Q.No	Question	Source
1	What happens during the Linux system boot process from power on to login prompt?	DEV.to [deadlock]
2	What steps occur when the `ls` command is issued in the terminal?	DEV.to [deadlock]
3	Explain what inodes are in Linux and their significance in the filesystem.	DEV.to [deadlock]; ZeroToMastery
4	What is the /proc filesystem and what information does it expose in Linux?	DEV.to [deadlock]
5	How would you troubleshoot a “filesystem full” error when `df` shows free space?	DEV.to [deadlock]
6	Name several performance monitoring tools you have used on Linux and describe their purpose.	DEV.to [deadlock]
7	What are the different types of filesystems supported on Linux and which have you used?	DEV.to [deadlock]
8	Explain kernel space vs user space and why this distinction is important.	DEV.to [deadlock]
9	What are processes and threads? How are they different?	DEV.to [deadlock]; ZeroToMastery
10	How does Linux manage process and thread scheduling?	ZeroToMastery
11	Explain Linux kernel memory management in brief.	DEV.to [deadlock]
12	What is the difference between stack and heap in process memory?	DEV.to [deadlock]
13	Describe race conditions and how Linux manages concurrency.	DEV.to [deadlock]
14	What does a high load average indicate? How do you investigate load average issues?	DEV.to [deadlock]
15	How would you troubleshoot high I/O issues on a Linux host?	DEV.to [deadlock]
16	What happens when the system runs out of inodes?	DEV.to [deadlock]
17	What is a zombie process? How do you find them and clean them up?	DEV.to [deadlock]
18	What is a runaway process and how would you identify it?	DEV.to [deadlock]
19	How do you change file permissions and ownership in Linux? Mention the commands.	ZeroToMastery
20	What is the purpose of the `chown` and `chmod` commands?	ZeroToMastery
21	What are sticky bits, setuid, and setgid?	ZeroToMastery
22	How do you manage and view running processes? Mention commands involved.	ZeroToMastery
23	How do you find and kill a process by name or port number in Linux?	ZeroToMastery
24	What is SSH? How do you secure SSH access for production servers?	LinkedIn: Praveen Singampalli

25	How do you set limits for a user or a process in Linux?	ZeroToMastery
26	What is `nice` and `renice`? How are they used?	ZeroToMastery
27	How do you delete or clean up empty files securely?	ZeroToMastery
28	Explain the differences and use-cases for hard links and symbolic links.	ZeroToMastery
29	How do you schedule recurring tasks on Linux?	ZeroToMastery
30	How would you monitor disk usage and find large files or directories quickly?	DEV.to [deadlock]; ZeroToMastery
31	What backup strategies and tools have you implemented on Linux?	DEV.to [deadlock]
32	How do you configure static and dynamic IP addresses on Linux interfaces?	ZeroToMastery
33	How would you set up and manage firewalls on a Linux server?	ZeroToMastery
34	What is SELinux/AppArmor and what role does it play in production infrastructure?	ZeroToMastery
35	How would you investigate and solve DNS resolution issues in Linux hosts?	DEV.to [deadlock]
36	What are the common troubleshooting steps when an application is unreachable over the network?	ZeroToMastery
37	How would you tune the Linux kernel parameters (sysctl)?	ZeroToMastery
38	What is shell scripting? Give an example of shell script automation in past work.	ZeroToMastery
39	How does package management differ between apt and yum?	ZeroToMastery
40	Which log files do you frequently check for troubleshooting? Where are logs stored?	DEV.to [deadlock]
41	What is systemd? How does it differ from init?	DEV.to [deadlock]
42	Describe the sequence of actions when a user logs into a system via SSH.	DEV.to [deadlock]
43	How do you investigate and resolve “out of memory” issues in Linux?	DEV.to [deadlock]
44	What is cgroups? How do you use cgroups to limit resource usage?	ZeroToMastery
45	Which commands or tools are useful for examining network usage and connections?	ZeroToMastery
46	What’s your strategy for securing a freshly deployed public Linux server?	ZeroToMastery
47	What user and group management commands do you know?	ZeroToMastery
48	How do you perform kernel upgrades safely?	ZeroToMastery
49	How do you automate deployments or configuration management on Linux?	LinkedIn: Praveen Singampalli

50	Explain the basic workflow of troubleshooting a failed service on Linux.	DEV.to [deadlock]

Answers to the above questions.

Q.No	Answer
1	During a Linux system boot: BIOS/UEFI initializes hardware, bootloader (GRUB/LILO) loads the kernel, which then initializes drivers. `init` or `systemd` starts essential services and user-space processes. Lastly, the login manager or shell prompts the user.
2	After `ls` is entered: The shell creates a process to run `/bin/ls`, which makes system calls to list directory contents, fetches metadata from inodes, and outputs to the terminal.
3	Inodes store filesystem metadata (owner, permissions, timestamps, disk block locations) for files/directories, but not names.
4	/proc is a virtual filesystem exposing kernel/process info (CPU, memory, process state) as files.
5	“Filesystem full” with space available: Check inode usage (`df -i`), disk quotas, files on hidden mount points, and open file descriptors holding deleted files (find via `lsof`).
6	Monitoring tools: `top`, `htop`, `vmstat`, `iostat`, `sar`, `dstat`, `free`, `nmon` are common, provide CPU, memory, IO, and process info.
7	Filesystems: ext4, xfs, btrfs, zfs, nfs, tmpfs, etc. ext4/xfs common for general use; nfs for network shares.
8	Kernel space: System and hardware code runs here, isolated from user space, where apps run. Separation prevents user bugs from crashing the system.
9	Process: Independent execution contexts with memory and resources. Thread: Lightweight, shares process memory/context. Threads within a process share most resources; processes are isolated.
10	Linux uses a scheduler (CFS in modern kernels) to allocate CPU time fairly among threads/processes based on priority and workload.
11	Linux manages memory with paging, virtual memory, cache, swap; using LRU and demand paging to optimize performance.
12	Stack: Stores function calls, local vars, grows/shrinks per call. Heap: Dynamically allocated at runtime with `malloc`, persists until freed.
13	Race conditions occur when processes/threads access shared resources concurrently. Use locks, semaphores, mutexes in Linux to avoid.
14	High load average indicates many runnable processes; check CPU, IO bottlenecks, run `top`, `uptime`, investigate blocking tasks.
15	Troubleshoot I/O: Use `iostat`, `iotop`, `vmstat` to check disk throughput, latency, IO wait states. Identify heavy writers/readers.

16	No inodes: Can’t create new files, even with disk space. Check `df -i`, find/delete small files/directories.
17	Zombie process: Defunct, finished execution but not reaped by parent. Find with `ps aux
18	Runaway process: Consumes excess CPU/memory, often an infinite loop or bug. Identify with `top`/`ps`, kill or debug as needed.
19	Permissions: `chmod` (change mode), `chown` (change owner), `chgrp` (group). E.g., `chmod 755 file.txt`, `chown user:group file.txt`.
20	`chown`: Change file ownership (user/group). `chmod`: Change file/directory permissions (read/write/execute).
21	Sticky bit: Restrict file deletion to owner/root. setuid/setgid: Run file with file owner/group privileges (`chmod u+s/g+s file`).
22	View processes: `ps`, `top`, `htop`, `pgrep`, `pstree`, `jobs` (in shell).
23	Find/kill process by name: `pkill processname`; by port: `fuser -n tcp PORTNUM` or `lsof -i :PORTNUM`, then `kill PID`.
24	SSH (Secure Shell) provides secure remote login. Secure by disabling root login, using key-based auth, changing default ports, restricting users, enabling fail2ban.
25	Set limits: User/per-process resource limits via `/etc/security/limits.conf` (`ulimit` for session), e.g. `nofile`, `nproc`, `memlock`.
26	`nice`: Set process priority at start. `renice`: Change priority of running processes. Lower value = higher priority.
27	Delete empty files: `find . -type f -empty -delete`. Secure delete: `shred`, `wipe`, or `srm`.
28	Hard link: Another name for same inode; symbolic (soft) link: pointer to filename. Hard links can’t span filesystems or link directories.
29	Recurring tasks: Use `cron` (`crontab -e`), or `at` for one-time jobs.
30	Find large files: `du -h --max-depth=1`, `find . -size +100M -print`, `ncdu`, or `df -h` for space usage.
31	Backup strategies: `rsync`, `tar`, `cpio`, `dd`, automated cron, cloud backup solutions (e.g. AWS S3, rclone). Incremental and full backups.
32	IP configuration: Edit `/etc/network/interfaces` (Debian) or `ifcfg-*` (RedHat), use `ip addr`, `ifconfig`, `nmcli` or `netplan` for newer systems.
33	Firewalls: Use `iptables`, `firewalld`, `ufw` (Ubuntu-friendly). Define/allow/deny by ports/IPs.
34	SELinux/AppArmor: Offer mandatory access control, restricting program abilities for defense-in-depth in production Linux systems.
35	DNS troubleshooting: Use `nslookup`, `dig`, `host` for lookups; check `/etc/resolv.conf`; verify network routes/firewalls.
36	Unreachable app: Check process status, logs, firewall, port availability (`netstat`/`ss`), DNS, and health checks.

37	Tune kernel: Edit `/etc/sysctl.conf` or use `sysctl -w key=value`; e.g., for networking, memory, file descriptors.
38	Shell scripts automate tasks. Example: update packages, rotate logs, schedule cron backups. Use Bash, sh, etc.
39	apt (Debian/Ubuntu): Uses deb packages; yum (RHEL/CentOS): Uses rpm packages, different dependency management, configuration.
40	Logs: Main: `/var/log/` (syslog, messages, auth.log, dmesg, journalctl). Check relevant logs for service/debug info.
41	systemd: Modern PID 1 init system, service manager; replaces older `init` with dependency management, parallel startup, logging.
42	SSH login: Client connects, authenticates via key/password, server spawns login shell, initializes user environment.
43	Out of memory: System may invoke OOM killer, killing large processes. Monitor with `vmstat`, `top`, check `/var/log/messages`, add swap, optimize memory use.
44	cgroups: Kernel feature to limit/resource groups of processes (CPU, memory, IO). Used by Docker, Kubernetes, systemd.
45	Network usage: `netstat`, `ss`, `iftop`, `nethogs`, `iptraf`, `tcpdump`, `wireshark`.
46	Securing Linux: Update, create non-root users, configure SSH, enable firewall, disable unused services, enforce password policies, audit logs.
47	User/group commands: `useradd`, `usermod`, `passwd`, `groupadd`, `groups`, `deluser`, `delgroup`.
48	Kernel upgrades: Use package manager (`apt`, `yum`), reboot, keep previous kernel(s) for fallback, test first in staging.
49	Automation/config mgmt: `Ansible`, `Chef`, `Puppet`, `SaltStack`, shell/Python scripts; CI/CD pipelines (Jenkins, GitLab CI), Docker, Kubernetes.
50	Service troubleshooting: Check status (`systemctl status`), logs (`journalctl`), config files, restart, diagnose dependencies/resources.

Sources:

- DEV.to SRE/DevOps Questions
- ZeroToMastery Linux Interview Prep
- LinkedIn: Praveen Singampalli