

3.5 gdisk Interactive Commands

Command	Description
p	Print partition table
n	New partition
d	Delete partition
t	Change partition type
w	Write changes
q	Quit

3.6 Creating GPT Partition

```
gdisk /dev/sdc
n
Partition number: 1
First sector: default
Last sector: +10G
w
```

3.7 Comparing fdisk vs gdisk

Feature	fdisk	gdisk
Partition Table	MBR	GPT
Disk Size	≤ 2 TB	> 2 TB
UEFI Support	No	Yes
Reliability	Moderate	High

---

4. LOGICAL VOLUME MANAGER (LVM)

4.1 What is LVM?

LVM is a **flexible storage management framework** that allows **dynamic resizing, pooling, and management of storage**.

4.2 Why LVM Is Needed

- Resize filesystems online
- Combine multiple disks
- Improve scalability
- Simplify storage administration

4.3 LVM Architecture

```
+----------------------+
| Logical Volume (LV)  |
+----------------------+
| Volume Group (VG)    |
+----------------------+
| Physical Volumes(PV) |
+----------------------+
| Disk / Partition     |
+----------------------+
```

4.4 LVM Components Explained

Physical Volume (PV)

- Disk or partition initialized for LVM

```
pvcreate /dev/sdb1
```

Volume Group (VG)

- Pool of storage created from PVs

```
vgcreate vg_data /dev/sdb1
```

Logical Volume (LV)

- Virtual partition carved from VG

```
lvcreate -L 5G -n lv_app vg_data
```

4.5 LVM Creation Workflow (End-to-End)

```
fdisk /dev/sdb
mkfs not required yet
pvcreate /dev/sdb1
vgcreate vg01 /dev/sdb1
lvcreate -L 10G -n lv01 vg01
mkfs.ext4 /dev/vg01/lv01
mount /dev/vg01/lv01 /mnt
```

4.6 Display LVM Information

```
pvs
vgs
lvs
```

4.7 Extending Logical Volume (Online)

Extend LV

```
lvextend -L +5G /dev/vg01/lv01
```

Resize Filesystem

```
resize2fs /dev/vg01/lv01
```

4.8 Reducing Logical Volume (Risky)

```
umount /mnt
e2fsck -f /dev/vg01/lv01
resize2fs /dev/vg01/lv01 5G
lvreduce -L 5G /dev/vg01/lv01
```

⚠ Data loss risk if done incorrectly

4.9 Removing LVM Components

```
lvremove /dev/vg01/lv01
vgremove vg01
pvremove /dev/sdb1
```

4.10 LVM Advantages & Disadvantages

Advantages

- Dynamic resizing
- Disk pooling
- Snapshot support
- Enterprise-ready

Disadvantages

- Slight overhead
- More complexity
- Boot partition cannot be LVM (older systems)

---

5. FILESYSTEM & MOUNTING WITH LVM

Mounting

```
mount /dev/vg01/lv01 /data
```

Persistent Mount (/etc/fstab)

```
/dev/vg01/lv01  /data  ext4  defaults  0 0
```

---

6. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Disk Discovery

- Use `lsblk`, `blkid`, `df`
- Identify free disks

Lab 2 – fdisk

- Create MBR partition
- Format and mount

Lab 3 – gdisk

- Create GPT partition
- Format with XFS

Lab 4 – LVM

- Create PV, VG, LV
- Mount filesystem
- Extend LV online

---

7. SECURITY & ADMINISTRATION CONSIDERATIONS

- Always backup before resizing
- Restrict disk tools to root
- Use UUIDs in `/etc/fstab`
- Monitor disk usage
- Use LVM for critical data

---

8. PERFORMANCE & BEST PRACTICES

- Align partitions properly
- Use LVM for servers
- Separate OS and data volumes
- Monitor I/O bottlenecks
- Plan capacity growth

---

9. EXAM & INTERVIEW POINTS

- fdisk = MBR only
- gdisk = GPT support
- LVM = dynamic storage
- PV → VG → LV
- `resize2fs` resizes filesystem
- LVM enables online expansion

---

10. MINI CASE STUDY

Scenario:
Database server storage is running out of space.

Solution:

- Add new disk
- Create PV
- Extend VG
- Extend LV online
- Resize filesystem with zero downtime

SESSION 9 – NETWORK IMPLEMENTATION & PRINT SERVICES (LINUX)

---

1. NETWORK IMPLEMENTATION IN LINUX

1.1 Definition

Network implementation in Linux refers to **configuring, managing, and maintaining network connectivity and services** so that systems can communicate securely and reliably over LAN, WAN, or the Internet.

1.2 Objectives of Network Implementation

- Enable host-to-host communication
- Provide network-based services
- Ensure security and access control
- Support scalability and reliability
- Enable centralized management

1.3 Linux Networking Architecture

```
+-----------------------------+
| Network Applications        |
| (SSH, FTP, HTTP, NFS, etc.) |
+-----------------------------+
| Socket Interface            |
+-----------------------------+
| TCP / UDP / ICMP            |
+-----------------------------+
| IP (IPv4 / IPv6)            |
+-----------------------------+
| Network Device Drivers      |
+-----------------------------+
| Network Interface Hardware  |
+-----------------------------+
```

1.4 Network Interface Naming

- `eth0`, `eth1` (legacy)
- `ens33`, `enp0s3` (predictable naming)

1.5 Network Configuration Files

RHEL / CentOS / Rocky

`/etc/sysconfig/network-scripts/ifcfg-<interface>`

Example:

```
DEVICE=ens33
BOOTPROTO=static
IPADDR=192.168.1.20
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS1=8.8.8.8
ONBOOT=yes
```

Ubuntu / Debian (Netplan)

```
/etc/netplan/01-netcfg.yaml
```

1.6 Network Management Tools

nmcli (NetworkManager CLI)

```
nmcli device status
nmcli con show
nmcli con up ens33
```

---

ip Command

```
ip addr show
ip link set ens33 up
ip route show
```

---

ifconfig (Legacy)

```
ifconfig ens33
```

1.7 Network Service Configuration

Hostname Configuration

```
hostnamectl set-hostname server1.example.com
```

DNS Configuration

```
/etc/resolv.conf
```

Example:

```
nameserver 8.8.8.8
nameserver 1.1.1.1
```

Testing Network Connectivity

```
ping google.com
traceroute google.com
```

```
netstat -tuln
ss -tuln
```

1.8 Network Services Commonly Implemented

Service	Purpose
SSH	Secure remote access
FTP/SFTP	File transfer
HTTP/HTTPS	Web services
NFS	Network file sharing
SMB/CIFS	Windows file sharing
DNS	Name resolution
DHCP	Dynamic IP assignment

1.9 Firewall Integration

firewalld

```
firewall-cmd --list-all
firewall-cmd --add-service=ssh --permanent
firewall-cmd --reload
```

1.10 Network Security Considerations

- Disable unused services
- Use encrypted protocols
- Restrict ports via firewall
- Use SELinux
- Monitor network logs

---

2. PRINT SERVICES IN LINUX

2.1 Introduction to Print Services

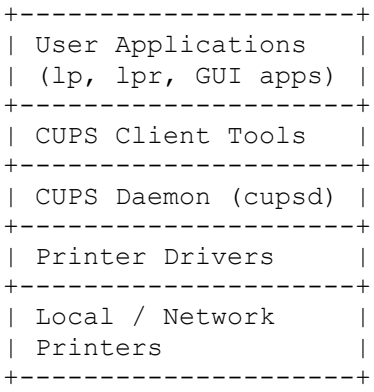
Print services allow **Linux systems to manage local and network printers**, handle print queues, and provide shared printing functionality.

2.2 CUPS (Common UNIX Printing System)

Definition

CUPS is the **standard printing system for Linux**, developed by Apple, based on **IPP (Internet Printing Protocol)**.

2.3 CUPS Architecture



2.4 Key CUPS Components

- `cupsd` – Print server daemon
- IPP – Printing protocol
- PPD – Printer description files
- Print queues

2.5 CUPS Configuration Files

Main Configuration

`/etc/cups/cupsd.conf`

Printer Definitions

`/etc/cups/printers.conf`

2.6 Installing CUPS

```
yum install cups
systemctl start cups
systemctl enable cups
```

2.7 Accessing CUPS Web Interface

```
http://localhost:631
```

2.8 Printer Management Commands

lpstat

```
lpstat -p
lpstat -t
```

---

lp

```
lp file.txt
```

---

lpr

```
lpr file.txt
```

---

cancel

```
cancel job-id
```

---

enable / disable Printer

```
cupsenable printer_name
cupsdisable printer_name
```

2.9 Adding Printers

Local Printer

- USB / Parallel
- Auto-detected by CUPS

Network Printer

- IPP
- LPD
- SMB
- JetDirect

Example URI:

```
ipp://printer-ip/ipp/print
```

2.10 Printer Sharing

Enable Sharing

```
cupsctl --share-printers
```

Firewall Rule

```
firewall-cmd --add-service=ipp --permanent
firewall-cmd --reload
```

2.11 Print Queue Management

- Jobs are queued
- Priorities can be set
- Jobs can be paused/resumed

2.12 Printer Drivers & PPD

- PPD defines printer capabilities
- Installed via manufacturer or generic drivers

---

3. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Network Implementation

- Configure static IP
- Set hostname
- Test connectivity
- Configure firewall rules

Lab 2 – Network Services

- Enable SSH
- Test remote login
- Monitor open ports

Lab 3 – Print Services

- Install CUPS
- Access web interface
- Add local printer
- Print test page

Lab 4 – Network Printing

- Add network printer
- Share printer
- Configure firewall for IPP

---

4. SECURITY CONSIDERATIONS

- Restrict printer access
- Secure CUPS web interface
- Avoid anonymous printing

- Encrypt network traffic
- Monitor print logs

---

5. PERFORMANCE & ADMINISTRATION TIPS

- Remove unused printers
- Limit print job size
- Monitor spool directory
- Centralize print servers
- Use driverless printing where possible

---

6. EXAM & INTERVIEW POINTS

- CUPS uses IPP protocol
- Port 631 is used by CUPS
- NetworkManager manages interfaces
- `ip` command replaces `ifconfig`
- Printing queues manage jobs
- Firewall must allow IPP for sharing

---

7. MINI CASE STUDY

Scenario:
Office with 50 users requires centralized printing.

Solution:

- Deploy Linux CUPS server
- Configure network printers
- Enable printer sharing
- Secure access via firewall
- Monitor print queues

SESSION 10 – SERVICES MANAGEMENT, SYSTEM CONFIGURATION FILES & NIS

1. SERVICES MANAGEMENT IN LINUX

1.1 Definition

Service management in Linux refers to **controlling background processes (daemons)** that provide system and network functionality such as SSH, HTTP, DNS, printing, logging, and authentication.

1.2 What is a Service (Daemon)?

A **service/daemon** is a background process that:

- Starts at boot or on demand
- Runs without user interaction
- Provides continuous functionality

Examples:

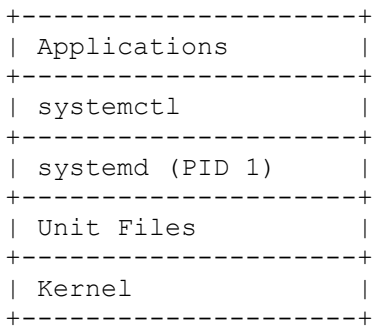
- `sshd` – Secure Shell
- `httpd` – Web server
- `cupsd` – Print service
- `crond` – Scheduler

1.3 init Systems (Evolution)

Init System	Description
SysV init	Traditional, sequential
Upstart	Event-based
systemd	Modern, parallel, dependency-based

➡ **Modern Linux uses systemd**

1.4 systemd Architecture



1.5 systemd Units

Unit Type	Extension	Purpose
Service	.service	Daemons
Target	.target	Runlevels
Mount	.mount	Filesystems
Socket	.socket	On-demand
Timer	.timer	Scheduling

1.6 Service Management Commands (systemctl)

Check Service Status

```
systemctl status sshd
```

Start / Stop / Restart Service

```
systemctl start httpd
systemctl stop httpd
systemctl restart httpd
```

Reload Configuration

```
systemctl reload sshd
```

Enable / Disable at Boot

```
systemctl enable httpd
systemctl disable httpd
```

List All Services

```
systemctl list-units --type=service
systemctl list-unit-files
```

1.7 Service Unit File Structure

Example:

```
/usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH server
After=network.target

[Service]
ExecStart=/usr/sbin/sshd -D
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

1.8 Legacy Service Commands (Backward Compatibility)

```
service sshd start
chkconfig sshd on
```

⚠ Deprecated but still available

1.9 Service Logs (journalctl)

```
journalctl -u sshd
journalctl -xe
journalctl --since "1 hour ago"
```

1.10 Service Security Best Practices

- Disable unused services
- Use least-privilege
- Monitor logs regularly

- Protect unit files
- Use SELinux

---

2. SYSTEM CONFIGURATION FILES

2.1 What Are System Configuration Files?

Text-based files (mostly in `/etc`) that control:

- System behavior
- Services
- Users
- Networking
- Security policies

2.2 Importance of /etc Directory

`/etc` contains **host-specific configuration files**.

```
/etc
├── passwd
├── shadow
├── group
├── hosts
├── hostname
├── resolv.conf
├── fstab
├── ssh/
├── cups/
└── sysconfig/
```

2.3 Critical Configuration Files (With Purpose)

/etc/passwd

User account information

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

/etc/shadow

Encrypted passwords and aging policies
Readable only by root

/etc/group

Group membership data

/etc/hostname

System hostname

/etc/hosts

Local hostname resolution

```
127.0.0.1 localhost
192.168.1.10 server1
```

/etc/resolv.conf

DNS configuration

```
nameserver 8.8.8.8
```

/etc/fstab

Persistent filesystem mounts

```
UUID=xxxx /data ext4 defaults 0 0
```

/etc/sysconfig/

RHEL-based system configuration files

- Network
- Services
- Daemons

/etc/ssh/

SSH configuration directory

- `sshd_config`
- `ssh_config`

2.4 Configuration File Best Practices

- Always backup before editing
- Use `vim` or `nano`
- Validate syntax
- Restart/reload services
- Use version control

---

3. NIS (NETWORK INFORMATION SERVICE)

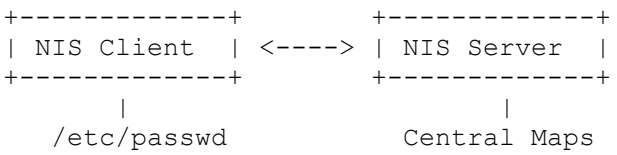
3.1 What is NIS?

NIS is a **centralized authentication and directory service** that allows **sharing user and group information across multiple Linux systems**.

3.2 Why NIS is Used

- Centralized user management
- Single user database
- Reduced administrative overhead
- Consistent UID/GID across systems

3.3 NIS Architecture



3.4 NIS Components

Component	Description
ypserv	NIS server daemon
ypbind	NIS client daemon
yppasswdd	Password updates
NIS maps	User/group databases

3.5 NIS Domain Name

- Logical grouping
- Not DNS domain

```
domainname example-nis
```

3.6 NIS Server Configuration

Install NIS Packages

```
yum install ypserv ypbind
```

Set NIS Domain

```
echo "example-nis" > /etc/sysconfig/network
domainname example-nis
```

Configure NIS Makefile

```
/var/yp/Makefile
```

Controls which maps are built.

Start NIS Services

```
systemctl start ypserv
systemctl enable ypserv
```

Build NIS Maps

```
/usr/lib64/yp/ypinit -m
```

3.7 NIS Client Configuration

Install Client Packages

```
yum install ypbind
```

Configure yp.conf

```
/etc/yp.conf
domain example-nis server nis-server
```

Start Client Service

```
systemctl start ypbind
systemctl enable ypbind
```

Update NSS Configuration

```
/etc/nsswitch.conf
passwd: files nis
group:  files nis
shadow: files nis
```

3.8 Verifying NIS Configuration

```
ypwhich
```

```
ypcat passwd
getent passwd
```

3.9 Security Considerations of NIS

⚠ NIS is NOT encrypted

Risks:

- Password exposure
- Network sniffing
- No strong authentication

Mitigation:

- Use secure networks
- Firewall NIS ports
- Prefer LDAP/SSSD for production

3.10 NIS vs LDAP

Feature	NIS	LDAP
Encryption	No	Yes
Scalability	Limited	High
Security	Weak	Strong
Modern Use	Legacy	Recommended

---

4. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Service Management

- Start/stop SSH
- Enable HTTP service
- Analyze logs using journalctl

Lab 2 – Configuration Files

- Edit `/etc/hosts`
- Configure `/etc/fstab`
- Change hostname safely

Lab 3 – NIS Setup

- Configure NIS server
- Add users centrally
- Verify login from client

---

5. SECURITY & ADMINISTRATION BEST PRACTICES

- Disable unnecessary services
- Restrict root access
- Monitor service logs
- Secure `/etc` permissions
- Avoid NIS in hostile networks

---

6. EXAM & INTERVIEW POINTS

- systemd PID = 1
- `systemctl` controls services
- `/etc` contains system configs
- NIS centralizes user accounts
- `nsswitch.conf` controls auth lookup
- NIS is legacy, LDAP is preferred

---

7. MINI CASE STUDY

Scenario:
Organization needs centralized user management for 20 Linux servers.

Solution:

- Deploy NIS server
- Configure NIS clients
- Centralize `/etc/passwd`
- Control services via systemd
- Monitor logs centrally

SESSION 11 – PATCH MANAGEMENT, SYSTEM MANAGEMENT & X CONFIGURATION SERVER

---

1. PATCH MANAGEMENT IN LINUX

1.1 Definition

Patch management is the **process of identifying, acquiring, testing, deploying, and verifying software updates (patches)** to fix bugs, close security vulnerabilities, and improve system stability.

1.2 Why Patch Management Is Critical

- Fixes security vulnerabilities (CVEs)
- Improves system stability
- Ensures compliance (ISO, PCI-DSS)
- Prevents exploitation
- Maintains vendor support eligibility

1.3 Types of Patches

Patch Type	Description
Security	Fixes vulnerabilities
Bug Fix	Resolves functional issues
Enhancement	Improves features
Kernel	Updates Linux kernel
Firmware	Hardware-level updates

1.4 Patch Management Lifecycle

```
Patch Identification
        ↓
Risk Assessment
        ↓
Testing (Staging)
        ↓
Deployment
        ↓
Verification
        ↓
Rollback (if needed)
```

1.5 Linux Patch Sources

- Distribution repositories
- Vendor security advisories
- CVE databases
- Kernel.org

1.6 Patch Management Tools

YUM / DNF (RHEL, CentOS, Rocky)

```
yum check-update
dnf update
dnf update --security
```

Explanation:

- `check-update` → Lists available patches
- `update` → Applies all updates
- `--security` → Applies only security patches

APT (Debian / Ubuntu)

```
apt update
apt upgrade
apt full-upgrade
```

Kernel Patch Handling

```
uname -r
dnf update kernel
reboot
```

⚠ Kernel patches require reboot

1.7 Automated Patch Management

dnf-automatic

```
dnf install dnf-automatic
systemctl enable dnf-automatic.timer
```

Cron-Based Updates

```
crontab -e
```

1.8 Patch Testing Strategy

- Clone production environment
- Apply patches in staging
- Validate services
- Schedule maintenance window

1.9 Rollback & Recovery

- Snapshot (VM/LVM)
- Package downgrade

```
dnf downgrade package
```

1.10 Security Considerations

- Patch regularly

- Use trusted repositories only
- Test before production rollout
- Monitor CVEs

---

2. SYSTEM MANAGEMENT

2.1 Definition

System management involves **monitoring, maintaining, optimizing, and controlling Linux systems** to ensure high availability, performance, and reliability.

2.2 Core System Management Areas

Area	Description
Process Management	CPU & process control
Memory Management	RAM & swap usage
Disk Management	Storage utilization
Network Management	Connectivity & throughput
User Management	Access control
Service Management	Daemon lifecycle

2.3 System Information Commands

```
uname -a
hostnamectl
uptime
lsb_release -a
```

2.4 Process Management

Viewing Processes

```
ps aux
top
htop
```

Controlling Processes

```
kill PID
kill -9 PID
```

2.5 Memory Management

```
free -h
vmstat
swapon -s
```

2.6 Disk Management Monitoring

```
df -h
du -sh /var
lsblk
```

2.7 Log Management

```
journalctl
journalctl -xe
journalctl -u sshd
```

2.8 Scheduled Tasks (Cron)

```
crontab -e
```

Example:

```
0 2 * * * dnf update -y
```

2.9 Backup & Recovery

- rsync
- tar
- snapshots
- centralized backup servers

2.10 System Performance Tuning

- Tune kernel parameters (`sysctl`)
- Disable unnecessary services
- Monitor CPU, RAM, I/O

---

3. X CONFIGURATION SERVER (X WINDOW SYSTEM)

3.1 What Is X Configuration Server?

The **X Window System (X11)** is a **network-transparent graphical display system** that provides the foundation for graphical user interfaces (GUI) in Linux.

3.2 Components of X Window System

```
+------------------------+
| GUI Applications       |
+------------------------+
| Window Manager         |
+------------------------+
| X Server (Xorg)        |
+------------------------+
| Display / Keyboard / Mouse |
+------------------------+
```

3.3 X Server (Xorg)

- Controls display hardware
- Handles keyboard & mouse input
- Renders graphics
- Communicates using X protocol

3.4 X Configuration Files

Primary Configuration File

```
/etc/X11/xorg.conf
```

⚠ Often auto-generated or absent in modern systems

Directory-Based Configuration

```
/etc/X11/xorg.conf.d/
```

3.5 xorg.conf Structure

```
Section "Device"
    Identifier "GPU0"
    Driver "nvidia"
EndSection

Section "Monitor"
    Identifier "Monitor0"
EndSection

Section "Screen"
    Identifier "Screen0"
    Device "GPU0"
    Monitor "Monitor0"
EndSection
```

3.6 Display Managers

Display Manager	Description
GDM	GNOME
LightDM	Lightweight
SDDM	KDE

3.7 Managing Graphical Targets

```
systemctl get-default
systemctl set-default graphical.target
systemctl set-default multi-user.target
```

3.8 Starting & Stopping X Server

```
startx
systemctl start gdm
systemctl stop gdm
```

3.9 Remote X Access

X Forwarding via SSH

```
ssh -X user@server
```

3.10 X Server Security Considerations

- Disable unused X services
- Restrict X forwarding
- Avoid running GUI as root
- Secure display manager

---

4. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Patch Management

- Check updates
- Apply security patches
- Verify kernel version

Lab 2 – System Management

- Monitor processes
- Analyze memory usage
- Schedule cron job

Lab 3 – X Configuration

- Switch between CLI & GUI
- Modify display resolution
- Test X forwarding

---

5. BEST PRACTICES

- Regular patch cycles
- Centralized monitoring
- Minimal GUI on servers
- Document changes
- Use automation tools

---

6. EXAM & INTERVIEW POINTS

- Patch management closes CVEs
- Kernel updates require reboot
- systemd manages system targets
- X Server handles display
- `graphical.target` enables GUI
- `startx` launches X session

---

7. MINI CASE STUDY

Scenario:
Enterprise Linux servers require secure operation and limited GUI use.

Solution:

- Automated patching
- CLI-based system management
- Disable GUI on servers
- Enable X only on workstations
- Centralized monitoring

SESSION 12 – DNS CONCEPTS & DNS CONFIGURATION IN LINUX

---

1. DNS CONCEPTS

1.1 What is DNS? (Definition)

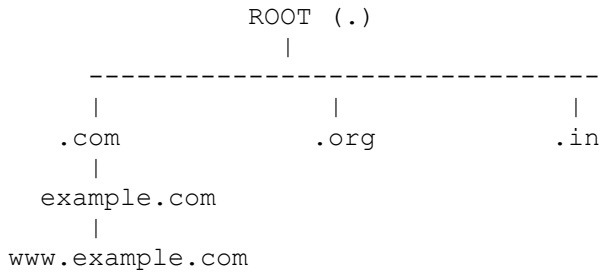
DNS (Domain Name System) is a **hierarchical, distributed naming system** that translates **human-readable domain names** (e.g., `www.example.com`) into **machine-readable IP addresses** (IPv4 or IPv6).

➡ DNS is often called the **"phonebook of the Internet."**

1.2 Why DNS is Required

- Humans remember names, not IPs
- Enables service discovery (web, mail, FTP)
- Supports load balancing & redundancy
- Decouples services from IP changes
- Critical for Internet and enterprise networks

1.3 DNS Architecture (Hierarchical Model)



1.4 DNS Resolution Workflow

```
Client
  ↓
Local Resolver
  ↓
Root Server
  ↓
TLD Server (.com)
  ↓
```

```
Authoritative Name Server
  ↓
IP Address Returned
```

1.5 DNS Components

Component	Description
DNS Client	Requests name resolution
Resolver	Queries DNS hierarchy
Authoritative Server	Stores zone data
Root Servers	Top-level DNS
TLD Servers	Manage domain extensions

1.6 Types of DNS Servers

Server Type	Function
Recursive Resolver	Resolves queries fully
Authoritative	Answers for its zone
Primary (Master)	Read-write zone
Secondary (Slave)	Read-only replica
Caching Server	Stores previous results

1.7 DNS Record Types

Record	Purpose
A	IPv4 address
AAAA	IPv6 address
NS	Name server
MX	Mail exchange
CNAME	Alias
PTR	Reverse lookup
SOA	Zone authority
TXT	Text info (SPF, DKIM)

Example DNS Records

```
www     IN  A      192.168.1.10
```

```
mail    IN  MX 10  mail.example.com.
```

1.8 Forward vs Reverse DNS

Type	Purpose
Forward DNS	Name → IP
Reverse DNS	IP → Name

1.9 DNS Ports & Protocols

- **UDP 53** → Standard queries
- **TCP 53** → Zone transfers & large responses

---

2. DNS CONFIGURATION IN LINUX

2.1 DNS Software in Linux

The most commonly used DNS server software is:

BIND (Berkeley Internet Name Domain)

- De facto standard
- Highly configurable
- Supports DNSSEC
- Enterprise ready

2.2 BIND Architecture

```
+-----------------------+
| DNS Clients           |
+-----------------------+
| named (BIND Daemon)   |
+-----------------------+
| Zone Files            |
+-----------------------+
| DNS Database          |
+-----------------------+
```

2.3 Installing BIND

RHEL / CentOS / Rocky

```
yum install bind bind-utils
```

Debian / Ubuntu

```
apt install bind9 dnsutils
```

2.4 BIND Configuration Files

File	Purpose
/etc/named.conf	Main config
/etc/named.rfc1912.zones	Default zones
/var/named/	Zone files
/etc/resolv.conf	DNS client config

2.5 Main Configuration File – named.conf

```
vi /etc/named.conf
```

Key Sections

```
options {
    directory "/var/named";
    listen-on port 53 { any; };
    allow-query { any; };
};

zone "." IN {
    type hint;
    file "named.ca";
};
```

2.6 Forward Zone Configuration

```
zone "example.com" IN {
    type master;
    file "example.com.zone";
```

```
};
```

2.7 Forward Zone File (example.com.zone)

```
$TTL 86400
@   IN  SOA ns1.example.com. admin.example.com. (
        2026010101 ; Serial
        3600       ; Refresh
        1800       ; Retry
        604800     ; Expire
        86400 )    ; Minimum

@       IN  NS      ns1.example.com.
ns1     IN  A       192.168.1.10
www     IN  A       192.168.1.20
mail    IN  A       192.168.1.30
@       IN  MX 10   mail.example.com.
```

2.8 Reverse Zone Configuration

```
zone "1.168.192.in-addr.arpa" IN {
    type master;
    file "192.168.1.rev";
};
```

2.9 Reverse Zone File (192.168.1.rev)

```
$TTL 86400
@   IN  SOA ns1.example.com. admin.example.com. (
        2026010101
        3600
        1800
        604800
        86400 )

@       IN  NS      ns1.example.com.
10      IN  PTR     ns1.example.com.
20      IN  PTR     www.example.com.
30      IN  PTR     mail.example.com.
```

2.10 Validating DNS Configuration

```
named-checkconf
named-checkzone example.com example.com.zone
```

2.11 Starting DNS Service

```
systemctl start named
systemctl enable named
```

2.12 Client DNS Configuration

```
/etc/resolv.conf
nameserver 192.168.1.10
search example.com
```

2.13 Testing DNS Resolution

```
nslookup www.example.com
dig example.com
host www.example.com
```

---

3. DNS SECURITY CONSIDERATIONS

3.1 Common DNS Threats

- DNS spoofing
- Cache poisoning
- Zone transfer abuse
- DDoS amplification
- Unauthorized updates

3.2 DNS Hardening Best Practices

Restrict Zone Transfers

```
allow-transfer { 192.168.1.11; };
```

Disable Recursion for Public Servers

```
recursion no;
```

Enable DNSSEC

- Digitally signs DNS records
- Prevents spoofing

Firewall Rules

```
firewall-cmd --add-service=dns --permanent
firewall-cmd --reload
```

3.3 Logging & Monitoring

```
journalctl -u named
```

---

4. LAB PERSPECTIVE (MANDATORY)

Lab 1 – DNS Server Setup

- Install BIND
- Configure forward zone
- Test name resolution

Lab 2 – Reverse DNS

- Configure reverse zone
- Test PTR lookups

Lab 3 – DNS Security

- Restrict zone transfers
- Disable recursion
- Test access control

---

5. EXAM & INTERVIEW POINTS

- DNS is hierarchical & distributed
- Port 53 (UDP/TCP)
- SOA defines zone authority
- Serial number change required after updates
- Forward = Name → IP
- Reverse = IP → Name
- BIND is default Linux DNS server

---

6. MINI CASE STUDY

Scenario:
Company needs internal DNS for 200 hosts.

Solution:

- Deploy BIND server
- Create forward & reverse zones
- Secure zone transfers
- Configure clients via DHCP
- Monitor DNS logs

SESSION 13 – LINUX SERVICES, NFS SERVER & FTP SERVER

---

1. INTRODUCTION TO LINUX SERVICES

1.1 What Are Linux Services? (Definition)

Linux services (also called **daemons**) are **background processes** that provide **continuous system or network functionality** without direct user interaction.

Examples:

- File sharing
- Remote access

- Web hosting
- Authentication
- Printing

1.2 Characteristics of Linux Services

- Run in background
- Start at boot or on demand
- Managed by `systemd`
- Controlled using `systemctl`
- Configured using text files (mostly in `/etc`)

1.3 Common Linux Network Services

Service	Purpose
SSH	Secure remote login
NFS	Network file sharing (Linux/Unix)
FTP	File transfer
HTTP/HTTPS	Web services
DNS	Name resolution
SMTP	Mail transfer

1.4 Service Management Recap (systemd)

Service Control

```
systemctl start service
systemctl stop service
systemctl restart service
systemctl status service
```

Enable at Boot

```
systemctl enable service
systemctl disable service
```

1.5 Service Ports & Firewalls

- Services listen on **well-known ports**
- Firewalls must explicitly allow them

Example:

```
firewall-cmd --add-service=nfs --permanent
firewall-cmd --reload
```

1.6 Security Principles for Services

- Enable only required services
- Restrict access by IP/user
- Use encryption where possible
- Monitor logs
- Apply patches regularly

---

2. NFS SERVER (NETWORK FILE SYSTEM)

2.1 What Is NFS?

NFS (Network File System) allows **remote directories to be mounted and accessed as if they are local directories** on Linux/Unix systems.

2.2 Why NFS Is Used

- Centralized file storage
- Transparent access to shared data
- Diskless clients
- Common in Linux servers & clusters

2.3 NFS Architecture

```
+-------------------+       +-------------------+
```

```
|   NFS Client      | <--> |   NFS Server      |
| mount /data       |      | exports /data     |
+-------------------+      +-------------------+
```

2.4 NFS Components

Component	Description
nfsd	NFS daemon
rpcbind	Port mapping
exportfs	Manages exports
/etc/exports	Share configuration

2.5 NFS Versions

Version	Features
NFSv3	Stateless
NFSv4	Stateful, better security
NFSv4.1	Performance & locking

2.6 Installing NFS Server

```
yum install nfs-utils
```

2.7 Starting NFS Services

```
systemctl start nfs-server
systemctl enable nfs-server
systemctl start rpcbind
```

2.8 NFS Configuration File – /etc/exports

Syntax

```
directory client(options)
```

Example

```
/data 192.168.1.0/24(rw,sync,no_root_squash)
```

2.9 Export Options (Very Important)

Option	Meaning
ro	Read-only
rw	Read-write
sync	Write changes immediately
async	Faster but unsafe
no_root_squash	Root has full access
root_squash	Root mapped to nobody

2.10 Apply Export Configuration

```
exportfs -rav
```

2.11 Firewall Configuration for NFS

```
firewall-cmd --add-service=nfs --permanent
firewall-cmd --add-service=rpc-bind --permanent
firewall-cmd --add-service=mountd --permanent
firewall-cmd --reload
```

2.12 NFS Client Configuration

Install Client

```
yum install nfs-utils
```

Mount NFS Share

```
mount server:/data /mnt
```

Persistent Mount (/etc/fstab)

```
server:/data  /mnt  nfs  defaults  0 0
```

2.13 NFS Security Considerations

- Use NFSv4
- Restrict IP ranges
- Avoid `no_root_squash`
- Use firewalls
- Combine with Kerberos for strong auth

---

3. FTP SERVER (FILE TRANSFER PROTOCOL)

3.1 What Is FTP?

FTP is a **client-server protocol** used to **transfer files between systems** over a network.

⚠ FTP transmits data **in plain text** (not secure).

3.2 FTP Architecture

```
FTP Client
   ↓
Control Connection (Port 21)
   ↓
Data Connection (Port 20 / Passive Ports)
   ↓
FTP Server
```

3.3 FTP Modes

Mode	Description
Active	Server initiates data connection
Passive	Client initiates data connection

3.4 FTP Ports

- Control: TCP 21
- Data: TCP 20 (Active)

- Passive: Dynamic ports

3.5 Popular FTP Server in Linux

vsftpd (Very Secure FTP Daemon)

3.6 Installing FTP Server

```
yum install vsftpd
```

3.7 FTP Configuration File

```
/etc/vsftpd/vsftpd.conf
```

3.8 Important vsftpd Configuration Options

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
chroot_local_user=YES
listen=YES
```

Option Explanation

- `anonymous_enable` → Disable anonymous access
- `local_enable` → Allow system users
- `write_enable` → Enable uploads
- `chroot_local_user` → Jail users in home directory
- `listen` → Standalone mode

3.9 Start & Enable FTP Service

```
systemctl start vsftpd
systemctl enable vsftpd
```

3.10 Firewall Configuration for FTP

```
firewall-cmd --add-service=ftp --permanent
firewall-cmd --reload
```

3.11 FTP User Access

- FTP uses **local Linux users**
- Home directory is FTP root
- Permissions controlled via filesystem ACLs

3.12 Testing FTP

```
ftp server_ip
```

Commands inside FTP:

```
ls
get file
put file
bye
```

3.13 FTP Security Issues

- Plaintext passwords
- Packet sniffing
- Firewall complexity

3.14 Secure Alternatives to FTP

Protocol	Security
FTPS	SSL/TLS encrypted
SFTP	SSH-based (recommended)

---

4. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Linux Services

- Check service status
- Enable/disable services
- Identify listening ports

Lab 2 – NFS

- Configure NFS server
- Export directory
- Mount from client
- Test permissions

Lab 3 – FTP

- Install vsftpd
- Configure secure access
- Upload/download files
- Analyze logs

---

5. SECURITY CONSIDERATIONS (EXAM & REAL-WORLD)

- Prefer **NFSv4** over older versions
- Never expose FTP publicly
- Use **SFTP** whenever possible
- Restrict service ports
- Monitor logs regularly

---

6. EXAM & INTERVIEW POINTS

- NFS provides transparent file access
- `/etc/exports` defines NFS shares
- `exportfs` manages exports
- FTP uses port 21
- vsftpd is most secure FTP daemon

- FTP is insecure by default

---

7. MINI CASE STUDY

Scenario:
Linux development team needs shared storage and file uploads.

Solution:

- Use **NFS** for internal shared codebase
- Use **SFTP** instead of FTP
- Restrict access using firewall & permissions
- Monitor usage and logs

SESSION 14 – SAMBA, DHCP & DNS SERVICES IN LINUX

---

1. SAMBA SERVER

1.1 Introduction to Samba

Definition

Samba is an **open-source implementation of the SMB/CIFS protocol** that allows **Linux systems to share files and printers with Windows systems**.

➡ Samba enables **cross-platform file and print sharing** between:

- Linux ↔ Windows
- Linux ↔ Linux

1.2 Why Samba Is Needed

- Windows–Linux interoperability

- Centralized file sharing
- Authentication integration (AD)
- Printer sharing
- Legacy Windows application support

1.3 Samba Architecture

```
+--------------------+        +--------------------+
| Windows Client     | <----> | Samba Server       |
| (SMB/CIFS)         |        | (smbd, nmbd)       |
+--------------------+        +--------------------+
```

1.4 Samba Components

Component	Purpose
smbd	File & printer sharing
nmbd	NetBIOS name resolution
winbindd	Windows user mapping
smb.conf	Main configuration file

1.5 Samba Ports

Port	Purpose
TCP 445	SMB over TCP
TCP 139	NetBIOS session
UDP 137	NetBIOS name
UDP 138	NetBIOS datagram

1.6 Installing Samba

```
yum install samba samba-client samba-common
```

1.7 Samba Configuration File

```
/etc/samba/smb.conf
```

1.8 Basic Samba Configuration

```
[global]
   workgroup = WORKGROUP
   security = user
   map to guest = bad user

[shared]
   path = /samba/share
   writable = yes
   browseable = yes
   guest ok = no
```

1.9 Samba User Management

Create Linux User

```
useradd sambauser
passwd sambauser
```

Add Samba User

```
smbpasswd -a sambauser
```

1.10 Starting Samba Services

```
systemctl start smb
systemctl enable smb
systemctl start nmb
```

1.11 Firewall Configuration

```
firewall-cmd --add-service=samba --permanent
firewall-cmd --reload
```

1.12 Accessing Samba Share

From Linux

```
smbclient //server/shared -U sambauser
```

From Windows

```
\\server\shared
```

1.13 Samba Security Considerations

- Disable guest access
- Use strong passwords
- Restrict IP access
- Integrate with AD where possible
- Audit access logs

---

2. DHCP SERVER

2.1 Introduction to DHCP

Definition

DHCP (Dynamic Host Configuration Protocol) automatically assigns **IP addresses and network configuration** to clients.

2.2 Why DHCP Is Required

- Eliminates manual IP configuration
- Prevents IP conflicts
- Centralized network management
- Scalable for large networks

2.3 DHCP Architecture

```
DHCP Client
   ↓ DISCOVER
DHCP Server
   ↓ OFFER
Client
   ↓ REQUEST
Server
   ↓ ACK
```

2.4 DHCP Message Flow (DORA)

Step	Description
Discover	Client broadcasts
Offer	Server offers IP
Request	Client accepts
Acknowledge	Server confirms

2.5 DHCP Ports

- UDP 67 – Server
- UDP 68 – Client

2.6 Installing DHCP Server

```
yum install dhcp-server
```

2.7 DHCP Configuration File

```
/etc/dhcp/dhcpd.conf
```

2.8 Basic DHCP Configuration

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.100 192.168.1.200;
  option routers 192.168.1.1;
  option domain-name-servers 192.168.1.10;
  default-lease-time 600;
  max-lease-time 7200;
}
```

2.9 Fixed IP Assignment

```
host printer {
  hardware ethernet 00:11:22:33:44:55;
  fixed-address 192.168.1.50;
```

```
}
```

2.10 Start DHCP Service

```
systemctl start dhcpd
systemctl enable dhcpd
```

2.11 Firewall Configuration

```
firewall-cmd --add-service=dhcp --permanent
firewall-cmd --reload
```

2.12 DHCP Logs

```
journalctl -u dhcpd
```

2.13 DHCP Security Considerations

- DHCP snooping
- Restrict to trusted interfaces
- Prevent rogue DHCP servers
- Monitor lease files

---

3. DNS SERVER (RECAP & INTEGRATION)

3.1 DNS Role in Network Integration

DNS provides:

- Name resolution
- Service discovery
- Integration with DHCP & Samba
- Active Directory support

3.2 DNS + DHCP Integration

- DHCP assigns IP
- DNS updates hostname records dynamically

```
ddns-update-style interim;
```

3.3 DNS + Samba Integration

- Required for Active Directory
- Kerberos authentication
- SRV records

3.4 BIND DNS Overview

- Authoritative server
- Forward & reverse zones
- Secure name resolution

3.5 Important DNS Files

File	Purpose
/etc/named.conf	Main config
/var/named/*.zone	Zone files
/etc/resolv.conf	Client DNS

3.6 DNS Security Considerations

- Restrict zone transfers
- Disable open recursion
- Enable DNSSEC
- Firewall port 53

---

4. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Samba

- Create shared directory
- Configure smb.conf
- Access share from Windows/Linux

Lab 2 – DHCP

- Configure IP pool
- Assign fixed IP
- Test client lease

Lab 3 – DNS Integration

- Configure DNS server
- Integrate with DHCP
- Verify name resolution

---

5. PERFORMANCE & BEST PRACTICES

- Use Samba with proper permissions
- Keep DHCP scopes clean
- Centralize DNS
- Monitor logs regularly
- Backup configuration files

---

6. EXAM & INTERVIEW POINTS

- Samba = Linux ↔ Windows sharing
- DHCP uses DORA process
- DHCP ports: 67/68
- DNS port: 53
- Samba uses SMB/CIFS
- DHCP reduces admin overhead

---

7. MINI CASE STUDY

Scenario:
Enterprise with Windows & Linux systems needs centralized services.

Solution:

- Samba for file sharing
- DHCP for IP assignment
- DNS for name resolution
- Secure integration & monitoring

SESSION 15 – APACHE WEB SERVER, APACHE SECURITY, VIRTUAL HOSTING & SQUID PROXY

---

1. APACHE WEB SERVER

1.1 Introduction to Apache

Definition

Apache HTTP Server (httpd) is an **open-source, modular, cross-platform web server** used to deliver **web content (HTML, PHP, APIs)** over HTTP/HTTPS.

1.2 Why Apache Is Widely Used

- Open source & stable
- Highly configurable
- Modular architecture
- Large community support
- Supports multiple languages & platforms

1.3 Apache Architecture

```
Client (Browser)
     ↓ HTTP/HTTPS
Apache Web Server
     ↓
MPM (Process/Thread Manager)
     ↓
Modules (mod_ssl, mod_rewrite, etc.)
     ↓
Filesystem / Backend Apps
```

1.4 Apache Processing Models (MPM)

MPM	Description
Prefork	Process-based
Worker	Thread-based
Event	Async, scalable

1.5 Installing Apache

RHEL / CentOS / Rocky

```
yum install httpd
```

Debian / Ubuntu

```
apt install apache2
```

1.6 Apache Service Management

```
systemctl start httpd
systemctl enable httpd
systemctl status httpd
```

1.7 Important Apache Directories & Files

Path	Purpose
/etc/httpd/conf/httpd.conf	Main config
/etc/httpd/conf.d/	Additional configs
/var/www/html/	Default web root
/var/log/httpd/	Logs

1.8 Apache Basic Configuration (httpd.conf)

```
ServerName www.example.com:80
Listen 80
DocumentRoot "/var/www/html"
```

1.9 Testing Apache

```
http://server-ip
```

---

2. APACHE SECURITY

2.1 Importance of Web Server Security

Web servers are exposed to:

- Unauthorized access
- Code injection
- DDoS attacks
- Information leakage

2.2 Apache Security Best Practices

Hide Server Information

```
ServerTokens Prod
ServerSignature Off
```

Disable Directory Listing

```
<Directory "/var/www/html">
    Options -Indexes
</Directory>
```

Restrict Access by IP

```
<Directory "/var/www/html/secure">
    Require ip 192.168.1.0/24
</Directory>
```

Disable Unused Modules

```
httpd -M
```

2.3 SSL/TLS (HTTPS)

Enable mod_ssl

```
yum install mod_ssl
```

SSL Configuration

```
<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /etc/pki/tls/certs/server.crt
    SSLCertificateKeyFile /etc/pki/tls/private/server.key
</VirtualHost>
```

2.4 Authentication & Authorization

Basic Authentication

```
<Directory "/var/www/html/private">
    AuthType Basic
    AuthName "Restricted"
    AuthUserFile /etc/httpd/.htpasswd
    Require valid-user
</Directory>
```

Create Password File

```
htpasswd -c /etc/httpd/.htpasswd user1
```

2.5 Apache Logs & Monitoring

```
tail -f /var/log/httpd/access_log
tail -f /var/log/httpd/error_log
```

---

3. VIRTUAL HOSTING

3.1 What Is Virtual Hosting?

Virtual hosting allows **multiple websites to be hosted on a single Apache server**.

3.2 Types of Virtual Hosting

Type	Description
Name-based	Based on domain name
IP-based	Based on IP address
Port-based	Based on port

3.3 Name-Based Virtual Host Example

```
<VirtualHost *:80>
    ServerName site1.example.com
    DocumentRoot /var/www/site1
</VirtualHost>

<VirtualHost *:80>
    ServerName site2.example.com
    DocumentRoot /var/www/site2
</VirtualHost>
```

3.4 Virtual Host Directory Setup

```
mkdir /var/www/site1
mkdir /var/www/site2
```

3.5 DNS Requirement

Each virtual host must resolve via DNS.

3.6 Virtual Hosting Security

- Separate directories
- Correct permissions
- SELinux contexts
- Disable directory browsing

---

4. SQUID PROXY SERVER

4.1 Introduction to Squid

Definition

Squid is a **caching proxy server** that improves **web performance, security, and access control**.

4.2 Why Use Squid?

- Reduce bandwidth usage
- Improve browsing speed
- Enforce browsing policies
- Centralized Internet access control

4.3 Squid Architecture

```
Client Browser
     ↓
Squid Proxy
     ↓
Internet Web Server
```

4.4 Squid Ports

- Default: **TCP 3128**

4.5 Installing Squid

```
yum install squid
```

4.6 Squid Configuration File

```
/etc/squid/squid.conf
```

4.7 Basic Squid Configuration

```
http_port 3128
acl localnet src 192.168.1.0/24
http_access allow localnet
http_access deny all
```

4.8 Access Control Lists (ACLs)

Block Websites

```
acl blocked_sites dstdomain .facebook.com
http_access deny blocked_sites
```

Time-Based Access

```
acl work_hours time MTWHF 09:00-18:00
http_access allow work_hours
```

4.9 Start & Enable Squid

```
systemctl start squid
systemctl enable squid
```

4.10 Firewall Configuration

```
firewall-cmd --add-port=3128/tcp --permanent
```

```
firewall-cmd --reload
```

4.11 Squid Security Considerations

- Restrict access by IP
- Enable authentication
- Monitor logs
- Disable open proxy

---

5. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Apache

- Install Apache
- Serve static website
- Analyze logs

Lab 2 – Apache Security

- Enable HTTPS
- Restrict directories
- Hide server version

Lab 3 – Virtual Hosting

- Configure multiple sites
- Test name-based hosting

Lab 4 – Squid

- Configure proxy
- Block websites
- Test caching

---

6. PERFORMANCE & BEST PRACTICES

- Use Event MPM
- Enable caching
- Secure Apache
- Monitor logs
- Regular updates

---

7. EXAM & INTERVIEW POINTS

- Apache uses port 80/443
- Virtual hosting hosts multiple sites
- Squid caches web content
- Proxy port is 3128
- mod_ssl enables HTTPS
- ACLs control Squid access

---

8. MINI CASE STUDY

Scenario:
Organization needs secure web hosting and controlled Internet access.

Solution:

- Apache with HTTPS
- Virtual hosts for departments
- Squid proxy for filtering
- Firewall + logging

SESSION 16 – LINUX MAIL SERVICES: EMAIL FLOW, POSTFIX, DOVECOT & SQUIRRELMAIL

---

1. EMAIL DELIVERY PROCESS

1.1 What is an Email System? (Definition)

An **email system** is a collection of **protocols, servers, services, and clients** that work together to **send, receive, store, and retrieve electronic mail** over a network.

1.2 Components of an Email System

```
Sender (MUA)
   ↓ SMTP
Outgoing Mail Server (MTA)
   ↓ SMTP
Recipient Mail Server (MTA)
   ↓ LDA
Mailbox (Mail Store)
   ↓ IMAP/POP3
Receiver (MUA/Webmail)
```

Key Components Explained

Component	Role
MUA	Mail User Agent (Outlook, Thunderbird, Webmail)
MTA	Mail Transfer Agent (Postfix, Sendmail)
MDA/LDA	Mail Delivery Agent (Dovecot LDA)
Mailbox	Storage (Maildir, mbox)
DNS	MX record resolution

1.3 Email Protocols

Protocol	Purpose	Port
SMTP	Send mail	25 / 587
POP3	Download mail	110 / 995
IMAP	Synchronize mail	143 / 993

1.4 Email Sending Flow (Step-by-Step)

1. User composes email in MUA
2. MUA sends mail via **SMTP** to MTA
3. MTA queries **DNS MX records**

4. Mail transferred to recipient MTA
5. Mail delivered to mailbox
6. User retrieves mail via **IMAP/POP3**

1.5 DNS Role in Email

MX Record Example

```
example.com.    IN  MX 10 mail.example.com.
```

➡ MX records define **which server receives mail** for a domain.

1.6 Common Email Threats

- Spam
- Phishing
- Spoofing
- Open relay abuse
- Malware attachments

---

2. POSTFIX (MAIL TRANSFER AGENT)

2.1 What is Postfix?

Postfix is a **high-performance, secure, and modular Mail Transfer Agent (MTA)** used to **send and receive email**.

➡ Designed as a **secure alternative to Sendmail**

2.2 Why Postfix is Popular

- Secure by design
- High performance

- Easy configuration
- Scalable
- Queue-based architecture

2.3 Postfix Architecture

```
SMTP Client
   ↓
smtpd (Postfix)
   ↓
Queue Manager
   ↓
Local Delivery / Remote SMTP
```

2.4 Postfix Components

Component	Purpose
smtpd	Receives mail
smtp	Sends mail
pickup	Picks mail from maildrop
cleanup	Cleans headers
qmgr	Manages queues

2.5 Installing Postfix

```
yum install postfix
```

2.6 Postfix Configuration Files

File	Purpose
/etc/postfix/main.cf	Main configuration
/etc/postfix/master.cf	Service control
/var/spool/postfix/	Mail queues

2.7 Main Configuration – main.cf

```
myhostname = mail.example.com
```

```
mydomain = example.com
myorigin = $mydomain
mydestination = $myhostname, localhost.$mydomain, $mydomain
inet_interfaces = all
```

2.8 Starting Postfix

```
systemctl start postfix
systemctl enable postfix
```

2.9 Mail Queue Management

```
mailq
postqueue -p
postqueue -f
```

2.10 Testing Postfix

```
mail user@example.com
```

2.11 Postfix Security Best Practices

- Disable open relay
- Enable SMTP authentication
- Use TLS encryption
- Restrict networks (`mynetworks`)
- Enable spam filtering

---

3. DOVECOT (IMAP/POP3 SERVER)

3.1 What is Dovecot?

Dovecot is a **secure, high-performance IMAP and POP3 server** used for **mailbox access and authentication**.

3.2 Why Dovecot is Used

- Fast mailbox access
- Secure authentication
- Supports Maildir & mbox
- Integrates with Postfix

3.3 Dovecot Architecture

```
Mail Client
   ↓ IMAP/POP3
Dovecot
   ↓
Mail Storage (Maildir)
```

3.4 Installing Dovecot

```
yum install dovecot
```

3.5 Dovecot Configuration Files

```
/etc/dovecot/dovecot.conf
/etc/dovecot/conf.d/
```

3.6 Important Dovecot Settings

```
protocols = imap pop3
mail_location = maildir:~/Maildir
```

3.7 Authentication Configuration

```
disable_plaintext_auth = yes
auth_mechanisms = plain login
```

3.8 Start Dovecot

```
systemctl start dovecot
systemctl enable dovecot
```

3.9 Dovecot & Postfix Integration

```
mailbox_command = /usr/libexec/dovecot/deliver
```

➡ Dovecot acts as **LDA (Local Delivery Agent)**

3.10 Dovecot Security

- Enable SSL/TLS
- Disable plaintext logins
- Restrict access by firewall
- Secure authentication backends

---

4. SQUIRRELMAIL (WEBMAIL CLIENT)

4.1 What is SquirrelMail?

SquirrelMail is a **PHP-based webmail application** that allows users to **access email via a web browser**.

4.2 Why Webmail is Used

- No client installation
- Accessible from anywhere
- Works with IMAP servers
- Lightweight & simple

4.3 SquirrelMail Architecture

```
Web Browser
   ↓ HTTP/HTTPS
Web Server (Apache)
```

```
   ↓
SquirrelMail (PHP)
   ↓ IMAP/SMTP
Dovecot / Postfix
```

4.4 Installing SquirrelMail

```
yum install squirrelmail
```

4.5 SquirrelMail Configuration

```
/usr/share/squirrelmail/config/conf.pl
```

4.6 Important Settings

- IMAP Server: localhost
- SMTP Server: localhost
- Domain name
- Data directory permissions

4.7 Accessing Webmail

```
http://server-ip/webmail
```

4.8 SquirrelMail Security Considerations

- Use HTTPS only
- Keep PHP updated
- Restrict admin access
- Strong user passwords

---

5. COMPLETE MAIL SERVER WORKFLOW

```
User → SquirrelMail (Web)
      ↓ SMTP
   Postfix (MTA)
      ↓ LDA
```

```
Dovecot (Mail Store)
   ↓ IMAP
User Reads Mail
```

---

6. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Postfix

- Configure hostname & domain
- Send test mail
- Monitor mail queue

Lab 2 – Dovecot

- Configure IMAP
- Create Maildir
- Login via mail client

Lab 3 – Webmail

- Install Apache + PHP
- Configure SquirrelMail
- Login & send mail

---

7. SECURITY CONSIDERATIONS (CRITICAL)

- Use TLS for SMTP/IMAP
- Disable open relay
- Implement spam filtering
- Monitor mail logs
- Harden DNS (SPF, DKIM, DMARC)

---

8. EXAM & INTERVIEW POINTS

- Postfix = MTA
- Dovecot = IMAP/POP3 + LDA
- SMTP sends mail
- IMAP synchronizes mail
- Webmail uses IMAP + SMTP
- MX records route email

---

9. MINI CASE STUDY

Scenario:
Organization needs secure internal email system.

Solution:

- Postfix for mail transfer
- Dovecot for mailbox access
- SquirrelMail for web access
- TLS encryption enabled
- Firewall-restricted access

SESSION 17 – PERFORMANCE TUNING, MAINTENANCE & TROUBLESHOOTING, THREAT MODEL & PROTECTION

---

1. PERFORMANCE TUNING IN LINUX

1.1 Definition

Performance tuning is the **systematic process of optimizing Linux system resources** (CPU, memory, disk, network, kernel) to achieve:

- Maximum throughput
- Minimum latency
- High availability
- Predictable performance

1.2 Why Performance Tuning Is Required

- Prevent bottlenecks
- Improve application response time
- Support scalability
- Reduce operational cost
- Ensure SLA compliance

1.3 Performance Tuning Layers

```
Application Layer
       ↓
Service / Daemon Layer
       ↓
OS / Kernel Layer
       ↓
Hardware Layer
```

1.4 CPU Performance Tuning

1.4.1 CPU Concepts

- Cores & threads
- Context switching
- Load average
- CPU scheduling

1.4.2 Monitoring CPU Usage

```
top
htop
mpstat
uptime
```

Load Average Interpretation

```
1.00 → One core fully used
```

1.4.3 CPU Tuning Techniques

- Reduce unnecessary services
- Optimize application threads
- Use task affinity

```
taskset -c 0,1 command
```

1.5 MEMORY PERFORMANCE TUNING

1.5.1 Memory Components

- RAM
- Buffer cache
- Page cache
- Swap

1.5.2 Memory Monitoring

```
free -h
vmstat
top
```

1.5.3 Swap Tuning

```
cat /proc/sys/vm/swappiness
```

Reduce swapping:

```
sysctl vm.swappiness=10
```

1.5.4 Memory Optimization Techniques

- Increase RAM
- Optimize cache usage
- Tune swappiness
- Use huge pages (DB servers)

1.6 DISK & I/O PERFORMANCE TUNING

1.6.1 Disk I/O Concepts

- IOPS
- Throughput
- Latency
- Block size

1.6.2 Disk Monitoring Tools

```
iostat
iotop
df -h
du -sh
```

1.6.3 Disk Optimization Techniques

- Use SSD/NVMe
- Enable read-ahead
- Separate OS & data disks
- Use LVM striping
- Avoid disk overcommitment

1.7 NETWORK PERFORMANCE TUNING

1.7.1 Network Metrics

- Bandwidth
- Latency
- Packet loss
- Errors

1.7.2 Network Monitoring

```
ip -s link
ss -s
netstat -i
```

1.7.3 Network Tuning

- Increase socket buffers
- Tune TCP parameters

```
sysctl net.core.rmem_max
sysctl net.ipv4.tcp_fin_timeout
```

1.8 KERNEL PERFORMANCE TUNING

1.8.1 Kernel Parameters

Managed using:

```
sysctl
```

Example:

```
sysctl -a
```

1.8.2 Permanent Kernel Tuning

```
/etc/sysctl.conf
```

1.9 PERFORMANCE TUNING BEST PRACTICES

- Measure before tuning
- Tune one component at a time
- Monitor continuously
- Document changes
- Rollback if needed

---

2. MAINTENANCE AND TROUBLESHOOTING

2.1 Definition

System maintenance and troubleshooting involve **keeping Linux systems stable, secure, and operational**, and **systematically diagnosing and resolving issues**.

2.2 Types of Maintenance

Type	Description
Preventive	Patch updates, cleanup
Corrective	Fixing issues
Predictive	Monitoring trends
Emergency	Incident response

2.3 Preventive Maintenance Tasks

- Patch management
- Log rotation
- Backup verification
- Disk cleanup
- Service health checks

2.4 Troubleshooting Methodology

```
Identify Problem
      ↓
Gather Information
      ↓
Analyze Root Cause
      ↓
Apply Fix
      ↓
Verify
      ↓
Document
```

2.5 Common Troubleshooting Areas

2.5.1 Boot Issues

- GRUB misconfiguration
- Missing initramfs
- Disk failure

Commands:

```
journalctl -xb
```

2.5.2 Service Failures

```
systemctl status service
journalctl -u service
```

2.5.3 Disk Full Issues

```
df -h
du -sh /*
```

2.5.4 Network Issues

```
ip addr
ping
traceroute
ss -tuln
```

2.5.5 Performance Degradation

- High CPU → runaway process
- High memory → memory leak
- High I/O → disk bottleneck

2.6 Log Files for Troubleshooting

File	Purpose
/var/log/messages	System events
/var/log/secure	Authentication

File	Purpose
/var/log/boot.log	Boot logs
journalctl	systemd logs

2.7 Maintenance Best Practices

- Use maintenance windows
- Automate repetitive tasks
- Keep documentation updated
- Maintain rollback plans

---

3. THREAT MODEL AND PROTECTION METHODS

3.1 What is a Threat Model?

A **threat model** identifies:

- Assets
- Threats
- Vulnerabilities
- Attack vectors
- Mitigation strategies

3.2 Linux Threat Model Components

```
Assets → Threats → Vulnerabilities → Attacks → Impact
```

3.3 Common Linux Threats

Threat	Description
Malware	Trojans, rootkits
Brute force	Password attacks
Privilege escalation	Kernel/user exploits
DoS/DDoS	Resource exhaustion

Threat	Description
Misconfiguration	Weak permissions

3.4 Attack Surfaces in Linux

- Open ports
- Weak services
- Insecure configurations
- Unpatched software
- User privileges

3.5 Protection Methods (Defense in Depth)

```
Physical Security
   ↓
Network Security
   ↓
System Security
   ↓
Application Security
   ↓
User Security
```

3.6 SYSTEM-LEVEL SECURITY CONTROLS

3.6.1 User & Access Control

- Least privilege
- sudo instead of root
- Strong passwords
- Account locking

3.6.2 Firewall

```
firewall-cmd --list-all
```

3.6.3 SELinux / AppArmor

- Mandatory access control
- Limits damage of compromised services

3.6.4 Patch Management

- Regular updates
- CVE tracking

3.7 NETWORK-LEVEL PROTECTION

- Firewalls
- IDS/IPS
- Secure protocols (SSH, HTTPS)
- Disable unused services

3.8 MONITORING & INCIDENT DETECTION

- Log monitoring
- File integrity monitoring
- Intrusion detection tools
- Alerting systems

3.9 INCIDENT RESPONSE PROCESS

```
Detection
   ↓
Containment
   ↓
Eradication
   ↓
Recovery
   ↓
Post-Incident Review
```

3.10 SECURITY BEST PRACTICES

- Harden systems

- Audit regularly
- Backup data
- Encrypt sensitive data
- Educate users

---

4. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Performance

- Identify CPU bottleneck
- Tune memory & swap
- Analyze disk I/O

Lab 2 – Troubleshooting

- Recover failed service
- Analyze logs
- Fix disk-full issue

Lab 3 – Security

- Harden SSH
- Configure firewall rules
- Detect suspicious activity

---

5. EXAM & INTERVIEW POINTS

- Performance tuning is data-driven
- CPU, memory, disk & network all matter
- Logs are primary troubleshooting tools
- Threat modeling = proactive security
- Defense-in-depth is key strategy
- Prevention is cheaper than recovery

---

6. MINI CASE STUDY

Scenario:
Production Linux server experiences slow response and suspicious login attempts.

Solution:

- Analyze CPU & memory usage
- Identify rogue process
- Patch system
- Harden SSH & firewall
- Monitor logs and alert admins

SESSION 18 – BASIC SERVICE SECURITY, LOGGING, NTP & BIND/DNS SECURITY

---

1. BASIC SERVICE SECURITY

1.1 Definition

Basic Service Security refers to the **hardening and protection of Linux system and network services** to minimize attack surfaces, prevent unauthorized access, and ensure secure operation.

1.2 Why Service Security Is Critical

- Services run continuously (high-value targets)
- Exposed to networks
- Often run with elevated privileges
- Misconfiguration leads to breaches
- Foundation of defense-in-depth strategy

1.3 Common Linux Services & Risks

Service	Risk
SSH	Brute force, key theft
FTP	Plaintext credentials
HTTP	Web exploits
DNS	Amplification, poisoning
Mail	Spam relay, spoofing
NFS	Unauthorized file access

1.4 Service Hardening Principles

```
Disable → Restrict → Harden → Monitor → Patch
```

1.5 Disable Unnecessary Services

List Active Services

```
systemctl list-units --type=service
```

Disable Service

```
systemctl stop telnet
systemctl disable telnet
```

1.6 Secure Service Configuration

Run Services with Least Privilege

- Avoid running as root
- Use dedicated service users

Restrict Network Access

```
firewall-cmd --add-service=ssh --permanent
firewall-cmd --reload
```

Bind Services to Required Interfaces

```
Listen 127.0.0.1
```

1.7 Service Isolation Mechanisms

systemd Security Options

```
PrivateTmp=true
NoNewPrivileges=true
ProtectSystem=strict
```

SELinux / AppArmor

- Mandatory Access Control
- Limits service damage even if compromised

1.8 Service Security Best Practices

- Minimal installation
- Strong authentication
- Encrypted protocols only
- Regular audits
- Continuous monitoring

---

2. LOGGING IN LINUX

2.1 What Is Logging?

Logging is the **systematic recording of system, service, security, and application events** to support:

- Monitoring
- Troubleshooting
- Auditing
- Incident response
- Compliance

2.2 Importance of Logging in Security

- Detect intrusions
- Track user activity
- Investigate incidents
- Meet compliance standards
- Provide forensic evidence

2.3 Linux Logging Architecture

```
Kernel / Applications
        ↓
systemd-journald
        ↓
rsyslog
        ↓
Log Files (/var/log)
```

2.4 systemd-journald

View Logs

```
journalctl
journalctl -xe
journalctl -u sshd
```

Persistent Logging

```
/etc/systemd/journald.conf
Storage=persistent
```

2.5 rsyslog

Role

- Traditional syslog daemon
- Log routing & remote logging

Configuration File

```
/etc/rsyslog.conf
/etc/rsyslog.d/*.conf
```

Log Routing Example

```
authpriv.*    /var/log/secure
```

2.6 Important Log Files

Log File	Purpose
/var/log/messages	System events
/var/log/secure	Auth attempts
/var/log/boot.log	Boot events
/var/log/cron	Scheduled tasks
/var/log/audit/audit.log	Audit events

2.7 Log Rotation

logrotate

```
/etc/logrotate.conf
/etc/logrotate.d/
```

Prevents disk exhaustion.

2.8 Centralized Logging

- rsyslog forwarding
- SIEM integration
- Improves visibility
- Prevents log tampering

2.9 Logging Security Best Practices

- Protect log permissions
- Enable remote logging

- Monitor logs in real-time
- Alert on suspicious events

---

3. NTP (NETWORK TIME PROTOCOL)

3.1 What Is NTP?

NTP synchronizes **system clocks across networked systems** with high accuracy.

3.2 Why Time Synchronization Matters

- Log correlation
- Authentication (Kerberos)
- Certificate validation
- Forensics accuracy
- Distributed systems consistency

3.3 NTP Architecture

```
Stratum 0 → Atomic Clock / GPS
Stratum 1 → Primary NTP Server
Stratum 2 → Secondary Server
Stratum 3 → Clients
```

3.4 NTP Software in Linux

- **chrony** (modern, preferred)
- ntpd (legacy)

3.5 Installing chrony

```
yum install chrony
```

3.6 chrony Configuration

```
/etc/chrony.conf
server time.google.com iburst
allow 192.168.1.0/24
```

3.7 Starting chrony

```
systemctl start chronyd
systemctl enable chronyd
```

3.8 Verify Time Sync

```
chronyc sources
timedatectl
```

3.9 NTP Security Risks

- Time spoofing
- Log manipulation
- Authentication bypass

3.10 NTP Security Best Practices

- Restrict clients
- Use authenticated NTP
- Firewall NTP port (UDP 123)
- Use trusted time sources

---

4. BIND AND DNS SECURITY

4.1 Why DNS Security Is Critical

DNS is:

- Core Internet service
- Target of amplification attacks
- Critical to authentication & routing
- Single point of failure

4.2 Common DNS Attacks

Attack	Description
Cache Poisoning	Fake DNS entries
Amplification	DDoS via DNS
Zone Transfer Abuse	Info leakage
Spoofing	Fake responses
Tunneling	Data exfiltration

4.3 BIND Security Architecture

```
DNS Client
   ↓
BIND (named)
   ↓
Zone Files
```

4.4 Secure BIND Configuration

Restrict Recursion

```
recursion no;
```

Limit Queries

```
allow-query { localnets; };
```

Restrict Zone Transfers

```
allow-transfer { 192.168.1.11; };
```

4.5 Run BIND as Non-Root

- Starts as root
- Drops privileges to `named`

4.6 Chroot Jail for BIND

- Limits file system exposure
- Reduces impact of compromise

4.7 DNSSEC (DNS Security Extensions)

Purpose

- Cryptographically signs DNS records
- Prevents spoofing & poisoning

DNSSEC Components

- Public/private keys
- Signed zones
- Trust chain

4.8 Firewall Protection

```
firewall-cmd --add-service=dns --permanent
firewall-cmd --reload
```

4.9 Logging DNS Events

```
journalctl -u named
```

4.10 DNS Security Best Practices

- Disable open resolvers
- Use DNSSEC
- Monitor query patterns
- Patch BIND regularly
- Separate authoritative & recursive servers

---

5. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Service Security

- Disable unused services
- Harden SSH
- Apply firewall rules

Lab 2 – Logging

- Enable persistent journaling
- Configure rsyslog
- Analyze authentication logs

Lab 3 – NTP

- Configure chrony server
- Sync clients
- Verify timestamps

Lab 4 – DNS Security

- Restrict recursion
- Configure zone transfer limits
- Enable DNS logging

---

6. EXAM & INTERVIEW POINTS

- Service security reduces attack surface
- Logs are forensic evidence
- Time sync is critical for security
- NTP uses UDP 123
- DNSSEC prevents spoofing
- Open DNS resolvers are dangerous

---

7. MINI CASE STUDY

Scenario:
Organization experienced suspicious login attempts and DNS anomalies.

Solution:

- Harden SSH & services
- Centralize logging
- Synchronize time via chrony
- Secure BIND with recursion & DNSSEC
- Monitor logs continuously

SESSION 19 – LDAP, NIS, APACHE CLUSTERING & LOAD BALANCING

---

1. LDAP (LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL)

1.1 Definition

LDAP is an **open, vendor-neutral, application-level protocol** used to **store, retrieve, and manage directory information** in a **centralized, hierarchical database**.

➡ LDAP is the **modern replacement for NIS** in enterprise environments.

1.2 Why LDAP Is Needed

- Centralized user authentication
- Single Sign-On (SSO)
- Consistent UID/GID across servers
- Scalability & security
- Integration with Linux, Windows (AD), applications

1.3 LDAP Use Cases

- Centralized Linux user management
- Email address book
- Application authentication
- Identity management
- Active Directory backend

1.4 LDAP Architecture

```
LDAP Client
   ↓
LDAP Server (Directory)
   ↓
Directory Information Tree (DIT)
```

1.5 LDAP Components

Component	Description
LDAP Client	Queries directory
LDAP Server	Stores directory
DIT	Hierarchical structure
Schema	Data structure definition
Entries	Objects (users, groups)

1.6 LDAP Directory Structure (DIT)

```
dc=example,dc=com
│
├── ou=People
│      ├── uid=john
│      └── uid=alice
│
```

```
└── ou=Groups
      └── cn=admins
```

1.7 LDAP Data Model

Entry

- Collection of attributes
- Identified by DN (Distinguished Name)

DN Example

```
uid=john,ou=People,dc=example,dc=com
```

1.8 LDAP Object Classes

- inetOrgPerson
- posixAccount
- posixGroup

1.9 Installing OpenLDAP

```
yum install openldap openldap-servers openldap-clients
```

1.10 LDAP Configuration Files

File	Purpose
/etc/openldap/slapd.d/	Main config
/etc/openldap/schema/	Schemas
/var/lib/ldap	Database

1.11 Starting LDAP Service

```
systemctl start slapd
systemctl enable slapd
```

1.12 LDAP Authentication with Linux (SSSD)

```
Application
   ↓
PAM
   ↓
SSSD
   ↓
LDAP
```

1.13 LDAP Security Considerations

- Use **LDAPS (SSL/TLS)**
- Restrict anonymous binds
- Access Control Lists (ACLs)
- Encrypt credentials
- Regular backups

1.14 LDAP vs NIS

Feature	LDAP	NIS
Security	Strong (TLS)	Weak
Scalability	High	Limited
Encryption	Yes	No
Modern Use	Recommended	Legacy

---

2. NIS (NETWORK INFORMATION SERVICE)

2.1 Definition

NIS is a **legacy centralized authentication service** used to **share user and group information** across Linux/UNIX systems.

2.2 Why NIS Was Used

- Central user management
- Simplified administration
- Consistent UID/GID

2.3 NIS Architecture

```
NIS Client
   ↓
NIS Server
   ↓
Maps (passwd, group, hosts)
```

2.4 NIS Components

Component	Purpose
ypserv	NIS server
ypbind	Client binding
yppasswdd	Password updates
Maps	Databases

2.5 NIS Domain

- Logical grouping
- Independent of DNS

```
domainname example-nis
```

2.6 NIS Configuration Files

File	Purpose
/etc/yp.conf	Client config
/var/yp/	Maps
/etc/nsswitch.conf	Auth lookup

2.7 NIS Security Limitations

⚠ **Not encrypted**

- Susceptible to sniffing
- Weak authentication
- Not suitable for hostile networks

2.8 When to Use NIS

- Legacy systems
- Closed, trusted networks
- Academic labs

---

3. APACHE CLUSTERING

3.1 What Is Apache Clustering?

Apache clustering is the **deployment of multiple Apache web servers** working together to provide:

- High availability
- Load sharing
- Fault tolerance
- Scalability

3.2 Why Apache Clustering Is Required

- Handle high traffic
- Avoid single point of failure
- Improve performance
- Enable rolling updates

3.3 Apache Cluster Architecture

```
Client
  ↓
Load Balancer
```

```
  ↓
--------------------
| Apache Node 1    |
| Apache Node 2    |
| Apache Node 3    |
--------------------
```

3.4 Apache Clustering Models

Model	Description
Active-Active	All nodes serve traffic
Active-Passive	Standby node
Session-based	Sticky sessions

3.5 Session Handling in Apache Clusters

- Shared storage (NFS)
- Database-based sessions
- Sticky sessions (LB)

3.6 Apache Cluster Configuration Concepts

- Identical configs
- Same document root
- Centralized logs
- Health checks

3.7 Apache Cluster Security

- TLS termination
- Secure backend communication
- Firewall segmentation
- SELinux policies

---

4. LOAD BALANCER

4.1 Definition

A **Load Balancer** distributes incoming network traffic across **multiple backend servers** to ensure:

- High availability
- Scalability
- Reliability

4.2 Load Balancer Types

Type	Description
Hardware	Dedicated appliance
Software	HAProxy, Nginx
L4	TCP/UDP
L7	HTTP/HTTPS

4.3 Load Balancing Algorithms

Algorithm	Description
Round Robin	Sequential
Least Connections	Fewest active
IP Hash	Sticky client
Weighted	Priority-based

4.4 HAProxy (Popular Linux Load Balancer)

HAProxy Architecture

```
Client
   ↓
HAProxy
   ↓
Backend Servers
```

Install HAProxy

```
yum install haproxy
```

HAProxy Configuration File

```
/etc/haproxy/haproxy.cfg
```

Basic HAProxy Config

```
frontend web
   bind *:80
   default_backend apache_nodes

backend apache_nodes
   balance roundrobin
   server web1 192.168.1.101:80 check
   server web2 192.168.1.102:80 check
```

4.5 Health Checks

- Automatic backend monitoring
- Remove failed nodes
- Improve uptime

4.6 Load Balancer Security

- TLS offloading
- Rate limiting
- DDoS protection
- Access control

4.7 Load Balancer High Availability

- Keepalived
- VRRP
- Floating IP

---

5. INTEGRATION SCENARIOS

LDAP + Apache

- Centralized authentication
- Web application SSO

LDAP + Load Balancer

- Central identity for cluster nodes

Apache Cluster + Load Balancer

- Scalable web infrastructure

---

6. LAB PERSPECTIVE (MANDATORY)

Lab 1 – LDAP

- Install OpenLDAP
- Create users/groups
- Authenticate Linux client

Lab 2 – NIS

- Configure NIS server
- Bind NIS client
- Test centralized login

Lab 3 – Apache Cluster

- Deploy two Apache nodes
- Serve same content
- Test failover

Lab 4 – Load Balancer

- Configure HAProxy
- Test traffic distribution
- Simulate node failure

---

7. SECURITY CONSIDERATIONS

- Prefer LDAP over NIS
- Encrypt authentication traffic
- Restrict directory access
- Secure backend communication
- Monitor logs continuously

---

8. EXAM & INTERVIEW POINTS

- LDAP = modern directory service
- NIS = legacy, insecure
- Apache clustering = scalability + HA
- Load balancer distributes traffic
- HAProxy is L4/L7 load balancer
- Sticky sessions manage user state

---

9. MINI CASE STUDY

Scenario:
Enterprise requires centralized authentication and scalable web services.

Solution:

- LDAP for identity management
- Apache cluster for web tier
- HAProxy for load balancing

- Secure TLS communication
- Centralized logging & monitoring

SESSION 20 – VIRTUAL MACHINE MANAGEMENT & VIRTUAL MACHINE NETWORK CONFIGURATION

---

1. VIRTUAL MACHINE MANAGEMENT

1.1 Definition

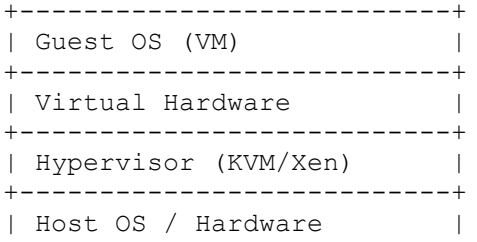
Virtual Machine (VM) Management is the process of **creating, configuring, operating, monitoring, maintaining, and optimizing virtual machines** running on a virtualization platform (hypervisor).

A **Virtual Machine** is a software-based emulation of a physical computer that runs an operating system and applications.

1.2 Why Virtual Machine Management Is Important

- Server consolidation
- Better hardware utilization
- Faster provisioning
- Isolation & security
- High availability
- Disaster recovery
- Cost reduction

1.3 Virtualization Architecture



```
+--------------------------+
```

1.4 Types of Hypervisors

Type-1 (Bare Metal)

- Runs directly on hardware
- High performance
- Used in data centers

Examples:

- VMware ESXi
- Xen
- Hyper-V

Type-2 (Hosted)

- Runs on host OS
- Used for desktops & labs

Examples:

- VirtualBox
- VMware Workstation

1.5 Linux Virtualization Stack (KVM)

```
+-------------------------+
| Virtual Machine         |
+-------------------------+
| QEMU (Emulation)        |
+-------------------------+
| KVM Kernel Module       |
+-------------------------+
| Linux Kernel            |
+-------------------------+
| Hardware (VT-x / AMD-V) |
+-------------------------+
```

1.6 Core Virtualization Components

Component	Purpose
KVM	Kernel-based Virtual Machine
QEMU	Hardware emulation
libvirt	VM management API
virt-manager	GUI tool
virsh	CLI tool

1.7 Installing Virtualization Packages

RHEL / CentOS / Rocky

```
yum install qemu-kvm libvirt virt-install virt-manager
```

Enable & Start libvirtd

```
systemctl start libvirtd
systemctl enable libvirtd
```

1.8 Virtual Machine Lifecycle

```
Create → Configure → Start → Pause → Stop → Delete
```

1.9 Creating a Virtual Machine

Using virt-install

```
virt-install \
--name vm1 \
--ram 2048 \
--vcpus 2 \
--disk size=20 \
--os-type linux \
--os-variant rhel8.0 \
--cdrom rhel.iso \
--network network=default
```

Explanation:

- `--name` → VM name
- `--ram` → Memory (MB)
- `--vcpus` → CPU cores

- `--disk` → Disk size
- `--network` → VM network

1.10 Managing Virtual Machines (virsh)

List VMs

```
virsh list
virsh list --all
```

Start / Stop / Shutdown

```
virsh start vm1
virsh shutdown vm1
virsh destroy vm1
```

Suspend / Resume

```
virsh suspend vm1
virsh resume vm1
```

Delete VM

```
virsh undefine vm1
```

1.11 VM Resource Management

CPU Allocation

- vCPU pinning

```
virsh vcpupin vm1 0 2
```

Memory Management

- Ballooning
- Dynamic memory allocation

Storage Management

- QCOW2
- RAW
- Snapshots

```
virsh snapshot-create-as vm1 snap1
```

1.12 VM Monitoring & Performance

```
virsh dominfo vm1
virsh domstats vm1
top
htop
```

1.13 VM Backup & Migration

Live Migration

```
virsh migrate --live vm1 qemu+ssh://host2/system
```

1.14 VM Management Best Practices

- Standard VM templates
- Right-size CPU & RAM
- Regular snapshots
- Patch guest OS
- Monitor performance
- Automate provisioning

---

2. VIRTUAL MACHINE NETWORK CONFIGURATION

2.1 Why VM Networking Is Important

- VM communication
- Internet access
- Service exposure
- Multi-tenant isolation

- Security enforcement

2.2 Virtual Networking Components

```
VM NIC
 ↓
Virtual Switch / Bridge
 ↓
Physical NIC
 ↓
External Network
```

2.3 Types of Virtual Networks

2.3.1 NAT (Default Network)

```
VM → NAT → Host → Internet
```

Features:

- Internet access
- No external access to VM
- Safe for labs

2.3.2 Bridged Networking

```
VM ↔ Bridge ↔ Physical Network
```

Features:

- VM gets LAN IP
- VM accessible from network
- Used in servers

2.3.3 Host-Only Network

- VM ↔ Host only

- No external access

2.3.4 Isolated Network

- VM ↔ VM only
- No host or internet

2.4 libvirt Network Types

Network	Description
default	NAT-based
bridge	Direct LAN access
macvtap	High performance
VLAN	Segmented networks

2.5 Viewing Virtual Networks

```
virsh net-list
virsh net-list --all
```

2.6 Creating a Bridge Network

Bridge Configuration (Host)

```
nmcli con add type bridge ifname br0
nmcli con add type bridge-slave ifname eth0 master br0
nmcli con up br0
```

2.7 Attaching VM to Bridge

```
virsh attach-interface \
--domain vm1 \
--type bridge \
--source br0 \
--model virtio \
--config --live
```

2.8 VM Network Interface Types

Model	Performance
e1000	Emulated
rtl8139	Legacy
virtio	High performance (recommended)

2.9 IP Addressing for VMs

- Static IP
- DHCP from:
  - libvirt
  - External DHCP server

2.10 VM Network Configuration (Guest OS)

```
ip addr
nmcli device status
```

2.11 Firewall & Security for VM Networks

- Host firewall
- VM firewall
- Network isolation
- SELinux policies
- VLAN segmentation

2.12 Troubleshooting VM Networking

Issue	Check
No IP	DHCP service
No internet	NAT/bridge
Slow network	virtio driver
Cannot reach VM	Firewall

---

3. LAB PERSPECTIVE (MANDATORY)

Lab 1 – VM Management

- Install KVM
- Create VM
- Start/stop VM
- Take snapshot

Lab 2 – NAT Networking

- Use default network
- Verify internet access

Lab 3 – Bridge Networking

- Create bridge
- Assign VM to bridge
- Access VM from LAN

Lab 4 – Network Troubleshooting

- Break VM network
- Diagnose issue
- Restore connectivity

---

4. SECURITY CONSIDERATIONS

- Isolate tenants
- Use virtio drivers
- Harden host OS
- Patch hypervisor
- Limit VM privileges
- Monitor VM traffic

---

5. PERFORMANCE BEST PRACTICES

- Use virtio
- Avoid over-commitment
- Pin CPUs for critical VMs
- Separate storage & network
- Monitor I/O & latency

---

6. EXAM & INTERVIEW POINTS

- KVM uses hardware virtualization
- libvirt manages VMs
- virsh is CLI tool
- NAT = safe, bridge = accessible
- virtio = best performance
- VM lifecycle management is critical

---

7. MINI CASE STUDY

Scenario:
Company wants to host multiple application servers with network access.

Solution:

- KVM hypervisor
- Standard VM templates
- Bridge networking for servers
- NAT for test VMs
- Monitoring & backups

SESSION 21 & 22 – BASH SHELL SCRIPTING & AUTOMATION

INTRODUCTION TO BASH

What is Bash?

Bash (Bourne Again SHell) is a **command-line interpreter and scripting language** used to:

- Interact with the Linux OS
- Automate administrative tasks
- Write reusable scripts
- Manage system operations

Why Bash is Important

- Native to Linux/Unix
- Lightweight automation
- Used in DevOps, CI/CD, system administration
- Foundation for higher automation tools (Ansible, Terraform scripts, etc.)

Shell Types

Shell	Description
sh	Bourne shell
bash	Bourne Again Shell
csh	C shell
ksh	Korn shell
zsh	Z shell

Check Current Shell

```
echo $SHELL
```

---

1. BASH CLI (COMMAND LINE INTERFACE)

1.1 Bash CLI Basics

Command Structure

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

Example:

```
ls -l /etc
```

1.2 Command History

```
history
!!        # repeat last command
!5        # run command number 5
```

1.3 Command Substitution

```
date=$(date)
echo $date
```

1.4 Aliases

```
alias ll='ls -l'
unalias ll
```

1.5 Environment Variables

```
env
printenv
export PATH=$PATH:/opt/bin
```

---

2. VARIABLES & STRINGS

2.1 Variables in Bash

Rules

- No spaces around =
- Case-sensitive
- No data types (string by default)

```
name="Linux"
count=10
```

2.2 Accessing Variables

```
echo $name
```

2.3 Read User Input

```
read username
echo "Hello $username"
```

2.4 Special Variables

Variable	Meaning
$0	Script name
$1…$9	Positional parameters
$#	Number of arguments
$@	All arguments
$?	Exit status
$$	Process ID

2.5 String Operations

```
str="LinuxAutomation"
```

Operation	Example
Length	`${#str}`
Substring	`${str:0:5}`
Replace	`${str/Linux/Bash}`

---

3. INPUT / OUTPUT REDIRECTION

3.1 Standard Streams

Stream	Description
stdin (0)	Input

Stream	Description
stdout (1)	Output
stderr (2)	Error

3.2 Output Redirection

```
ls > output.txt
ls >> output.txt
```

3.3 Error Redirection

```
ls /no_dir 2> error.txt
```

3.4 Combined Redirection

```
command > all.txt 2>&1
```

3.5 Pipes

```
ps aux | grep root
```

---

4. ERROR HANDLING IN BASH

4.1 Exit Status

- `0` → Success
- Non-zero → Failure

```
ls /tmp
echo $?
```

4.2 Manual Exit

```
exit 1
```

4.3 Conditional Error Handling

```
command || echo "Command failed"
command && echo "Command succeeded"
```

4.4 Trap (Signal Handling)

```
trap "echo Script interrupted" SIGINT SIGTERM
```

4.5 Strict Mode (Best Practice)

```
set -euo pipefail
```

Option	Meaning
-e	Exit on error
-u	Error on unset variable
-o pipefail	Pipeline error detection

---

5. DEBUGGING & SCRIPT TROUBLESHOOTING

5.1 Debug Mode

```
bash -x script.sh
```

5.2 Inline Debugging

```
set -x
commands
set +x
```

5.3 Syntax Check

```
bash -n script.sh
```

5.4 Logging in Scripts

```
echo "Process started at $(date)" >> script.log
```

---

6. CONTROL STRUCTURES & LOOPS

6.1 for Loop

```
for i in 1 2 3
do
  echo $i
done
```

6.2 for Loop with Range

```
for i in {1..5}
do
  echo $i
done
```

6.3 while Loop

```
count=1
while [ $count -le 5 ]
do
  echo $count
  ((count++))
done
```

6.4 until Loop

```
until [ $count -gt 5 ]
do
  echo $count
  ((count++))
done
```

6.5 break & continue

```
break
continue
```

---

7. CONDITIONAL STATEMENTS

7.1 if Statement

```
if [ $a -gt $b ]
then
  echo "a is greater"
fi
```

7.2 if-else

```
if [ -f /etc/passwd ]
then
  echo "File exists"
else
  echo "File missing"
fi
```

7.3 elif

```
if [ $x -eq 1 ]
then
  echo "One"
elif [ $x -eq 2 ]
then
  echo "Two"
else
  echo "Other"
fi
```

7.4 Test Conditions

Numeric

```
-eq -ne -gt -lt -ge -le
```

String

```
= != -z -n
```

File

```
-f -d -r -w -x
```

---

8. REGULAR EXPRESSIONS (REGEX)

8.1 What are Regular Expressions?

Regular expressions are **patterns used to search, match, and manipulate text**.

8.2 Regex Metacharacters

Symbol	Meaning
.	Any character
^	Start of line
$	End of line
*	Zero or more
+	One or more
?	Optional
[]	Character set
\	Escape

8.3 grep with Regex

```
grep "^root" /etc/passwd
grep "[0-9]\{3\}" file
```

8.4 Extended Regex

```
grep -E "root|admin" /etc/passwd
```

8.5 sed (Stream Editor)

```
sed 's/Linux/Bash/g' file.txt
```

8.6 awk (Pattern Scanning)

```
awk -F: '{print $1}' /etc/passwd
```

---

9. PRACTICAL LAB PERSPECTIVE (MANDATORY)

Lab 1 – Bash CLI

- Write script to display system info
- Use variables & arguments

Lab 2 – Error Handling

- Check service status
- Exit on failure

Lab 3 – Loops

- Script to create multiple users
- Loop through files

Lab 4 – Regex

- Extract usernames
- Validate IP addresses
- Replace strings in files

---

10. SECURITY & BEST PRACTICES

- Use `set -euo pipefail`
- Quote variables
- Validate user input
- Restrict script permissions

```
chmod 700 script.sh
```

---

11. EXAM & INTERVIEW POINTS

- Bash is default Linux shell
- `$?` gives exit status
- `&&` and `||` control flow
- Regex enables powerful text processing
- Loops automate repetitive tasks
- Debugging is critical for production scripts

---

12. MINI CASE STUDY

Scenario:
Admin needs to automate log cleanup and alert on errors.

Solution:

- Bash script with loops
- Regex to detect errors
- Redirection for logs
- Exit codes for monitoring
- Cron for scheduling

SESSION 23 – TASK AUTOMATION USING BASH & SECURITY PATCH MANAGEMENT

PART 1: TASK AUTOMATION USING BASH

---

1. INTRODUCTION TO TASK AUTOMATION

1.1 What is Task Automation?

Task Automation is the process of using scripts and tools to **automatically perform repetitive, scheduled, or conditional tasks** without manual intervention.

In Linux, **Bash scripting** is the most widely used automation mechanism.

1.2 Why Automation is Critical

- Reduces human error
- Saves time and effort
- Ensures consistency
- Improves reliability
- Essential for DevOps, SRE, Cloud & Security Operations

1.3 Common Automation Use Cases

Area	Examples
System Admin	User creation, cleanup
Operations	Service monitoring
Security	Log scanning, patching
Backup	Scheduled backups
DevOps	Build & deployment scripts

---

2. BASH SCRIPTING FOR AUTOMATION

2.1 Structure of a Bash Script

```
#!/bin/bash
# Script Name: example.sh
# Purpose: Sample automation script

echo "Automation started"
```

Explanation

- `#!/bin/bash` → Shebang (interpreter)
- `#` → Comment
- Executable via `chmod +x`

2.2 Making Script Executable

```
chmod +x script.sh
./script.sh
```

2.3 Using Variables in Automation

```
BACKUP_DIR="/backup"
DATE=$(date +%F)

echo "Backup directory: $BACKUP_DIR"
```

2.4 Automating File & Directory Tasks

Example: Directory Cleanup Script

```
#!/bin/bash
find /tmp -type f -mtime +7 -exec rm -f {} \;
```

Explanation

- `find` → Search files
- `-mtime +7` → Older than 7 days
- `rm -f` → Force delete

2.5 Automating User Management

Create Multiple Users

```
for user in user1 user2 user3
do
  useradd $user
  echo "Password@123" | passwd --stdin $user
done
```

2.6 Automating Service Monitoring

```
#!/bin/bash

systemctl is-active sshd >/dev/null
if [ $? -ne 0 ]; then
  systemctl start sshd
```

```
  echo "SSHD restarted"
fi
```

2.7 Automating Backups

Backup Script Using tar

```
#!/bin/bash
SRC="/etc"
DEST="/backup/etc_$(date +%F).tar.gz"

tar -czf $DEST $SRC
```

2.8 Scheduling Automation – CRON

Cron Basics

```
crontab -e
```

Field	Meaning
Minute	0–59
Hour	0–23
Day	1–31
Month	1–12
Weekday	0–7

Example: Daily Backup at 2 AM

```
0 2 * * * /root/backup.sh
```

2.9 Automation with Conditional Logic

```
if df -h | grep -q "/dev/sda1.*90%"; then
  echo "Disk usage critical"
fi
```

2.10 Automation with Functions

```
check_service() {
  systemctl status $1 >/dev/null || systemctl start $1
}
```

```
check_service httpd
```

2.11 Logging in Automation Scripts

```
LOG="/var/log/automation.log"
echo "$(date) - Task executed" >> $LOG
```

2.12 Error Handling in Automation

```
set -euo pipefail
```

Option	Purpose
`-e`	Exit on error
`-u`	Error on unset variable
`pipefail`	Catch pipeline errors

2.13 Security Best Practices for Scripts

- Avoid hard-coded passwords
- Restrict permissions (`chmod 700`)
- Validate inputs
- Use absolute paths
- Run as least-privileged user

PART 2: SECURITY PATCH MANAGEMENT

---

3. INTRODUCTION TO SECURITY PATCHES

3.1 What is a Security Patch?

A **security patch** is a **software update** designed to:

- Fix vulnerabilities
- Close security loopholes

- Prevent exploits and attacks

3.2 Why Security Patching is Critical

- Linux vulnerabilities are publicly disclosed (CVEs)
- Unpatched systems are easy targets
- Required for compliance (ISO, PCI, SOC)
- Reduces attack surface

3.3 Types of Patches

Patch Type	Description
Security	Fix vulnerabilities
Kernel	OS core fixes
Bug Fix	Stability issues
Enhancement	Feature improvements

---

4. PATCH MANAGEMENT PROCESS

4.1 Patch Management Lifecycle

```
Identify → Assess → Test → Deploy → Verify → Rollback
```

4.2 Patch Sources

- Linux distribution repositories
- Vendor advisories
- CVE databases
- Red Hat Security Advisories (RHSA)

---

5. PATCH MANAGEMENT TOOLS

5.1 YUM / DNF (RHEL-based Systems)

Check Available Updates

```
dnf check-update
```

Apply All Updates

```
dnf update -y
```

Apply Only Security Updates

```
dnf update --security
```

5.2 APT (Debian/Ubuntu)

```
apt update
apt upgrade
```

5.3 Kernel Patch Management

```
uname -r
dnf update kernel
reboot
```

⚠ Kernel patches require reboot

---

6. AUTOMATING SECURITY PATCHES

6.1 Automated Patching with Bash

```
#!/bin/bash
dnf update --security -y
if [ $? -eq 0 ]; then
  echo "Security patches applied"
else
```

```
  echo "Patch failed"
fi
```

6.2 Scheduling Security Patches

```
0 3 * * 0 /root/security_patch.sh
```

➡ Weekly security patching at 3 AM on Sunday

6.3 dnf-automatic (Recommended)

```
dnf install dnf-automatic
systemctl enable dnf-automatic.timer
```

6.4 Patch Verification

```
dnf history
dnf history info
```

6.5 Rollback & Recovery

```
dnf downgrade package-name
```

Or:

- VM snapshots
- LVM snapshots
- Backups

---

7. PATCH MANAGEMENT SECURITY CONSIDERATIONS

- Test patches in staging
- Backup before patching
- Avoid peak business hours
- Monitor services after patching
- Track CVEs

---

8. AUTOMATION + SECURITY PATCH INTEGRATION

End-to-End Automation Flow

```
Cron
 ↓
Bash Script
 ↓
Apply Security Patches
 ↓
Restart Services
 ↓
Log Results
 ↓
Send Alerts (Optional)
```

---

9. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Bash Automation

- Write backup script
- Schedule with cron
- Add logging

Lab 2 – Patch Automation

- Check pending patches
- Apply security updates
- Verify patch history

Lab 3 – Combined Automation

- Automate patching
- Restart affected services
- Generate patch report

---

10. EXAM & INTERVIEW POINTS

- Bash is core Linux automation tool
- Cron schedules tasks
- `set -euo pipefail` improves script reliability
- Security patches fix CVEs
- Kernel patches require reboot
- Automated patching reduces risk

---

11. MINI CASE STUDY

Scenario:
Enterprise Linux servers must remain secure with minimal downtime.

Solution:

- Bash-based patch automation
- Weekly security updates
- Pre-patch backups
- Logging & verification
- Rollback planning

SESSION 24 – LOGGING & MONITORING USING BASH SCRIPTS

PART A: LOGGING USING BASH SCRIPTS

---

1. INTRODUCTION TO LOGGING

1.1 What is Logging?

Logging is the process of **recording system, application, security, and operational events** in log files so they can be:

- Monitored

- Analyzed
- Audited
- Used for troubleshooting and forensics

In Bash automation, logging provides **visibility and accountability**.

1.2 Why Logging is Critical

- Detect failures early
- Track automation behavior
- Debug scripts
- Meet compliance requirements
- Provide forensic evidence after incidents

1.3 Types of Logs

Log Type	Description
System Logs	OS & kernel events
Application Logs	App-specific activity
Security Logs	Auth, sudo, SSH
Audit Logs	Compliance events
Custom Logs	Script-generated

---

2. LOGGING MECHANISMS IN BASH

2.1 Standard Streams Recap

Stream	Descriptor
stdin	0
stdout	1
stderr	2

2.2 Basic Logging with Redirection

Output Logging

```
echo "Task completed" >> /var/log/myscript.log
```

Error Logging

```
command 2>> /var/log/myscript_error.log
```

Combined Logging

```
command >> /var/log/myscript.log 2>&1
```

2.3 Timestamped Logging (Best Practice)

```
log() {
  echo "$(date '+%F %T') : $1" >> /var/log/myscript.log
}

log "Backup started"
```

2.4 Script-Wide Logging Using exec

```
exec >> /var/log/myscript.log 2>&1
```

➡ Redirects all output from script to log file

2.5 Logging Command Execution Status

```
command
if [ $? -eq 0 ]; then
  log "Command executed successfully"
else
  log "Command failed"
fi
```

2.6 Rotating Script Logs (logrotate)

```
/etc/logrotate.d/myscript
/var/log/myscript.log {
  weekly
```

```
  rotate 4
  compress
  missingok
}
```

2.7 Logging Levels

```
INFO="INFO"
WARN="WARN"
ERROR="ERROR"
log "$INFO Disk usage normal"
log "$ERROR Disk usage critical"
```

2.8 Logging Security Events

```
grep "Failed password" /var/log/secure >> /var/log/auth_alert.log
```

PART B: MONITORING USING BASH SCRIPTS

---

3. INTRODUCTION TO MONITORING

3.1 What is Monitoring?

Monitoring is the **continuous observation of system health, performance, and security** to:

- Detect anomalies
- Prevent downtime
- Trigger alerts
- Ensure SLA compliance

3.2 Types of Monitoring

Type	Description
Availability	Service up/down
Performance	CPU, memory, I/O
Capacity	Disk usage

Type	Description
Security	Unauthorized access
Process	Zombie/hung processes

---

4. SYSTEM MONITORING USING BASH

4.1 CPU Monitoring Script

```
CPU_IDLE=$(top -bn1 | awk '/Cpu/ {print $8}')
CPU_USAGE=$(echo "100 - $CPU_IDLE" | bc)

if (( $(echo "$CPU_USAGE > 80" | bc -l) )); then
  log "High CPU usage: $CPU_USAGE%"
fi
```

4.2 Memory Monitoring Script

```
MEM_FREE=$(free -m | awk '/Mem/ {print $4}')
if [ $MEM_FREE -lt 500 ]; then
  log "Low memory: ${MEM_FREE}MB"
fi
```

4.3 Disk Usage Monitoring Script

```
df -h | awk 'NR>1 {print $5 " " $6}' | while read use mount; do
  percent=${use%\%}
  if [ $percent -gt 80 ]; then
    log "Disk usage critical on $mount ($use)"
  fi
done
```

4.4 Service Monitoring Script

```
SERVICE=sshd
systemctl is-active $SERVICE >/dev/null
if [ $? -ne 0 ]; then
  systemctl start $SERVICE
  log "$SERVICE restarted"
fi
```

4.5 Process Monitoring Script

```
ps aux | grep httpd | grep -v grep || log "Apache not running"
```

4.6 Network Monitoring Script

```
ping -c 2 google.com >/dev/null || log "Network unreachable"
```

4.7 Log File Monitoring (Security)

```
tail -n 20 /var/log/secure | grep "Failed password" && log "SSH brute-force
attempt detected"
```

4.8 Uptime Monitoring

```
UPTIME=$(uptime -p)
log "System uptime: $UPTIME"
```

---

5. ALERTING FROM BASH SCRIPTS

5.1 Email Alerts

```
echo "Disk usage critical" | mail -s "Alert" admin@example.com
```

5.2 Syslog Alerts

```
logger "High CPU usage detected"
```

5.3 Exit Codes for External Monitoring

```
exit 2
```

Code	Meaning
0	OK
1	Warning
2	Critical

6. INTEGRATING LOGGING & MONITORING

Unified Monitoring Script Structure

```
#!/bin/bash
LOG="/var/log/monitor.log"

log() {
  echo "$(date '+%F %T') $1" >> $LOG
}

check_disk
check_cpu
check_memory
```

7. SECURITY & BEST PRACTICES

- Run scripts as least-privileged user
- Protect log files (`chmod 600`)
- Avoid log flooding
- Validate input values
- Rotate logs regularly

8. LAB PERSPECTIVE (MANDATORY)

Lab 1 – Logging

- Create logging function
- Log script execution
- Rotate logs

Lab 2 – Monitoring

- Monitor CPU, memory, disk
- Restart failed service

- Generate alert

Lab 3 – Security Monitoring

- Detect failed SSH attempts
- Log unauthorized access
- Trigger alerts

---

9. EXAM & INTERVIEW POINTS

- Logging = visibility & audit
- Monitoring = proactive detection
- Bash enables lightweight monitoring
- Exit codes integrate with tools
- Log rotation prevents disk exhaustion
- Alerts reduce MTTR

---

10. MINI CASE STUDY

Scenario:
Production Linux server suffers intermittent outages.

Solution:

- Bash-based monitoring scripts
- Centralized logging
- Disk & service checks
- Automated alerts
- Scheduled execution via cron