Top 50 Intermediate and Advanced Linux Interview Questions and Answers

Introduction

This guide provides 50 intermediate and advanced Linux interview questions and answers, organized by topic (Commands, File Systems, Permissions, Processes, Networking) and skill level (Intermediate and Advanced). It is designed for candidates with a solid Linux foundation, offering detailed explanations to prepare for technical interviews. Each answer includes practical examples and context to demonstrate in-depth understanding.

1 Linux Commands

1.1 Intermediate

1.1.1 1. How do you search for a file by name and execute a command on it?

The `find` command with `-exec` locates files and executes commands on them. For example, to find all `.log` files and delete them:

```
find /var/log -type f -name "*.log" -exec rm -v {} \;
```

`-type f` specifies files, `-name "*.log"` matches the pattern, and `-exec rm -v` runs `rm` on each match. The `-v` flag shows deleted files. Use `{}` to represent each found file and to end the command.

1.1.2 2. How does `grep` handle recursive searches with multiple files?

`grep -r` searches directories recursively. To find "error" in all files under `/var/log`:

```
grep -r "error" /var/log
```

Use `-i` for case-insensitive searches, `-l` to list only matching filenames, or `-include="*.log"` to filter file types. For example:

```
grep -r --include="*.log" "error" /var/log
```

This improves performance by limiting the search scope.

1.1.3 3. How do you redirect both stdout and stderr to a file?

Use `>` for stdout and `2>` for stderr. To redirect both:

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

Alternatively, use `>` (bash-specific):

```
command &> output.txt
```

This captures all output, useful for debugging scripts.

1.1.4 4. What is the difference between `tee` and redirection?

`tee` reads from stdin and writes to both a file and stdout, unlike `>` which only writes to a file. Example:

```
ls | tee file.txt
```

Use `-a` to append. `tee` requires `sudo` for protected files:

```
echo "text" | sudo tee /etc/file
```

1.1.5 5. How do you schedule a recurring task with `cron`?

Edit the crontab with `crontab -e`. A cron job has six fields: minute, hour, day of month, month, day of week, and command. Example to run `backup.sh` daily at 2 AM:

```
0 2 * * * /path/to/backup.sh
```

Verify with `crontab -l`. Logs are typically in `/var/log/syslog` or `/var/log/cron`.

1.2 Advanced

1.2.1 6. How do you use `awk` to parse complex log files?

`awk` processes text by fields. To extract the IP and status code from an Apache log:

```
awk '{print $1, $9}' access.log
```

To count occurrences of status codes:

```
awk '{count[$9]++} END {for (code in count) print code, count[code]}' access.log
```

`awk` is powerful for pattern matching and calculations in scripts.

1.2.2 7. How do you use `sed` for in-place file editing?

`sed -i` edits files directly. To replace "old" with "new" in `file.txt`:

```
sed -i 's/old/new/g' file.txt
```

Use a backup with `-i.bak` to save the original. For complex patterns, use regex:

```
sed -i '/^ERROR/s/info/warning/' file.txt
```

1.2.3 8. How do you chain commands conditionally?

Use `&&` (run if previous succeeds) or `||` (run if previous fails):

```
make && ./program || echo "Build failed"
```

For complex logic, use `if` in scripts:

```
if ! command1; then echo "Failed"; exit 1; fi
```

1.2.4 9. How do you write a shell script to monitor disk usage?

A script to alert if disk usage exceeds 80%:

```
#!/bin/bash
THRESHOLD=80
df -h | grep "/dev/sd" | awk '{print $5}' | cut -d'%' -f1 | while read usage; do
    if [ $usage -gt $THRESHOLD ]; then
        echo "Warning: Disk usage at $usage% on $(date)"
    fi
done
```

Save as `monitor.sh`, make executable (`chmod +x`), and schedule with `cron`.

1.2.5 10. How do you debug a shell script?

Use `set -x` for verbose output or `bash -x script.sh`. Add `set -e` to exit on errors. For specific lines, use `echo` for debugging:

```
echo "Variable value: $var"
```

Check exit codes with ?.

2 File Systems

2.1 Intermediate

2.1.1 11. What is the Linux filesystem hierarchy standard (FHS)?

The FHS defines directory purposes:

- `/bin`: Essential user binaries.
- `/etc`: Configuration files.
- `/var`: Variable data (logs, caches).
- `/usr`: User programs and libraries.
- `/proc`: Virtual filesystem for system info.

It ensures consistency across distributions.

2.1.2 12. How do you mount a filesystem manually?

Use mount with the device and mount point:

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

Check with df -h. Edit /etc/fstab for persistent mounts:

```
/dev/sdb1 /mnt ext4 defaults 0 2
```

2.1.3 13. What is the difference between ext4 and btrfs?

ext4 is a stable, journaled filesystem for general use. btrfs supports snapshots, compression, and subvolumes but is more complex. Example for btrfs snapshot:

```
btrfs subvolume snapshot /btrfs /btrfs/snap
```

2.1.4 14. How do you check disk usage by directory?

Use du:

```
du -sh /path/*
```

Sort by size:

```
du -h /path | sort -hr
```

2.1.5 15. What is /proc, and how is it used?

/proc is a virtual filesystem for runtime system information. Example:

```
cat /proc/cpuinfo
```

Shows CPU details. Use /proc/meminfo for memory stats.

2.2 Advanced

2.2.1 16. How do you repair a corrupted filesystem?

Use fsck on an unmounted filesystem:

```
umount /dev/sdX
fsck -y /dev/sdX
```

-y auto-fixes issues. Run in single-user mode for root filesystems.

2.2.2 17. How do you manage Logical Volume Manager (LVM)?

LVM allows dynamic volume resizing. Create a logical volume:

```
pvcreate /dev/sdb
vgcreate myvg /dev/sdb
lvcreate -L 10G -n mylv myvg
mkfs.ext4 /dev/myvg/mylv
```

Resize with lvresize.

2.2.3 18. How do you handle inode exhaustion?

Check inode usage with df -i. Delete unnecessary files or increase inodes during filesystem creation:

```
mkfs.ext4 -N 1000000 /dev/sdX
```

2.2.4 19. How do you create a swap file?

Create and enable a swap file:

```
fallocate -l 2G /swapfile
chmod 600 /swapfile
mkswap /swapfile
swapon /swapfile
```

Add to /etc/fstab for persistence.

2.2.5 20. What are filesystem quotas?

Quotas limit disk usage per user or group. Enable with quotaon and set with edquota:

```
edquota -u user
```

Requires quota package and usrquota in /etc/fstab.

3 Permissions

3.1 Intermediate

3.1.1 21. How do you set permissions numerically?

Use chmod with octal notation (read=4, write=2, execute=1). Example:

```
chmod 640 file
```

Owner: read/write (6), group: read (4), others: none (0).

3.1.2 22. What is the sticky bit, and when is it used?

The sticky bit restricts deletion in shared directories like /tmp:

```
chmod +t /tmp
```

Only the file owner, directory owner, or root can delete files.

3.1.3 23. How do you manage user groups?

Add a user to a group with usermod:

```
usermod -aG groupname username
```

Check groups with groups username.

3.1.4 24. What is sudo, and how is it configured?

sudo allows privileged command execution, configured in /etc/sudoers using visudo:

```
user ALL=(ALL) ALL
```

Grants user full sudo access.

3.1.5 25. How do you change ownership recursively?

Use alcohols chown -R:

```
chown -R user:group /path
```

3.2 Advanced

3.2.1 26. How do Access Control Lists (ACLs) work?

ACLs extend permissions for specific users or groups:

```
setfacl -m u:user:rw file
getfacl file
```

Useful for fine-grained access control beyond standard permissions.

3.2.2 27. What is the setuid bit, and what are its risks?

Setuid allows a program to run with the owners permissions:

```
chmod u+s program
```

Risks include privilege escalation if the program is insecure.

3.2.3 28. How do you audit permissions on a directory?

Use find to list permissions:

```
find /path -type d -ls
```

Filter for specific permissions:

```
find /path -type f -perm /022
```

Finds world-writable files.

3.2.4 29. How do you manage sudo without passwords?

Edit /etc/sudoers with visudo:

```
user ALL=(ALL) NOPASSWD:ALL
```

Use sparingly due to security risks.

3.2.5 30. What is SELinux, and how do you check its status?

SELinux enforces mandatory access control. Check status:

```
sestatus
```

Modify contexts with chcon or manage policies with semanage.

4 Processes

4.1 Intermediate

4.1.1 31. How do you monitor processes in real-time?

Use top or htop for an interactive view:

```
htop
```

Sort by CPU/memory or filter by user.

4.1.2 32. How do you manage process priorities?

Use nice (new processes) or renice (existing):

```
nice -n 10 command
renice 10 -p PID
```

Lower values mean higher priority.

4.1.3 33. How do you run a process in the background?

Use & or bg after Ctrl+Z:

```
command &
```

Manage with jobs and fg.

4.1.4 34. What is a daemon process?

Daemons run in the background (e.g., sshd). Managed by systemd:

```
systemctl status sshd
```

4.1.5 35. How do you find a process by name?

Use pgrep:

```
pgrep -u user process_name
```

Lists PIDs for matching processes.

4.2 Advanced

4.2.1 36. How do you trace system calls of a process?

Use strace:

```
strace -p PID
```

Tracks system calls, useful for debugging performance issues.

4.2.2 37. What are zombie processes, and how do you resolve them?

Zombies are defunct processes awaiting parent cleanup. Identify with ps aux | grep Z. Resolve by signaling the parent:

```
kill -SIGCHLD parent_PID
```

Or kill the parent if necessary.

4.2.3 38. How do you limit process resources?

Use ulimit or cgroups. Example with ulimit:

```
ulimit -u 100
```

Limits user processes to 100. For cgroups, configure via /sys/fs/cgroup.

4.2.4 39. How do you analyze process memory usage?

Use pmap or /proc/PID/maps:

```
pmap PID
```

Shows memory mappings. Use smem for detailed usage.

4.2.5 40. What is systemd, and how do you create a service?

systemd manages services. Create a service file in /etc/systemd/system/myservice.ser

```
[Unit]
Description=My Service
[Service]
ExecStart=/path/to/script.sh
Restart=always
[Install]
WantedBy=multi-user.target
```

Enable with systemctl enable myservice.

5 Networking

5.1 Intermediate

5.1.1 41. How do you test connectivity to a specific port?

Use nc:

```
nc -zv host port
```

Checks if a port is open (e.g., nc -zv example.com 80).

5.1.2 42. How do you view the routing table?

Use ip route:

```
ip route show
```

Shows routing rules and gateways.

5.1.3 43. What is /etc/hosts, and how is it used?

Maps hostnames to IPs, overriding DNS:

127.0.0.1 myhost

Useful for local testing or blocking sites.

5.1.4 44. How do you check listening ports?

Use ss or netstat:

ss -tuln

Lists TCP/UDP listening ports.

5.1.5 45. How do you configure a static IP?

Edit /etc/network/interfaces (Debian) or /etc/sysconfig/network-scripts (RHEL):

```
auto eth0
iface eth0 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    gateway 192.168.1.1
```

Restart networking with systemctl restart networking.

5.2 Advanced

5.2.1 46. How do you capture and analyze network traffic?

Use tcpdump:

tcpdump -i eth0 port 80 -w capture.pcap

Analyze with wireshark or tcpdump -r capture.pcap.

5.2.2 47. How do you configure firewall rules with iptables?

Add a rule to allow SSH:

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

Save with iptables-save > /etc/iptables/rules.v4.

5.2.3 48. How do you perform a port scan?

Use nmap:

nmap -sS host

Performs a stealth SYN scan to detect open ports.

5.2.4 49. What is netfilter, and how does it relate to iptables?

netfilter is the Linux kernel framework for packet filtering. iptables is a user-space tool to configure netfilter rules. Example:

iptables -A FORWARD -p tcp -j DROP

Drops all forwarded TCP packets.

5.2.5 50. How do you troubleshoot DNS issues?

Check /etc/resolv.conf for nameservers:

nameserver 8.8.8.8

Test with dig or nslookup:

dig example.com

Verify connectivity to the DNS server with ping or nc.

6 Practical Tips for Interviews

- **Intermediate**: Practice scripting for automation (e.g., log parsing, backups).
- **Advanced**: Understand kernel modules, systemd services, and network diagnostics.
- Be ready to debug with strace, tcpdump, or log analysis (/var/log).
- Explain commands with real-world scenarios, e.g., troubleshooting a service failure.
- Demonstrate familiarity with tools like awk, sed, and iptables.