
# 200 LINUX SCENARIO BASED Q&A

By DevOps Shack

# 200 Linux Scenario Based Questions & Answers

# DevOps Shack

**Q1. User can see (ls) a file but cannot edit it.**

**Scenario:**
User lists a file but gets Permission denied when opening it with an editor like vim or nano.

**Root Cause:**
They have **read** permission (r) but not **write** permission (w) on the file. Editing requires write access.

**Solution:**

1. Check ownership and permissions:

```
ls -l /path/to/file
```

2. Fix ownership (if needed):

```
sudo chown username:groupname /path/to/file
```

3. Or add write permission:

```
sudo chmod u+w /path/to/file
```

**Tip:**
Always use **group permissions** instead of giving **everyone** (others) write access for corporate systems.

**Q2. Accidentally deleted a config file like /etc/ssh/sshd_config.**

**Scenario:**
Important system config file is accidentally deleted, system services depending on it might break after restart.

**Root Cause:**
When you delete a file, only its metadata link is removed. If the file is open by a running service, recovery is possible.

**Solution:**

1. Find if any process is still using the deleted file:

```
lsof | grep deleted
```

2. Recover the file:

```
cp /proc/PID/fd/FD /path/to/newfile
```

3. If no running process:
    - Need to restore from backup
    - If no backup, use file system recovery tools (like extundelete).

**Tip:**
Always configure **automatic daily backups** of /etc to avoid system outage due to config loss.

---

**Q3. "Read-only file system" error when creating or modifying files.**

**Scenario:**
Commands like touch, mkdir, or editors fail with Read-only file system error.

**Root Cause:**
The file system was mounted read-only by the OS due to detecting disk or file system corruption to avoid further damage.

**Solution:**

1. Confirm filesystem is read-only:

```
mount | grep ro
```

2. Check disk errors:

```
dmesg | tail
```

3. Reboot into rescue mode and run fsck:

```
sudo fsck /dev/sdX
```

4. Allow fsck to fix errors and reboot.

**Tip:**
Monitor disk health regularly using tools like smartctl to detect hardware failure early.

---

**Q4. "Permission denied" when creating file under /tmp/.**

**Scenario:**
Even in /tmp/, user cannot create or write files and gets permission denied.

**Root Cause:**
Permissions on /tmp were accidentally changed. /tmp should have 1777 permissions (world-writable with sticky bit).

**Solution:**

1. Check permissions:

```
ls -ld /tmp
```

2. Restore correct permissions:

```
sudo chmod 1777 /tmp
```

**Tip:**
/tmp must always have 1777 permissions; without it, apps and users can face weird errors.

---

**Q5. "No such file or directory" when executing a binary.**

**Scenario:**
A binary file exists, but running it gives No such file or directory.

**Root Cause:**
Either missing dependent shared libraries, or wrong binary compiled for another architecture.

**Solution:**

1. Check file architecture:

```
file ./binaryfile
```

2. Check dynamic dependencies:

```
ldd ./binaryfile
```

3. Install missing libraries if needed.

**Tip:**
Always compile binaries on the same OS and architecture when preparing for production.

---

**Q6. Deleted /var/log/messages log file by mistake.**

**Scenario:**
Important system logs are removed; logging services may fail.

**Root Cause:**
Services like rsyslogd or journald rely on files under /var/log.

**Solution:**

1. Restart syslog service:

```
sudo systemctl restart rsyslog
```

2. Log files will be recreated automatically.

**Tip:**
Avoid manually deleting log files; use logrotate to manage logs safely.

---

**Q7. "Text file busy" when replacing a binary.**

---

**Scenario:**
Trying to overwrite an in-use binary results in Text file busy error.

**Root Cause:**
The binary is still being executed by a running process.

**Solution:**

1. Identify process:

```
fuser binaryfile
```

2. Kill the process:

```
kill -9 PID
```

3. Replace binary safely.

**Tip:**
Use safer deployment methods like versioned binaries + symlinks to avoid runtime issues.

---

**Q8. Cannot run a newly created shell script — "Permission denied".**

**Scenario:**
Created a script but getting Permission denied even though it exists.

**Root Cause:**
Script is missing executable (x) permission.

**Solution:**

1. Make it executable:

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

2. Then run:

```
./script.sh
```

**Tip:**
Always verify both the shebang (#!/bin/bash) and permissions when troubleshooting scripts.

---

**Q9. User can't access their own home directory.**

**Scenario:**
User gets "Permission denied" when trying to cd into their home.

**Root Cause:**
Wrong ownership or permission settings on the home directory.

**Solution:**

1. Fix ownership:

```
sudo chown username:username /home/username
```

2. Fix permissions:

```
chmod 700 /home/username
```

**Tip:**
Corporate servers often set /home/username with 700 or 750 permissions for security.

---

**Q10. Mistakenly set wrong permission on /etc/passwd, users can't login.**

**Scenario:**
Users are unable to login after wrong permissions were set on /etc/passwd.

**Root Cause:**
Critical system files like /etc/passwd must have proper permissions.

**Solution:**

1. Boot into recovery mode.
2. Correct permissions:

```
chmod 644 /etc/passwd
```

**Tip:**
Use ls -l carefully when changing permissions on system files; mistakes can break authentication.

**Q11. Created a new directory but can't cd into it — "Permission denied".**

**Scenario:**
After creating a directory, the user cannot access it via cd, even though they own it.

**Root Cause:**
Directory is missing **execute (x) permission**, which is required to enter (cd) a directory.

**Solution:**

1. Check permissions:

```
ls -ld /path/to/dir
```

2. Add execute permission:

```
chmod +x /path/to/dir
```

**Tip:**
Directories need **execute** permission to enter, and **read** permission to list their contents.

---

**Q12. A symbolic link shows "No such file" after reboot.**

**Scenario:**
A previously working symlink is now broken, showing "No such file or directory".

**Root Cause:**
The symlink points to a file that was either on a temporary filesystem (e.g., /tmp) or deleted.

**Solution:**

1. Check where symlink points:

```
ls -l /path/to/symlink
```

2. Correct the target if needed:

```
ln -sf /correct/target/path /path/to/symlink
```

**Tip:**
Prefer absolute paths when creating symlinks for important files.

---

**Q13. cp -r fails midway when copying a large directory.**

**Scenario:**
Copying a huge directory with cp -r stops with errors like "No space left on device" or permission errors.

**Root Cause:**

- Disk may be full.
- Permission issues on some files.
- Filesystem limits like inode exhaustion.

**Solution:**

1. Check disk space:

df -h

2. Check inode usage:

df -i

3. If permission errors:

sudo cp -rp source/ destination/

(-p preserves ownerships.)

**Tip:**
When copying critical data, use rsync -a instead of cp -r for better control and resume support.

---

**Q14. Tar backup created but getting error while extracting.**

**Scenario:**
tar -xvf backup.tar throws errors like "Unexpected EOF" or "corrupt archive".

**Root Cause:**

- Tarball creation was interrupted.
- File got partially copied.
- Wrong tar command or compression mismatch.

**Solution:**

1. Check if tar is readable:

```
tar -tvf backup.tar
```

2. Ignore bad blocks if minor damage:

```
tar --ignore-failed-read -xvf backup.tar
```

**Tip:**
Always use gzip or bzip2 (tar -czvf) for compression + integrity checks.

---

**Q15. Filesystem showing used space but no visible files.**

**Scenario:**
df -h shows disk full but ls -lh shows no big files.

**Root Cause:**
Deleted files are still held by running processes and occupying space.

**Solution:**

1. Find deleted files:

```
lsof | grep deleted
```

2. Kill the processes using those files:

```
kill PID
```

3. Disk space will be freed.

**Tip:**
On production, avoid killing critical daemons; restart them gracefully if possible.

---

**Q16. "Argument list too long" when deleting thousands of files.**

**Scenario:**
Running rm *.log on a folder with 100,000+ files throws "Argument list too long" error.

**Root Cause:**
Shell has a maximum command-line argument length (around 128KB).

**Solution:**
Use find instead of wildcards:

```
find . -name "*.log" -delete
```

**Tip:**
For mass file operations, find is always more reliable and scalable.

---

**Q17. Accidentally moved files to the wrong location.**

**Scenario:**
mv * /wrong/path/ by mistake; files need to be moved back.

**Root Cause:**
Wrong destination path supplied in mv command.

**Solution:**
Move back:

```
find /wrong/path/ -type f -exec mv {} /correct/path/ \;
```

Or manually move using mv.

**Tip:**
Use mv -i (interactive) to avoid accidental overwrites.

---

**Q18. User created directory but team members can't write into it.**

**Scenario:**
One user creates a project folder, but others cannot create files inside it.

**Root Cause:**
Group write permissions missing.

**Solution:**

1. Set correct group ownership:

```
sudo chown :projectgroup /path/to/projectdir
```

2. Add group write permission:

```
chmod g+w /path/to/projectdir
```

**Tip:**
Use **setgid bit** (chmod g+s) on shared directories to automatically inherit group ownership for new files.

---

**Q19. Finding out which file is filling up the disk.**

**Scenario:**
Disk is almost full but unsure which files/folders are consuming space.

**Root Cause:**
Hidden huge logs, database dumps, or backup folders.

**Solution:**
Find largest directories/files:

```
du -ahx / | sort -rh | head -20
```

or

```
ncdu /
```

(ncdu gives an interactive disk usage view.)

**Tip:**
Schedule weekly disk audits in production servers to prevent sudden disk full incidents.

---

**Q20. Mounted NFS share shows "stale file handle" error.**

**Scenario:**
Accessing NFS share gives:

nginx

Stale file handle

**Root Cause:**
The file/folder on the server was deleted or modified while client still holds a reference.

**Solution:**
Force unmount:

```
sudo umount -f /mnt/nfs
```

Then remount:

```
sudo mount /mnt/nfs
```

**Tip:**
Avoid hardcoding paths on NFS-mounted systems; handle stale handles with retry logic if scripting.

**Q21. System becomes very slow — commands take long to execute.**

**Scenario:**
Basic commands like ls, cd, or top take 5–10 seconds to respond.

**Root Cause:**

- System may be out of **free RAM** and using **swap heavily**.
- High CPU or disk I/O wait also possible.

**Solution:**

1. Check memory usage:

```
free -h
```

2. Check swap usage:

```
swapon --show
```

3. Check CPU and load:

```
top
```

```
iostat -xz 1
```

**Tip:**
Set up swapoff -a temporarily to test if swap is the cause; add more RAM or reduce memory-hungry apps.

---

**Q22. Disk is 100% full, can’t create or write files.**

**Scenario:**
App logs stop, can't save files, or services crash. df -h shows 100% usage.

**Root Cause:**
A large log, backup file, or temporary file has filled the disk.

**Solution:**

1. Find the largest files:

```
du -sh /* | sort -h
```

```
or use ncdu /
```

2. Remove or compress large files:

```
gzip huge-log.log
```

**Tip:**
Configure log rotation (logrotate) to prevent runaway log files.

---

**Q23. Disk shows 0% free in df -h, but du -sh shows little usage.**

**Scenario:**
Disk shows full, but there are no large files in sight.

**Root Cause:**
Files were **deleted** but are still held open by a running process.

**Solution:**

1. Find such processes:

```
lsof | grep deleted
```

2. Restart the process or:

```
kill -9 <PID>
```

**Tip:**
Always restart logging services after clearing large logs.

---

**Q24. "Cannot allocate memory" error when starting services.**

---

**Scenario:**
A service fails to start due to memory allocation errors.

**Root Cause:**
Out of RAM or **ulimit** limits exceeded.

**Solution:**

1. Check available RAM:

```
free -m
```

2. Check and increase memory limits:

```
ulimit -a
```

```
ulimit -v unlimited
```

**Tip:**
Set permanent limits in /etc/security/limits.conf for critical services.

---

**Q25. Load average is high but CPU usage is low.**

**Scenario:**
top shows load average > 5, but CPU idle is still high.

**Root Cause:**
Load average includes processes waiting for **disk** or **I/O**, not just CPU.

**Solution:**

1. Check I/O wait:

```
iostat -xz 1
```

2. Check blocked processes:

```
vmstat 1
```

**Tip:**
Use SSDs and proper disk schedulers for high-I/O workloads.

---

**Q26. Swap usage is always high even with free RAM.**

**Scenario:**
free -h shows high swap usage even when RAM is available.

**Root Cause:**
Kernel may have aggressively swapped out old memory pages.

**Solution:**

1. Reduce swappiness:

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

2. Persist it in /etc/sysctl.conf:

```
vm.swappiness=10
```

**Tip:**
Swappiness of 10 is ideal for most server workloads.

---

**Q27. Disk I/O spikes suddenly — application slows down.**

**Scenario:**
Random performance lag. iostat shows high %util and await.

**Root Cause:**
Too many read/write operations from one or more processes.

**Solution:**

1. Use iotop to identify culprit:

```
sudo iotop
```

2. Pause or throttle process:

```
ionice -c2 -n7 -p <PID>
```

**Tip:**
Use dedicated disk partitions for logs or data-heavy apps.

---

**Q28. "No space left on device" but df -h shows free space.**

---

**Scenario:**
df -h shows free space but still can't write to disk.

**Root Cause:**
Ran out of **inodes** (metadata structure used for files).

**Solution:**

1. Check inode usage:
2. Clean up folders with many small files (like /var/spool, /tmp):

```
find /var/spool -type f -delete
```

**Tip:**
Format file systems with more inodes if you expect millions of tiny files (e.g., mail servers).

---

**Q29. top shows one process using 99% CPU.**

**Scenario:**
System is slow, one process is hogging the CPU.

**Root Cause:**
An infinite loop, runaway script, or bad query.

**Solution:**

1. Identify process:

top

ps aux --sort=-%cpu | head

2. Pause or kill process:

kill -STOP PID

kill -9 PID

**Tip:**
Use nice or cpulimit for non-critical CPU-bound processes.

---

**Q30. df command hangs or freezes.**

**Scenario:**
Running df -h just hangs and never returns.

**Root Cause:**
A mounted NFS or remote filesystem is unavailable.

**Solution:**

1. List all mounts:

mount | grep nfs

2. Force unmount:

sudo umount -f /mnt/path

**Tip:**
Mount NFS with soft,timeo options to prevent such hangs.

---

**Q31. Memory usage slowly increases over time.**

**Scenario:**
RAM usage creeps up daily until system swap is used.

**Root Cause:**
**Memory leak** in a service or cron job.

**Solution:**

1. Monitor usage:

top

```
ps aux --sort=-%mem
```

2. Restart the leaking service periodically.

**Tip:**
Use monitoring tools like prometheus + grafana to track memory trends.

---

**Q32. Kernel OOM (Out of Memory) kills processes.**

---

**Scenario:**
Processes suddenly die and dmesg shows OOM killer.

**Root Cause:**
System ran out of memory and the kernel killed the biggest consumer.

**Solution:**

1. View logs:

```
dmesg | grep -i kill
```

2. Tune OOM behavior:

```
echo -17 > /proc/<PID>/oom_score_adj
```

**Tip:**
Use OOM protection for critical apps like databases using systemd or memory cgroups.

---

**Q33. Disk write speed is very low.**

**Scenario:**
Copying large files is taking unusually long time.

**Root Cause:**

- Disk I/O contention.
- Wrong disk scheduler.
- Fragmentation.

**Solution:**

1. Benchmark:

```
dd if=/dev/zero of=testfile bs=1G count=1 oflag=dsync
```

2. Check scheduler:

```
cat /sys/block/sdX/queue/scheduler
```

**Tip:**
Use deadline or noop scheduler for SSDs.

---

**Q34. Filesystem corruption detected on reboot.**

**Scenario:**
On boot, system enters emergency mode with fsck errors.

**Root Cause:**
Improper shutdown or hardware issues caused fs corruption.

**Solution:**

1. Enter rescue mode.
2. Run fsck manually:

```
fsck /dev/sdaX
```

**Tip:**
Always unmount disks cleanly or use journaling filesystems like ext4.

---

**Q35. CPU always runs at 100% on all cores.**

**Scenario:**
Even idle system shows high CPU usage on all cores.

**Root Cause:**
Background process like cron, rsync, or malware.

**Solution:**

1. Inspect with:

```
top
```

```
htop
```

```
ps aux --sort=-%cpu
```

2. Kill the culprit or analyze further.

**Tip:**
Use auditd or psacct to track rogue activity.

---

**Q36. Multiple zombie processes accumulate.**

**Scenario:**
ps aux shows <defunct> processes.

**Root Cause:**
Parent process failed to wait for child termination.

**Solution:**

1. Identify parent:

```
ps -ef | grep defunct
```

2. Restart or kill parent process.

**Tip:**
Zombie processes don't consume resources but too many indicate bad code logic.

---

**Q37. RAM usage shown as full, but system is fast.**

**Scenario:**
free -h shows most of the memory used, but no lag.

**Root Cause:**
Linux caches file system I/O in memory.

**Solution:**
Check actual usage:

```
free -h
```

Look at "available", not "used".

**Tip:**
Don't panic if RAM is full — Linux intelligently caches it.

---

**Q38. RAM is free, but app complains "cannot allocate memory".**

**Scenario:**
App errors out despite RAM being available.

**Root Cause:**
Per-process memory limit (ulimit) or overcommit policy.

**Solution:**

1. Increase limits:

```
ulimit -v unlimited
```

2. Tweak overcommit:

```
echo 1 > /proc/sys/vm/overcommit_memory
```

**Tip:**
Avoid setting overcommit=2 for memory-heavy applications.

---

**Q39. Application performance drops when copying files.**

**Scenario:**
App slows down whenever a large file is copied.

**Root Cause:**
Disk I/O contention; both app and copy operation use same disk.

**Solution:**

1. Use ionice for background copy:

```
ionice -c3 cp file /target/
```

**Tip:**
Use dedicated storage for app vs backups/logs.

---

**Q40. CPU usage randomly spikes — can't trace.**

**Scenario:**
System becomes slow randomly, CPU usage goes to 100% for a few minutes.

**Root Cause:**

- cron jobs
- Background scanning (like updatedb, mlocate, antivirus)

**Solution:**
Check cron:

```
cat /etc/crontab
```

Check logs:

```
journalctl --since "10 min ago"
```

**Tip:**
Disable or schedule heavy cron jobs during off-peak hours.

**Q41. Ping to external websites fails, but internal network is reachable.**

---

**Scenario:**
You can ping other internal servers, but external websites like google.com fail.

**Root Cause:**
Likely **missing or misconfigured DNS settings**.

**Solution:**

1. Check DNS in /etc/resolv.conf:

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

2. Add valid DNS (e.g., Google DNS):

```
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf
```

**Tip:**
For permanent DNS, configure via /etc/systemd/resolved.conf or /etc/netplan/ (Ubuntu) or /etc/sysconfig/network-scripts/ (RHEL).

---

**Q42. SSH connection takes 30 seconds to respond.**

**Scenario:**
SSH login delays for 30 seconds, then succeeds.

**Root Cause:**
Reverse DNS lookup delay or GSSAPI authentication hang.

**Solution:**

1. Edit SSH config:

```
sudo vi /etc/ssh/sshd_config
```

Add or update:

nginx

UseDNS no

GSSAPIAuthentication no

2. Restart SSH:

sudo systemctl restart sshd

**Tip:**
Disabling UseDNS is a common corporate tweak to speed up SSH logins.

---

**Q43. SSH fails with "Permission denied (publickey)".**

**Scenario:**
Trying to login via SSH with key, but always gets Permission denied (publickey).

**Root Cause:**

- Wrong key
- Missing authorized_keys file
- Incorrect permissions

**Solution:**

1. Ensure correct key is in:

~/.ssh/authorized_keys

2. Set proper permissions:

chmod 700 ~/.ssh

chmod 600 ~/.ssh/authorized_keys

**Tip:**
Always verify server-side logs:

---

**Q44. SSH server is running but port 22 is not reachable from outside.**

**Scenario:**
systemctl status sshd shows active, but remote users can't connect.

**Root Cause:**
Firewall or security group (cloud) blocking port 22.

**Solution:**

1. Check port open on local server:

```
sudo ss -tuln | grep :22
```

2. Check firewall:

```
sudo ufw status   # or sudo iptables -L
```

3. Open port 22:

```
sudo ufw allow 22
```

```
sudo systemctl restart sshd
```

**Tip:**
In AWS/GCP, check security group settings in addition to the OS firewall.

---

**Q45. "Connection refused" when trying to SSH into a server.**

**Scenario:**
SSH fails with connection refused immediately.

**Root Cause:**
SSHD not running, or nothing is listening on port 22.

**Solution:**

1. Start SSH server:

```
sudo systemctl start sshd
```

2. Ensure it's enabled at boot:

```
sudo systemctl enable sshd
```

**Tip:**
Always check ss -tuln to verify if any service is bound to port 22.

---

**Q46. Server has no internet access, but IP and DNS look fine.**

**Scenario:**
ping 8.8.8.8 and curl fail, but IP and DNS are configured.

**Root Cause:**
Missing or incorrect **default gateway** route.

**Solution:**

1. View routing table:

```
ip route show
```

2. Add default gateway:

```
sudo ip route add default via 192.168.1.1
```

**Tip:**
Always validate both DNS and default route to confirm full internet access.

---

**Q47. Network goes down after reboot.**

**Scenario:**
After rebooting, server comes up without IP address or fails to connect.

**Root Cause:**
Static IP config missing or not applied properly in network configs.

**Solution (Ubuntu - netplan):**

1. Edit config:

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

2. Apply changes:

```
sudo netplan apply
```

**Tip:**
Always validate netplan configs with netplan try to prevent lockout.

---

**Q48. Firewall blocks traffic after reboot even though ports were allowed.**

**Scenario:**
Rules worked before reboot, but after restart, services are unreachable.

**Root Cause:**
Firewall rules were not **persisted**.

**Solution:**

1. On Ubuntu with ufw:

```
sudo ufw enable
```

```
sudo ufw allow 80
```

2. On iptables-based systems:

```
sudo iptables-save > /etc/iptables/rules.v4
```

**Tip:**
Use iptables-persistent or firewalld's --permanent flag to save rules.

---

**Q49. Can't SSH as root user even though password is correct.**

**Scenario:**
Login as root gives access denied, but other users can login.

**Root Cause:**
Root SSH login disabled by default in many distributions.

**Solution:**

1. Edit /etc/ssh/sshd_config:

```
PermitRootLogin yes
```

2. Restart SSH:

```
systemctl restart sshd
```

**Tip:**
Allow root login only for emergency. Use regular users + sudo.

---

**Q50. Port is open locally but not reachable from another server.**

**Scenario:**
App listens on port 3000, reachable via localhost, but not externally.

**Root Cause:**
Application bound to 127.0.0.1, not 0.0.0.0.

**Solution:**

1. Check with:

```
sudo ss -tuln
```

2. Change app binding:
    - From: 127.0.0.1:3000
    - To: 0.0.0.0:3000
3. Restart the app.

**Tip:**
For Node.js, Flask, etc., always bind to 0.0.0.0 for external access.

---

**Q51. Port 443 shows open, but HTTPS doesn't work.**

**Scenario:**
Website doesn't load over https://, even though port 443 is open.

**Root Cause:**
SSL certificate missing, expired, or web server misconfigured.

**Solution:**

1. Check certificate:

```
openssl s_client -connect yourdomain.com:443
```

2. Check Nginx/Apache SSL config.
3. Renew cert:

```
sudo certbot renew
```

**Tip:**
Use a cron job or systemd timer to automate SSL renewal with Certbot.

---

**Q52. SSH sessions randomly freeze or timeout.**

---

**Scenario:**
SSH session hangs during idle or long data transfer.

**Root Cause:**
Firewall NAT timeout or session keepalive not configured.

**Solution:**

1. Add to /etc/ssh/sshd_config:

```
ClientAliveInterval 60
ClientAliveCountMax 3
```

2. Restart sshd.

**Tip:**
Also set ServerAliveInterval in client ~/.ssh/config.

---

**Q53. Server cannot resolve internal hostname.**

**Scenario:**
ping myapp.internal fails, even though IP is correct.

**Root Cause:**
Missing or incorrect /etc/hosts entry or broken DNS resolver.

**Solution:**

1. Add entry manually:

echo "10.0.1.5 myapp.internal" | sudo tee -a /etc/hosts

2. Or fix internal DNS resolution.

**Tip:**
Avoid hardcoding hosts for dynamic services. Use proper internal DNS.

---

**Q54. Network interface name changed after reboot.**

**Scenario:**
After upgrade or reboot, eth0 becomes ens33 or similar.

**Root Cause:**
Predictable Network Interface Names (systemd/udev feature).

**Solution:**
Use ip a to confirm interface, then fix config.

To revert:

sudo ln -s /dev/null /etc/udev/rules.d/80-net-setup-link.rules

**Tip:**
Stick with default naming unless you have automation depending on eth0.

---

**Q55. Server reachable only via IP, not domain name.**

**Scenario:**
You can ping IP but not the domain.

**Root Cause:**
DNS resolution issue.

**Solution:**

1. Check:

dig yourdomain.com

nslookup yourdomain.com

2. Fix /etc/resolv.conf, or update DNS.

**Tip:**
Always check for DNS TTL expiration if domain recently changed.

---

**Q56. Connection refused from one server, but works from another.**

**Scenario:**
SSH or HTTP works from server A, fails from server B.

**Root Cause:**
Firewall is **blocking specific IP** or subnet.

**Solution:**

1. Check firewall:

sudo iptables -L

sudo ufw status

2. Allow that specific IP:

sudo ufw allow from <IP>

**Tip:**
Use centralized firewall management in corporate (e.g., firewalld zones or cloud firewalls).

---

**Q57. iptables rules lost after reboot.**

**Scenario:**
Custom firewall rules vanish after reboot.

**Root Cause:**
iptables rules not saved to persistent storage.

**Solution:**

1. Install persistence:

sudo apt install iptables-persistent

sudo netfilter-persistent save

**Tip:**
Always use iptables-save and iptables-restore in production.

---

**Q58. Public IP changed after reboot (cloud server).**

**Scenario:**
SSH fails after reboot; public IP is different.

**Root Cause:**
Using **ephemeral public IP** not an elastic/static one.

**Solution:**

- In AWS/GCP: allocate **static IP** and attach to instance.

**Tip:**
Always assign static public IP for production servers.

---

**Q59. SSH key authentication fails but password works.**

**Scenario:**
Password SSH works, but key-based login fails.

**Root Cause:**
sshd_config disables PubkeyAuthentication.

**Solution:**

1. Edit /etc/ssh/sshd_config:

PubkeyAuthentication yes

2. Restart SSH:

systemctl restart sshd

**Tip:**
Verify correct key is in ~/.ssh/authorized_keys with chmod 600.

---

**Q60. After changing hostname, SSH gives "WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!"**

**Scenario:**
SSH gives security warning after host reinstallation or hostname change.

**Root Cause:**
SSH key mismatch; host key changed, but client stored old one.

**Solution:**

1. On client:

```
ssh-keygen -R server_ip
```

2. Try again:

```
ssh user@server_ip
```

**Tip:**
This is expected after reinstallation. Don’t ignore blindly in real SSH MITM risks.

---

**Q61. Application hangs, doesn't respond to input, but still runs in ps.**

**Scenario:**
App seems frozen. It's listed in ps, but no output or interaction possible.

**Root Cause:**
App is **stuck in an uninterruptible state** (D), usually due to I/O wait.

**Solution:**

1. Check process state:

```
ps -eo pid,stat,cmd | grep '^ *PID'
```

State D = Disk sleep (uninterruptible)

2. Check disk usage:

```
iotop
```

```
dmesg | tail
```

**Tip:**
Restarting the application is often the only fix if it's blocked on I/O.

---

**Q62. High number of defunct (zombie) processes.**

**Scenario:**
ps aux shows many processes with <defunct> status.

**Root Cause:**
Parent process isn’t collecting the exit status of child processes.

**Solution:**

1. Find parent:

```
ps -ef | grep defunct
```

2. Restart the parent:

```
kill -HUP <PARENT_PID>
```

**Tip:**
Well-coded apps should handle child exits with wait(); monitor bad processes.

---

**Q63. A process refuses to die using kill.**

**Scenario:**
App doesn't stop after kill PID. Even kill -9 doesn't help.

**Root Cause:**
Kernel marks process as **uninterruptible sleep** — stuck on I/O or kernel-level operation.

**Solution:**

1. Confirm status:

```
ps -o pid,stat,cmd -p <PID>
```

2. Only fix:

Reboot the system

**Tip:**
This is rare but happens with disk/NFS issues — fix underlying I/O after reboot.

---

**Q64. You started a long-running command and accidentally closed the terminal.**

**Scenario:**
You closed the terminal, and the command stopped.

**Root Cause:**
Foreground processes are terminated when the controlling terminal exits.

**Solution:**
Next time:

- Use nohup:

```
nohup long-running-command &
```

- Or use screen / tmux for detachable terminals.

**Tip:**
For production, always use terminal multiplexers for stability.

---

**Q65. CPU is stuck at 100%, system lagging heavily.**

**Scenario:**
System nearly unresponsive, fans running loud, all CPU cores maxed.

**Root Cause:**
One or more processes are consuming full CPU in an infinite loop or runaway thread.

**Solution:**

1. Identify with:

```
top -o %CPU
```

2. Kill the culprit:

```
kill -9 <PID>
```

**Tip:**
For permanent control, use cpulimit or nice for background tasks.

---

**Q66. Service crashes with segmentation fault (core dumped).**

**Scenario:**
A command crashes and prints: Segmentation fault (core dumped)

**Root Cause:**
Memory violation due to bad code or incompatible library.

**Solution:**

1. Enable core dumps:

```
ulimit -c unlimited
```

2. Analyze core:

```
gdb /path/to/bin core
```

**Tip:**
Recompile with debug symbols for deeper insight. Common in custom or C/C++ apps.

---

**Q67. A script keeps running indefinitely and doesn't stop.**

---

**Scenario:**
Shell script keeps running without output or finishing.

**Root Cause:**
Possible infinite loop or read command waiting for user input.

**Solution:**

1. Use strace to observe:

strace -p <PID>

2. Modify script to add debug logs (set -x, echo).

**Tip:**
Always put timeout logic in critical automation scripts.

---

**Q68. Cron job runs but nothing happens.**

**Scenario:**
Job is listed in crontab, marked as run in logs, but no output or action.

**Root Cause:**

- Missing environment variables (e.g., PATH)
- Script paths incorrect when run from cron

**Solution:**

1. Add full paths to all commands inside the script.
2. Redirect output to a log file:

* * * * * /path/to/script.sh > /tmp/script.log 2>&1

**Tip:**
Use env at the top of your script to debug cron job environments.

---

**Q69. Systemd service fails to start even though the binary works manually.**

**Scenario:**
Manual execution is fine, but systemctl start myapp fails.

**Root Cause:**
Wrong user, missing Environment, or incorrect WorkingDirectory in unit file.

**Solution:**

1. Check service logs:

```
journalctl -u myapp.service
```

2. Fix unit file:

```
ExecStart=/full/path/to/binary
WorkingDirectory=/path
User=appuser
```

**Tip:**
Run systemd-analyze verify file.service to lint systemd files.

---

**Q70. You can’t stop a service via systemd — says "failed".**

**Scenario:**
systemctl stop myapp fails with timeout or “failed to stop unit”.

**Root Cause:**
The process does not respond to SIGTERM or leaves behind child processes.

**Solution:**

1. Force kill:

```
systemctl kill -s SIGKILL myapp
```

2. Then restart:

```
systemctl restart myapp
```

**Tip:**
In unit file, add:

```
KillMode=control-group
```

To ensure child processes are killed too.

---

**Q71. ps shows a process but you can't find the command or script behind it.**

**Scenario:**
You see a process by PID but don't know what it is or what script it runs.

**Root Cause:**
Might be a forked shell, background script, or binary without obvious path.

**Solution:**

1. Use:

```
ps -p PID -o cmd
```

```
lsof -p PID
```

```
readlink /proc/PID/exe
```

**Tip:**
Combine strace with lsof for live debugging.

---

**Q72. Process runs fine but doesn't log any output.**

**Scenario:**
App runs, works, but you see no logs anywhere.

**Root Cause:**

- Stdout/stderr redirected to /dev/null
- Logging not configured

**Solution:**

1. Run in foreground:

```
./app
```

2. Check logging path in config.
3. Check systemd logs:

```
journalctl -u app
```

**Tip:**
Always configure both **file-based** and **journald-based** logging in production.

---

**Q73. You need to run multiple processes in background from a script.**

**Scenario:**
Script launches 3–4 background jobs, but only the first runs.

**Root Cause:**
Process chaining or incorrect use of &, wait.

**Solution:**
Use:

```
command1 &
command2 &
```

**Tip:**
Use trap to clean up if script is interrupted.

---

**Q74. A scheduled process keeps getting killed randomly.**

**Scenario:**
A background task or cronjob dies midway.

**Root Cause:**
OOM killer, memory limit, or wrong ulimit settings.

**Solution:**

1. Check:

```
dmesg | grep -i kill
```

2. Set memory limits properly:

```
ulimit -v unlimited
```

**Tip:**
Use cgroups for more granular resource control in critical systems.

---

**Q75. System boots, but one service fails to start automatically.**

**Scenario:**
Manual start works, but service doesn't auto-start at boot.

**Root Cause:**
Service is not **enabled**, or has incorrect After= dependency.

**Solution:**

1. Enable it:

```
systemctl enable myservice
```

2. Check dependency timing in unit:

```
After=network.target
```

**Tip:**
Use systemd-analyze blame to see what's blocking boot.

---

**Q76. Process binds to port, but logs say "address already in use".**

**Scenario:**
App fails to start, error: EADDRINUSE.

**Root Cause:**
Another process is using the port.

**Solution:**

1. Find the PID:

```
sudo lsof -i :PORT
```

2. Kill or reconfigure that process.

**Tip:**
Use ss -tuln to list all open ports before binding new services.

---

**Q77. Process always crashes after 1 minute of running.**

**Scenario:**
Service starts, runs fine, then crashes after 60 seconds.

**Root Cause:**
Could be systemd **WatchdogTimeout**, app memory leak, or config reload failure.

**Solution:**

1. Check logs:

```
journalctl -xe
```

2. Inspect unit file WatchdogSec=...
3. Run manually to confirm if app crashes outside systemd too.

**Tip:**
Avoid hardcoded timers unless watchdogs are explicitly needed.

---

**Q78. Need to ensure a script auto-restarts on crash.**

**Scenario:**
You want a script to always restart if it crashes.

**Root Cause:**
Default shell scripts exit on failure and aren't managed.

**Solution:**
Wrap in a systemd service:

```
[Service]
ExecStart=/path/to/script.sh
Restart=always
RestartSec=5
```

Enable the service.

**Tip:**
This is more reliable than while true loops.

---

**Q79. Background process blocks terminal until killed.**

**Scenario:**
You ran something with &, but it still ties up your terminal.

**Root Cause:**
It still outputs to the terminal or reads input.

**Solution:**

1. Use:

```
nohup yourcommand > /dev/null 2>&1 &
```

disown

**Tip:**
Use screen or tmux to avoid such issues entirely.

---

**Q80. Process forks too many threads and slows down system.**

**Scenario:**
You notice 1000+ threads from one app, slowing things down.

**Root Cause:**
Thread leak or wrong concurrency setting.

**Solution:**

1. Check thread count:

```
ps -eLf | grep appname | wc -l
```

2. Reduce threads in app config.

**Tip:**
Use ulimit -u to cap total user-level processes and threads.

**Q81. Server boots into emergency mode.**

---

**Scenario:**
After reboot, the system shows Welcome to emergency mode and drops to a shell.

**Root Cause:**

- Fstab has invalid or missing mount entry
- Critical system file/directory is missing or corrupted

**Solution:**

1. Check fstab:

```
cat /etc/fstab
```

2. Comment the problematic mount (e.g., external drive), then:

```
mount -a
```

reboot

**Tip:**
Always use nofail in /etc/fstab for non-critical mounts to avoid boot failures.

---

**Q82. Kernel panic on boot: "Unable to mount root fs".**

**Scenario:**
System shows a kernel panic and reboots or halts immediately.

**Root Cause:**
The kernel cannot find or mount the root filesystem.

**Solution:**

1. Boot from a live CD or rescue mode.
2. Check and repair:

```
fsck /dev/sdXn
```

3. Rebuild initramfs:

```
chroot /mnt
update-initramfs -u
```

**Tip:**
Keep a rescue ISO or recovery snapshot ready for fast recovery.

---

**Q83. Grub menu doesn’t show up at boot.**

**Scenario:**
System boots directly into OS, can't access grub menu.

**Root Cause:**
GRUB_TIMEOUT=0 or hidden menu config.

**Solution:**

1. Edit:

```
sudo vi /etc/default/grub
```

Set:

```
GRUB_TIMEOUT=5
GRUB_HIDDEN_TIMEOUT=0
```

2. Update grub:

```
sudo update-grub
```

**Tip:**
Hold **Shift** (BIOS) or **Esc** (UEFI) during boot to access GRUB temporarily.

---

**Q84. Grub rescue prompt after reboot.**

**Scenario:**
After booting, you see grub rescue> instead of the OS.

**Root Cause:**
Missing GRUB bootloader, corrupted MBR, or deleted boot partition.

**Solution:**

1. Boot from live CD.
2. Mount and chroot:

```
mount /dev/sdaX /mnt
grub-install --boot-directory=/mnt/boot /dev/sda
update-grub
```

**Tip:**
Take disk snapshots before resizing partitions or reinstalling bootloaders.

---

**Q85. System stuck at "Loading initial ramdisk".**

**Scenario:**
Boot hangs on the initramfs stage.

**Root Cause:**
Corrupted initrd.img or incorrect init binary location.

**Solution:**

1. Boot into recovery shell or live CD.
2. Mount root and rebuild initramfs:

```
update-initramfs -u -k all
```

**Tip:**
Avoid removing old kernel/initramfs versions unless space is low.

---

**Q86. Boot takes over 5 minutes.**

---

**Scenario:**
Boot process is unusually slow, hangs at certain services.

**Root Cause:**
Timed out services (e.g., network, NFS) or disk UUID mismatches.

**Solution:**

1. Analyze boot time:

```
systemd-analyze blame
```

2. Disable slow services:

```
systemctl disable service-name
```

**Tip:**
Check journalctl -b for detailed boot-time logs.

---

**Q87. Kernel upgrade causes server to not boot.**

**Scenario:**
After upgrading kernel, boot fails — blank screen or crash.

**Root Cause:**
New kernel incompatible with modules or driver missing (e.g., for disk/NIC).

**Solution:**

1. Reboot into older kernel from GRUB.
2. Set default kernel:

```
sudo grub-set-default <menuentry>
```

update-grub

**Tip:**
Never remove older kernel versions until new one is fully tested.

---

**Q88. Black screen after boot — no login prompt.**

**Scenario:**
System boots, but shows only a black screen. No shell, no GUI.

**Root Cause:**
Wrong target (graphical instead of multi-user), X11/GDM issue, or no TTY active.

**Solution:**

1. Switch TTY: Press Ctrl + Alt + F2
2. Set correct target:

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

```
sudo reboot
```

**Tip:**
Use multi-user target for servers; graphical is only for desktop systems.

---

**Q89. init or systemd process missing or corrupted.**

**Scenario:**
Boot fails with error like No init found.

**Root Cause:**
/sbin/init or /lib/systemd/systemd is missing or broken.

**Solution:**

1. Boot from live ISO.
2. Mount root and reinstall systemd:

```
sudo chroot /mnt
```

```
apt install --reinstall systemd
```

**Tip:**
Avoid deleting or modifying /sbin/init manually.

---

**Q90. Reboot command doesn't work — system hangs.**

**Scenario:**
reboot or shutdown -r now hangs and doesn't reboot the machine.

**Root Cause:**
A service or process is refusing to stop or a mounted NFS is blocking shutdown.

**Solution:**

1. Use force:

```
reboot -f
```

2. Or via magic sysrq:

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

**Tip:**
Enable SysRq if disabled:

```
echo 1 > /proc/sys/kernel/sysrqz
```

---

**Q91. Boot fails with "UUID not found" error.**

**Scenario:**
mount fails during boot due to missing UUID.

**Root Cause:**
Drive layout changed but /etc/fstab still uses old UUID.

**Solution:**

1. Get current UUIDs:

```
blkid
```

2. Update /etc/fstab with the correct UUID.

**Tip:**
Use LABEL= or /dev/sdX instead of UUIDs for removable disks.

---

**Q92. Disk added but not detected during boot.**

**Scenario:**
You attached a new disk, but it's not mounted at startup.

**Root Cause:**
Disk not present in /etc/fstab, or system doesn't wait for it.

**Solution:**

1. Add to fstab with nofail:

```
UUID=xxxx /mnt/data ext4 defaults,nofail 0 2
```

2. Run:

```
sudo mount -a
```

**Tip:**
Use nofail to avoid boot blocking if disk isn’t present.

---

**Q93. /boot partition full — can't install kernel updates.**

**Scenario:**
apt upgrade or yum update fails due to no space in /boot.

**Root Cause:**
Too many old kernels.

**Solution:**

1. List installed kernels:

```
dpkg --list | grep linux-image
```

2. Remove old ones:

```
sudo apt remove linux-image-<version>
```

**Tip:**
Keep only **2 latest kernels** and remove others.

---

**Q94. Kernel modules fail to load.**

**Scenario:**
System says modprobe: FATAL: module not found.

**Root Cause:**

- Kernel module doesn't exist
- Wrong kernel version

**Solution:**

1. Check kernel version:

```
uname -r
```

2. Rebuild modules:

```
sudo depmod -a
```

**Tip:**
Use lsmod to list loaded modules; modprobe to add.

---

**Q95. Kernel ring buffer (dmesg) shows I/O or memory errors.**

---

**Scenario:**
You see disk or memory errors repeatedly in dmesg.

**Root Cause:**
Hardware-level faults — failing disk or RAM.

**Solution:**

1. For disk:

```
smartctl -a /dev/sdX
```

2. For memory:
    - Reboot and run **memtest86+**

**Tip:**
Replace failing hardware immediately; don't just suppress warnings.

---

**Q96. SELinux blocks services from starting after reboot.**

**Scenario:**
After reboot, web or DB service fails, logs show permission denied.

**Root Cause:**
SELinux denied access due to context mismatch.

**Solution:**

1. Temporarily disable:

```
setenforce 0
```

2. Fix context:

```
restorecon -Rv /var/www
```

**Tip:**
Use audit2allow to convert denial logs into policies if needed.

---

**Q97. Kernel messages flooding logs (dmesg, /var/log/messages).**

**Scenario:**
Logs get filled rapidly with repetitive kernel messages.

**Root Cause:**
Driver or device generating excessive warnings (e.g., USB, NIC)

**Solution:**

1. Temporarily suppress:

```
dmesg -n 1
```

2. Permanently fix by updating kernel or blacklisting module.

**Tip:**
Don’t ignore this; investigate the root hardware issue.

---

**Q98. New kernel doesn’t recognize a NIC or RAID device.**

**Scenario:**
NIC or disk not detected after kernel upgrade.

**Root Cause:**
Missing drivers or removed support in newer kernel.

**Solution:**

1. Boot into older kernel.
2. Check lspci and lsmod
3. Install proper kernel module or switch to known working version.

**Tip:**
Always test new kernels on staging systems first.

---

**Q99. Kernel build fails — "missing headers".**

**Scenario:**
Trying to build a driver or module fails: missing kernel headers.

**Root Cause:**
Kernel headers not installed.

**Solution:**
Install matching headers:

```
sudo apt install linux-headers-$(uname -r)
```

**Tip:**
Always install build-essential package when compiling kernel modules.

---

**Q100. Kernel upgrade silently skipped during update.**

**Scenario:**
You run apt upgrade, but kernel version stays the same.

**Root Cause:**
You need to run dist-upgrade or install explicitly.

**Solution:**

```
sudo apt update
```

```
sudo apt install linux-image-generic
```

**Tip:**
apt upgrade doesn't install new dependencies like kernels; use full-upgrade.

---

**Q101. apt update fails with "Hash Sum mismatch".**

**Scenario:**
When running apt update, it fails with:

```
E: Failed to fetch ... Hash Sum mismatch
```

**Root Cause:**
Corrupted or out-of-sync APT cache or mirror issues.

**Solution:**

1. Clean local cache:

```
sudo rm -rf /var/lib/apt/lists/*
```

```
sudo apt update
```

**Tip:**
Switch mirrors if issue persists: use official or country-based mirrors.

---

**Q102. yum install fails with "Cannot find a valid baseurl".**

**Scenario:**
YUM can't connect to any repo and fails with:

Cannot find a valid baseurl for repo

**Root Cause:**

- DNS resolution issue
- Broken repo URL
- Internet not reachable

**Solution:**

1. Check internet:

```
ping google.com
```

2. Check repo file:

```
cat /etc/yum.repos.d/*.repo
```

Fix or change baseurl.

**Tip:**
Use CentOS Vault or EPEL for older RHEL versions.

---

**Q103. Package install fails with "Package is already installed".**

**Scenario:**
Installing a package gives:

```
dpkg: error processing package -- already installed
```

**Root Cause:**
Broken or half-configured install.

**Solution:**

1. Fix broken packages:

```
sudo dpkg --configure -a
```

```
sudo apt install -f
```

**Tip:**
This happens often after CTRL+C during install — always let apt finish.

---

**Q104. Can't install .deb file due to dependency errors.**

**Scenario:**
Using dpkg -i package.deb fails with dependency errors.

**Root Cause:**
dpkg doesn't auto-resolve dependencies.

**Solution:**

1. Run:

```
sudo apt install -f
```

**Tip:**
Prefer apt install ./package.deb — it resolves dependencies automatically.

---

**Q105. RPM install fails with "Failed dependencies".**

**Scenario:**
Installing .rpm package fails due to missing libraries or dependencies.

**Root Cause:**
YUM/DNF not used, or dependencies not in repo.

**Solution:**

1. Try:

```
sudo yum localinstall package.rpm
```

2. Or download all deps using:

```
sudo dnf install ./package.rpm
```

**Tip:**
Use yum provides to find which package offers the missing dependency.

---

**Q106. apt install always installs old package version.**

**Scenario:**
APT pulls an outdated version, not the latest.

**Root Cause:**

- Repo cache not updated
- New version not available in default repo

**Solution:**

1. Update:

```
sudo apt update
```

2. Use versioned install:

```
sudo apt install pkgname=1.2.3-1
```

**Tip:**
Use apt policy pkgname to list available versions.

---

**Q107. Trying to uninstall a package removes multiple dependencies.**

---

**Scenario:**
Uninstalling a package triggers removal of many important ones.

**Root Cause:**
It's a dependency or part of a meta-package.

**Solution:**

1. Use apt remove carefully.
2. Simulate first:

```
sudo apt remove --simulate pkgname
```

**Tip:**
Never apt autoremove blindly — verify the list first.

---

**Q108. yum or dnf gives GPG key warning.**

**Scenario:**
Package install fails due to:

Public key for rpm not installed

**Root Cause:**
Repo's GPG key not imported.

**Solution:**

1. Import the key:

```
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

**Tip:**
Use --nogpgcheck only for testing — it's unsafe in production.

---

**Q109. Installing one package breaks another.**

**Scenario:**
Installing a package downgrades or breaks existing working software.

**Root Cause:**
Repo conflict or manual .deb/.rpm file overriding system-managed versions.

**Solution:**

1. Use pinning (APT):

```
echo -e "Package: pkgname\nPin: version 1.2.3\nPin-Priority: 1001" | sudo tee /etc/apt/preferences.d/pkgname
```

**Tip:**
For RHEL, use version-lock plugin:

```
yum versionlock add pkgname
```

---

**Q110. dpkg package is broken and blocks all operations.**

**Scenario:**
dpkg is stuck or corrupted. All apt/dpkg commands fail.

**Root Cause:**
Unfinished install or broken post-install script.

**Solution:**

1. Force fix:

sudo dpkg --configure -a

sudo apt install -f

**Tip:**
If package still stuck, remove .list and .postinst from /var/lib/dpkg/info/.

---

**Q111. apt update is very slow.**

**Scenario:**
APT update takes minutes on a fast internet connection.

**Root Cause:**
Slow mirror or IPv6 issues.

**Solution:**

1. Switch mirror:

sudo sed -i 's|archive.ubuntu.com|mirror.xyz.com|g' /etc/apt/sources.list

2. Disable IPv6 (if needed):

echo 'Acquire::ForceIPv4 "true";' | sudo tee /etc/apt/apt.conf.d/99force-ipv4

**Tip:**
Use mirrors.ubuntu.com/mirrors.txt to find the fastest local mirror.

---

**Q112. A package is installed but command not found.**

**Scenario:**
apt says package is installed, but running the command gives “command not found”.

**Root Cause:**
Package doesn’t install binary in PATH, or needs login shell to expose it.

**Solution:**

1. Find binary:

dpkg -L pkgname | grep bin

2. Use full path or symlink:

```
sudo ln -s /opt/tool/bin/exe /usr/local/bin/exe
```

**Tip:**
Use which or type -a to check path visibility.

---

**Q113. Need to downgrade a package.**

**Scenario:**
New version of package breaks functionality; want to rollback.

**Root Cause:**
APT/YUM does not downgrade by default.

**Solution:**

1. Use versioned install:

```
sudo apt install pkg=old_version
```

or for RPM:

```
sudo dnf downgrade pkgname
```

**Tip:**
Pin the version to prevent auto-upgrade in future.

---

**Q114. Package upgrade fails due to held packages.**

**Scenario:**
APT blocks upgrade due to held packages.

**Root Cause:**
Manual hold set earlier.

**Solution:**

1. List held packages:

```
apt-mark showhold
```

2. Unhold:

```
sudo apt-mark unhold pkgname
```

**Tip:**
Use hold for critical packages like kernels or libraries you don’t want changed.

---

**Q115. After installing software, man pages not available.**

**Scenario:**
Command works, but man tool gives "No manual entry".

**Root Cause:**
Man pages are in a separate package.

**Solution:**
Install documentation package:

sudo apt install pkgname-doc

**Tip:**
Some packages use info instead of man. Try:

info tool

---

**Q116. Want to remove all unused packages and dependencies.**

**Scenario:**
System is cluttered with unused libs and dependencies.

**Root Cause:**
Autoremove not run after uninstalling packages.

**Solution:**

1. Run cleanup:

sudo apt autoremove

sudo apt clean

**Tip:**
In RHEL, use:

dnf autoremove

---

**Q117. Installed two versions of Python, package installs into wrong one.**

**Scenario:**
pip install puts package in wrong Python path.

**Root Cause:**
Default pip not pointing to desired version.

**Solution:**

1. Use version-specific pip:

```
python3.10 -m pip install flask
```

**Tip:**
Always use virtualenv for app-specific dependencies.

---

**Q118. RPM installation reports "conflicts with file from package X".**

**Scenario:**
Installing .rpm reports file path already owned by another package.

**Root Cause:**
Two RPMs try to own same file.

**Solution:**

1. Remove conflicting package:

```
sudo rpm -e --nodeps conflictingpkg
```

2. Or skip file overwrite via rpm2cpio.

**Tip:**
Resolve properly, don't force installs in production.

---

**Q119. Package says installed, but binary is missing after reboot.**

**Scenario:**
App installed, works, then disappears on reboot.

**Root Cause:**
Installed on ephemeral or non-persistent mount (e.g., /tmp, overlay).

**Solution:**

1. Reinstall in persistent directory:

```
sudo apt install --reinstall pkgname
```

**Tip:**
Always verify mount output before installing in dynamic file systems.

---

**Q120. Want to know which package provides a specific command or file**

**Scenario:**
Need to install a package that contains a specific binary.

**Solution:**

1. On Debian/Ubuntu:

```
apt-file search /usr/bin/command
```

2. On RHEL/CentOS:

```
yum provides /usr/bin/command
```

**Tip:**
Install apt-file with sudo apt install apt-file && apt-file update

**Q121. New user created but cannot login via SSH.**

**Scenario:**
User exists, but SSH login fails with Permission denied.

**Root Cause:**
User's home directory or .ssh/authorized_keys missing or wrong permissions.

**Solution:**

1. Check home directory permissions:

```
ls -ld /home/username
```

```
chmod 700 /home/username
```

2. Fix .ssh permissions:

```
chmod 700 ~/.ssh
```

```
chmod 600 ~/.ssh/authorized_keys
```

**Tip:**
SSH is very strict about directory and file permissions for security.

---

**Q122. sudo command says "user is not in the sudoers file".**

**Scenario:**
Trying to run sudo, but gets:

user is not in the sudoers file. This incident will be reported.

**Root Cause:**
User isn't added to the sudo group or explicitly allowed.

**Solution:**

1. Add to sudo group:

```
sudo usermod -aG sudo username
```

2. Or edit sudoers safely:

```
sudo visudo
```

**Tip:**
Never edit /etc/sudoers manually — always use visudo to avoid syntax errors.

---

**Q123. Password expiry forces user logout immediately after login.**

**Scenario:**
User logs in but immediately sees "Your password has expired" and logout.

**Root Cause:**
Password expired but no chance to reset.

**Solution:**

1. Reset password:

```
sudo passwd username
```

2. Check expiry:

```
sudo chage -l username
```

**Tip:**
Set password policies carefully via /etc/login.defs.

---

**Q124. After a user is deleted, their processes still run.**

**Scenario:**
Deleted a user account, but their processes are still active.

**Root Cause:**
Deleting a user doesn't auto-kill processes.

**Solution:**

1. Kill remaining processes:

```
pkill -u username
```

**Tip:**
Always pkill -u immediately after deleting a user to cleanly terminate all tasks.

---

**Q125. Cannot su to another user — "Authentication failure".**

**Scenario:**
su - username fails with authentication error even with correct password.

**Root Cause:**

- Account locked
- Shell not valid

**Solution:**

1. Unlock account:

```
sudo passwd -u username
```

2. Check shell:

```
sudo usermod -s /bin/bash username
```

**Tip:**
Always ensure /etc/passwd has a valid shell for login users.

---

**Q126. New user created but home directory missing.**

**Scenario:**
User created, but /home/username directory not present.

**Root Cause:**
User created without -m option or default configs changed.

**Solution:**

1. Create manually:

```
sudo mkdir /home/username
sudo chown username:username /home/username
```

2. Or recreate user properly:

```
sudo useradd -m username
```

**Tip:**
Always create home directory unless it's a system/service account.

---

**Q127. id shows wrong groups after adding a user to a new group.**

**Scenario:**
User added to a new group but id username doesn't show updated group.

**Root Cause:**
Group membership cached in session. Logout/login required.

**Solution:**

1. User must logout and log back in.
2. Or re-exec shell:

```
exec su - username
```

**Tip:**
Use newgrp groupname for instant session group switch.

---

**Q128. sudo asks for password every time.**

---

**Scenario:**
User is in sudoers, but sudo always asks for password.

**Root Cause:**
Default sudo behavior.

**Solution:**

1. Edit sudoers:

```
sudo visudo
```

2. Add:

```
username ALL=(ALL) NOPASSWD:ALL
```

**Tip:**
Use NOPASSWD only for trusted accounts or specific commands.

---

**Q129. Cannot delete a user: "user is currently logged in".**

**Scenario:**
userdel fails with "user is logged in".

**Root Cause:**
Active shell/session is preventing deletion.

**Solution:**

1. Kill user session:

```
pkill -u username
```

```
sudo userdel -r username
```

**Tip:**
Use who or w to see all active sessions before deletion.

---

**Q130. /etc/passwd manually edited, now users can't login.**

**Scenario:**
Manually edited /etc/passwd, and now login fails.

**Root Cause:**
Syntax error or corrupted fields.

**Solution:**

1. Validate:

```
pwck
```

2. Restore from backup:

```
cp /etc/passwd- /etc/passwd
```

**Tip:**
Always use vipw to safely edit /etc/passwd.

---

**Q131. LDAP user cannot use sudo.**

**Scenario:**
LDAP-authenticated user logs in but gets "not in sudoers" error.

**Root Cause:**
LDAP config missing sudo rules, or sudo schema not enabled.

**Solution:**

1. Update /etc/nsswitch.conf:

```
sudoers: files ldap
```

2. Or assign sudo rights manually in /etc/sudoers.

**Tip:**
Corporate LDAP setups should push sudo policies centrally.

---

**Q132. /etc/shadow file is missing.**

**Scenario:**
System login completely broken — no password authentication possible.

**Root Cause:**
/etc/shadow accidentally deleted or corrupted.

**Solution:**

1. Boot into recovery mode.
2. Recreate or copy backup:

```
cp /etc/shadow- /etc/shadow
```

```
chmod 000 /etc/shadow
```

**Tip:**
Protect /etc/shadow — it's critical; restrict access to root-only.

---

**Q133. User account is locked accidentally.**

**Scenario:**
User cannot login; gets "account locked" message.

**Root Cause:**
Account locked with passwd -l or failed login attempts.

**Solution:**

1. Unlock:

```
sudo passwd -u username
```

**Tip:**
Monitor /etc/shadow for accounts with ! mark indicating lock.

---

**Q134. Users can su to root without password.**

**Scenario:**
Anyone can su to root without a password prompt.

**Root Cause:**
Root password is blank or /etc/pam.d/su misconfigured.

**Solution:**

1. Set root password:

```
sudo passwd root
```

2. Verify PAM config.

**Tip:**
Never leave root password blank, even if SSH is disabled.

---

**Q135. User can login even after being deleted.**

**Scenario:**
Deleted a user, but they still can SSH/login.

**Root Cause:**
SSH key in ~/.ssh/authorized_keys allows access even without /etc/passwd entry.

**Solution:**

1. Remove home directory:

```
rm -rf /home/username
```

2. Remove cached sessions:

```
sudo loginctl terminate-user username
```

**Tip:**
Always delete /home/username after user deletion.

---

**Q136. Cannot change user's shell — "Shell not found".**

**Scenario:**
Trying to set user's shell but error:

```
chsh: /new/shell: No such file
```

**Root Cause:**
Shell must exist in /etc/shells.

**Solution:**

1. Check available shells:

```
cat /etc/shells
```

2. Add new shell path if required.

**Tip:**
Only allow approved shells for security compliance.

---

**Q137. User keeps getting disconnected after login.**

---

**Scenario:**
User logs in successfully but immediately disconnected.

**Root Cause:**

- .bashrc, .profile, or shell scripts contain exit/logout command.

**Solution:**

1. Check user's .bashrc, .bash_profile, .profile.
2. Look for exit, logout, or fatal errors.

**Tip:**
Always test user startup scripts before deploying widely.

---

**Q138. Need to expire user account after a specific date.**

**Scenario:**
You want a temporary user to auto-expire after 7 days.

**Solution:**
Set expiry:

```
sudo chage -E $(date -d "+7 days" +%Y-%m-%d) username
```

**Tip:**
Use chage -l username to review expiry settings anytime.

---

**Q139. Need to force user password change at next login.**

**Scenario:**
Force users to reset passwords when they first login.

**Solution:**
Force change:

```
sudo chage -d 0 username
```

**Tip:**
Good practice when creating new corporate users.

---

**Q140. How to allow a user to run only one command via sudo.**

**Scenario:**
User should be allowed to run only specific commands with sudo.

**Solution:**
Edit sudoers:

username ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart nginx

**Tip:**
Limit sudo carefully for security — never give unrestricted shells via sudo.

---

**Q141. Service is enabled but not starting at boot.**

---

**Scenario:**
systemctl enable servicename done, but after reboot service is inactive.

**Root Cause:**

- Wrong target
- Dependency service not ready
- Service failed early but not retried

**Solution:**

1. Check target:

bash

CopyEdit

systemctl list-dependencies multi-user.target

2. Add proper After= dependencies in the unit file if needed.
3. Analyze boot logs:

journalctl -b -u servicename

**Tip:**
Always match service timing with network.target or remote-fs.target if network mount involved.

---

**Q142. systemctl start fails with "Unit not found".**

---

**Scenario:**
Trying to start service, error:

Unit myapp.service not found.

**Root Cause:**

- Service file missing in /etc/systemd/system/
- Typo in service name

**Solution:**

1. List all units:

systemctl list-units --all

2. Reload daemon:

sudo systemctl daemon-reexec

**Tip:**
After adding new .service files, always run systemctl daemon-reload.

---

**Q143. Edited systemd service file but changes don't apply.**

**Scenario:**
Modified .service file, but behavior didn’t change.

**Root Cause:**
Systemd caches units in memory; needs daemon reload.

**Solution:**

1. Reload:

sudo systemctl daemon-reload

2. Then restart service:

sudo systemctl restart myapp

**Tip:**
Use systemctl show servicename to verify active settings.

---

**Q144. systemctl restart command hangs indefinitely.**

**Scenario:**
Restarting a service hangs forever.

**Root Cause:**

- Service script not handling stop properly
- PID file stale

**Solution:**

1. Force stop:

sudo systemctl kill servicename

2. Check ExecStop and PIDFile configs in .service.

**Tip:**
Always set reasonable TimeoutSec=30 in critical services.

---

**Q145. Service keeps failing repeatedly (Restart loop).**

**Scenario:**
Service starts, crashes, auto-restarts in loop.

**Root Cause:**

- Restart settings too aggressive
- Actual binary error

**Solution:**

1. Set limits:

Restart=on-failure

RestartSec=5

2. Investigate real crash cause via logs.

**Tip:**
Don’t use Restart=always without a good reason — it can hide real failures.

---

**Q146. Status of service shows "inactive (dead)".**

---

**Scenario:**
Systemd shows service as inactive after starting it manually.

**Root Cause:**

- Service exited immediately after starting
- Missing Type=forking or wrong ExecStart

**Solution:**

1. Set proper Type:

Type=simple

Or if it forks:

Type=forking

**Tip:**
Forking daemons must have PID tracking for systemd to manage them properly.

---

**Q147. journalctl logs are missing after reboot.**

**Scenario:**
Old service logs disappear after system restart.

**Root Cause:**

- Logs stored in memory (volatile)
- Persistent logging disabled

**Solution:**

1. Configure persistent logs:

sudo mkdir -p /var/log/journal

sudo systemctl restart systemd-journald

**Tip:**
Check /etc/systemd/journald.conf — set Storage=persistent.

---

**Q148. Systemd service fails — "Exec format error".**

**Scenario:**
Trying to start service, error:

Exec format error

**Root Cause:**
Wrong shebang (#!) or executable permissions.

**Solution:**

1. Check script file:

```
head -1 /path/to/script.sh
```

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

2. Fix first line:

```
#!/bin/bash
```

**Tip:**
Always verify scripts run manually before using in systemd.

---

**Q149. Daemon is running but systemctl status shows failed.**

**Scenario:**
Service actually running, but systemctl status shows "failed".

**Root Cause:**
Service started outside systemd (manual nohup, etc.) or PID tracking broken.

**Solution:**

- Always start/stop services using systemd, not manually.
- Fix PIDFile parameter in unit file if used.

**Tip:**
Use Type=notify if app supports sd_notify() for better tracking.

---

**Q150. Service refuses to bind to port — permission denied.**

**Scenario:**
Service fails to bind to port below 1024 (e.g., port 80).

**Root Cause:**
Only root can bind ports < 1024.

**Solution:**

1. Run service as root, or
2. Use authbind / setcap:

```
sudo setcap 'cap_net_bind_service=+ep' /path/to/binary
```

**Tip:**
Safer to bind to high ports and use Nginx reverse proxy if possible.

---

**Q151. System boot time increased after enabling a service.**

**Scenario:**
Boot takes longer after you enabled a new custom service.

**Root Cause:**
Service startup delays waiting for external resources.

**Solution:**

1. Add TimeoutStartSec=30 inside service unit.
2. Or move service to After=network-online.target or remote-fs.target.

**Tip:**
Use systemd-analyze blame to identify slow services.

---

**Q152. Created custom service but it doesn't appear in systemctl list.**

**Scenario:**
New .service file created but systemctl list-units doesn’t show it.

**Root Cause:**
New unit file placed wrong, or daemon-reload missed.

**Solution:**

1. Move unit file to:

/etc/systemd/system/

2. Then:

sudo systemctl daemon-reload

**Tip:**
Units should always go into /etc/systemd/system/ (not /usr/lib/systemd/ manually).

---

**Q153. Cannot stop systemd service: "Job for xyz.service canceled".**

**Scenario:**
systemctl stop immediately returns job canceled.

**Root Cause:**
Service stop script fails or exits early.

**Solution:**

- Check ExecStop command inside unit file.
- Use kill if badly hung.

**Tip:**
Graceful shutdown scripts are critical in writing services.

---

**Q154. After disabling service, it still auto-starts.**

**Scenario:**
Disabled a service, but it restarts automatically after reboot.

**Root Cause:**

- Service is socket-activated
- Timer unit triggers it

**Solution:**

1. Check for related sockets:

systemctl list-sockets

2. Check for timers:

systemctl list-timers

Disable them if needed.

**Tip:**
Timer-based activations need explicit disabling too.

---

**Q155. A systemd service runs fine manually but fails under systemctl.**

**Scenario:**
Manual command works but systemctl fails.

**Root Cause:**
Environment variables missing inside systemd.

**Solution:**

1. Set environment in service:

```
Environment="ENV_VAR=value"
```

Or load file:

```
EnvironmentFile=/etc/myapp/env
```

**Tip:**
Always test with systemctl show-environment.

---

**Q156. System time wrong even after NTP enabled.**

**Scenario:**
timedatectl status shows NTP active, but wrong time.

**Root Cause:**

- NTP server unreachable
- Firewall blocking NTP ports

**Solution:**

1. Verify NTP servers:

```
chronyc sources
```

2. Or force sync:

```
sudo timedatectl set-ntp true
sudo ntpdate pool.ntp.org
```

**Tip:**
Always whitelist UDP 123 for NTP in firewall.

---

**Q157. Systemd journal files consuming too much disk space.**

**Scenario:**
/var/log/journal/ grows very large.

**Root Cause:**
Unlimited log retention policy.

**Solution:**

1. Edit /etc/systemd/journald.conf:

```
SystemMaxUse=500M
```

2. Restart journald:

```
systemctl restart systemd-journald
```

**Tip:**
Configure log rotation to control disk usage.

---

**Q158. Cannot mask a service — "Masking is not permitted".**

**Scenario:**
Trying to mask a critical system service but getting permission error.

**Root Cause:**

- Insufficient privilege
- Trying to mask protected units

**Solution:**

1. Run as root:

```
sudo systemctl mask servicename
```

**Tip:**
Mask critical services (like systemd-udevd) only when absolutely necessary.

---

**Q159. systemctl commands are extremely slow.**

**Scenario:**
Running systemctl commands takes 10–20 seconds to return.

**Root Cause:**

- DNS resolution delays
- Systemd waits on unresponsive hosts

**Solution:**

1. Disable slow DNS resolution in /etc/hosts.
2. Set UseDNS=no in sshd config if SSH delay involved.

**Tip:**
Check /etc/hosts and hostnamectl status for consistency.

---

**Q160. Need to limit resource usage of a service.**

**Scenario:**
Want to cap memory, CPU for a specific service.

**Solution:**
Add into unit file:

MemoryMax=500M

CPUQuota=50%

Reload daemon and restart service.

**Tip:**
Use systemd-cgtop to monitor live cgroup resource usage.

**Q161. Cronjob is scheduled but doesn't run.**

---

**Scenario:**
Added an entry to crontab, but script never executes.

**Root Cause:**

- Wrong cron syntax

- Path issues
- Script lacks execution permission

**Solution:**

1. Check cron syntax:
2. Add full paths inside scripts.
3. Make script executable:

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

**Tip:**
Cron uses a minimal environment — always specify absolute paths.

---

**Q162. Cronjob runs but output is missing.**

**Scenario:**
Cron executes but no output saved.

**Root Cause:**

- No output redirection
- Silent failure

**Solution:**
Redirect output:

```
* * * * * /path/to/script.sh > /tmp/script.log 2>&1
```

**Tip:**
Always log both stdout and stderr for cronjobs.

---

**Q163. Cron says "Permission denied" when running a script.**

**Scenario:**
Cron tries to run a script but permission error occurs.

**Root Cause:**
Script missing executable bit or ownership mismatch.

**Solution:**

1. Make executable:

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

2. Ensure user has access.

**Tip:**
Cron runs as the user who owns the crontab — permissions must match.

---

**Q164. crontab -e gives "no crontab for user".**

**Scenario:**
First time setting up cron, you see:

no crontab for user

**Root Cause:**
Normal behavior — no crontab yet.

**Solution:**

- Just create new entries after opening crontab -e.
- Save and exit.

**Tip:**
Use sudo crontab -e carefully — root's crontab is different from user's.

---

**Q165. Cron job fails silently when using sudo in the script.**

**Scenario:**
Script with sudo commands fails without visible errors.

**Root Cause:**
Cron environment may not allow sudo without tty.

**Solution:**

- Edit /etc/sudoers:

Defaults:username !requiretty

Or better: avoid sudo inside cron scripts — pre-authorize permissions.

**Tip:**
Safer to schedule privileged jobs using root’s crontab directly.

---

**Q166. Email alerts from cron not received.**

**Scenario:**
Cron job has errors but no email notification is received.

**Root Cause:**
Mail system (sendmail or postfix) not configured.

**Solution:**

1. Install MTA:

```
sudo apt install postfix
```

2. Ensure cron sends emails.

**Tip:**
Or redirect cron output manually to a mail command.

---

**Q167. A cronjob runs every minute instead of once daily.**

**Scenario:**
Job runs much more often than intended.

**Root Cause:**
Wrong crontab timing format.

**Solution:**
Correct format for daily at midnight:

```
0 0 * * * /path/to/script.sh
```

**Tip:**
Use crontab.guru website to validate cron expressions.

---

**Q168. Cron runs script but reports "command not found".**

**Scenario:**
Script runs from shell but fails under cron.

**Root Cause:**
Environment variables like PATH missing in cron.

**Solution:**
Specify full path inside script:

/usr/bin/python3 /path/to/script.py

Or define PATH at top of script.

**Tip:**
Print environment inside cron script (env > /tmp/env.txt) to debug.

---

**Q169. Cron job requires internet but fails after reboot.**

**Scenario:**
Script needs network but fails if cron runs early after boot.

**Root Cause:**
Network not yet ready at job execution.

**Solution:**
Delay job by adding sleep:

@reboot sleep 60 && /path/to/script.sh

**Tip:**
Or use systemd After=network-online.target units for critical network jobs.

---

**Q170. Need to run cron only on weekdays.**

**Scenario:**
Script must run Monday–Friday only.

**Solution:**
Use:

0 9 * * 1-5 /path/to/script.sh

(Runs 9 AM Monday to Friday)

**Tip:**
Numbers 1-5 represent Monday (1) through Friday (5) in cron.

---

**Q171. A cronjob modifies environment for future shells — doesn’t work.**

**Scenario:**
Script sets variables but next shell doesn't see them.

**Root Cause:**
Cron jobs don't persist to user's shell environment.

**Solution:**
Use scripts to export into a file:

/path/to/script.sh > ~/.bash_profile

Reload manually later.

**Tip:**
Better to have static environment variables for crontab — avoid dynamic ones.

---

**Q172. Want to schedule a cronjob every 15 minutes.**

**Scenario:**
Need periodic execution every 15 minutes.

**Solution:**
Add to crontab:

*/15 * * * * /path/to/script.sh

**Tip:**
*/15 covers minute 0, 15, 30, and 45 each hour.

---

**Q173. Cron logs missing even after enabling logging.**

**Scenario:**
No cron activity visible in logs.

**Root Cause:**
Syslog not logging cron or cron daemon misconfigured.

sudo tail -f /var/log/syslog

1. Enable cron service:

sudo systemctl enable cron

sudo systemctl restart cron

**Tip:**
On RHEL, cron logs usually go to /var/log/cron.

---

**Q174. Script runs manually but fails via cron — permissions error.**

**Scenario:**
Script works fine from terminal but fails when cron runs it.

**Root Cause:**
Different user context under cron.

**Solution:**

- Fix file permissions.
- Ensure script and files are readable/writable by cron job user.

**Tip:**
Always test cron jobs as the intended user (su - username).

---

**Q175. Need to run a heavy cronjob during off-peak hours.**

**Scenario:**
Backup job needs to run only at night.

**Solution:**
Schedule at 2 AM:

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

**Tip:**
Backup/maintenance jobs should always be scheduled during lowest usage windows.

---

**Q176. Cronjob scheduled but server reboot wipes all jobs.**

**Scenario:**
Cron entries disappear after reboot.

**Root Cause:**
Crontab file was not saved properly or running on ephemeral filesystem.

**Solution:**

- Always use crontab -e and save properly.
- Avoid editing /var/spool/cron/username manually.

**Tip:**
Check persistence settings if using cloud ephemeral disks.

---

**Q177. Multiple crontabs for same user conflicting.**

**Scenario:**
Two separate crontabs overwrite each other's jobs.

**Root Cause:**
Each crontab -e command edits the same user's crontab.

**Solution:**

- Combine all cron jobs into one crontab.
- Maintain a cron management script if needed.

**Tip:**
Use dedicated system cron files /etc/cron.d/ for complex cases.

---

**Q178. Need to disable a cronjob temporarily without deleting it.**

**Scenario:**
Want to stop cronjob but not remove it permanently.

**Solution:**

- Comment out the cron line by prefixing with #.
- Example:

```
# 0 3 * * * /path/to/script.sh
```

**Tip:**
Leave reason for disabling in comments for future reference.

---

**Q179. Need a cron job that runs only once after reboot.**

**Scenario:**
Script should run only once after boot.

**Solution:**
Use:

```
@reboot /path/to/script.sh && crontab -l | grep -v 'script.sh' | crontab -
```

**Tip:**
Or better: use systemd oneshot service for cleaner one-time startup jobs.

---

**Q180. Need separate cron schedules for different users.**

**Scenario:**
Different users need different cron schedules.

**Solution:**
Each user has their own crontab:

```
sudo crontab -u username -e
```

**Tip:**
Use /etc/cron.d/ if you want central control over multiple users' cronjobs.

---

**Q181. rsync backup fails with "Permission denied".**

**Scenario:**
Trying to backup files using rsync, but gets:

```
rsync: send_files failed: Permission denied
```

**Root Cause:**
Destination directory not writable by user.

**Solution:**

1. Ensure destination permissions:

```
chmod -R u+w /backup/dir
```

2. If using remote, add sudo properly.

**Tip:**
When doing remote backups, prefer SSH key-based access.

---

**Q182. tar extraction fails — "Cannot open: Permission denied".**

**Scenario:**
Extracting a tar file throws permission errors.

**Root Cause:**
Extracting as non-root user into a root-owned directory.

**Solution:**
Use:

```
sudo tar -xvf archive.tar -C /target/dir
```

**Tip:**
Always ensure correct target folder permissions before extracting.

---

**Q183. Need to create a compressed .tar.gz backup.**

**Scenario:**
Want to create a compressed backup file.

**Solution:**
Run:

```
tar -czvf backup.tar.gz /path/to/data
```

- c → create
- z → gzip compress
- v → verbose
- f → file output

**Tip:**
Add date in filenames: backup-$(date +%F).tar.gz

---

**Q184. Restore only one file from a .tar.gz archive.**

**Scenario:**
Want to extract a single file from a huge backup.

**Solution:**
List files:

```
tar -tzf backup.tar.gz
```

Extract one:

```
tar -xzf backup.tar.gz path/to/neededfile
```

**Tip:**
Always preview (-t) before extraction to avoid overwriting.

---

**Q185. rsync shows 100% CPU usage during large sync.**

**Scenario:**
System slows down when doing large rsync backups.

**Root Cause:**
Rsync checksum calculations are CPU-heavy.

**Solution:**
Use --whole-file for local transfers:

```
rsync --whole-file src/ dest/
```

**Tip:**
For very large syncs, prefer rsync -z for compression only on slow networks.

---

**Q186. Rsync copies everything again instead of just differences.**

**Scenario:**
Each rsync re-copies all files, not just changed ones.

**Root Cause:**

- Timestamps mismatch
- File permissions changed

**Solution:**
Use proper flags:

```
rsync -avz src/ dest/
```

- -a preserves timestamps and permissions.

**Tip:**
Always avoid modifying timestamps (touch) between syncs unless necessary.

---

**Q187. Rsync deletes important files at destination.**

**Scenario:**
Accidentally added --delete flag, files lost.

**Root Cause:**
--delete removes files at destination not present at source.

**Solution:**
Avoid unless absolutely necessary.

For safe delete, use:

```
rsync --delete --backup --backup-dir=/path/to/recovery src/ dest/
```

**Tip:**
Always simulate first with --dry-run before risky rsync operations.

---

**Q188. Tar backup is corrupt — can't extract.**

**Scenario:**
Extraction fails midway, saying unexpected EOF.

**Root Cause:**
Partial transfer, disk full, or network timeout.

**Solution:**

1. Check tar file size.
2. Recreate backup:

```
tar -czvf newbackup.tar.gz /data
```

**Tip:**
Always verify backup integrity immediately after creation.

---

**Q189. Need incremental backup using tar.**

**Scenario:**
Full backup every day is too large; want only changes.

**Solution:**
Use:

```
tar --create --file=backup.tar --listed-incremental=snapshot.file /path
```

- snapshot.file tracks changes.

**Tip:**
Combine full + incremental backups smartly for faster restores.

---

**Q190. Rsync between two remote servers without local download.**

**Scenario:**
Copy files between two remote servers without pulling locally.

**Solution:**
Direct transfer:

```
rsync -avz -e ssh user1@host1:/path user2@host2:/path
```

**Tip:**
SSH key authentication speeds up remote rsync significantly.

---

**Q191. How to schedule daily automatic backups with rsync.**

**Scenario:**
Want to automate daily backups.

**Solution:**
Add crontab:

```
0 2 * * * rsync -avz /data/ /backup/
```

Runs daily at 2 AM.

**Tip:**
Redirect logs to file for troubleshooting cron-based backups.

---

**Q192. Need to verify integrity of tar backup.**

**Scenario:**
Worried if backup tar is not corrupt.

**Solution:**
Test tar without extracting:

```
tar -tvf backup.tar.gz > /dev/null
```

If no error = healthy archive.

**Tip:**
Use checksums (md5sum, sha256sum) for extra verification.

---

**Q193. Rsync skips hidden files.**

**Scenario:**
After rsync, hidden dotfiles missing at destination.

**Root Cause:**
Source path missing trailing slash or pattern filter wrong.

**Solution:**
Use:

```
rsync -avz /src/ /dest/
```

(trailing / matters!)

**Tip:**
Always specify source as /src/ not /src.

---

**Q194. Tar extraction overwrites existing files.**

**Scenario:**
Untarring a backup overwrites current files.

**Solution:**
Extract to a new directory:

```
mkdir newdir
```

```
tar -xvf backup.tar.gz -C newdir/
```

**Tip:**
Always extract cautiously in production environments.

---

**Q195. Rsync consumes too much network bandwidth.**

**Scenario:**
Rsync hogs all bandwidth, affecting other services.

**Solution:**
Throttle:

```
rsync --bwlimit=5000 -avz src/ dest/
```

(Limits to ~5MB/sec)

**Tip:**
Use --bwlimit during working hours; unthrottled during maintenance.

---

**Q196. Need to backup an entire Linux system.**

**Scenario:**
Want full Linux filesystem backup.

**Solution:**
Exclude runtime dirs:

```
rsync -aAXv --exclude={"/proc","/tmp","/dev","/sys","/run"} / /mnt/backup
```

**Tip:**
-AAX preserves ACLs, xattrs, hard links — crucial for system recovery.

---

**Q197. Restore a Linux server from rsync backup.**

**Scenario:**
Server failed, need to restore from backup.

**Solution:**

- Boot rescue mode.
- Partition and mount disks.
- Rsync backup onto fresh system:

```
rsync -aAXv /mnt/backup/ /mnt/newdisk/
```

- Reinstall bootloader:

```
grub-install /dev/sda
```

```
update-grub
```

**Tip:**
Double-check /etc/fstab UUIDs after restore.

---

**Q198. Rsync backup corrupts file permissions.**

**Scenario:**
Restored files have wrong permissions.

**Root Cause:**
Rsync without -a flag loses permissions.

**Solution:**
Use:

```
rsync -a src/ dest/
```

**Tip:**
-a == archive mode: preserves permissions, symlinks, timestamps.

---

**Q199. Rsync transfer interrupted — need to resume.**

---

**Scenario:**
Backup got interrupted midway due to network drop.

**Solution:**
Rerun rsync with the same command:

rsync -avzP src/ dest/

- -P enables partial resume.

**Tip:**
Rsync is intelligent — it resumes only missing parts!

---

**Q200. Need to encrypt a tar backup for secure storage.**

**Scenario:**
Want to backup data and encrypt it.

**Solution:**
Create backup:

tar -czf backup.tar.gz /data

Encrypt:

gpg -c backup.tar.gz

(Will prompt for passphrase)

**Tip:**
Use gpg --batch --passphrase-file for automated secure backups.

---