Unix Questions Asked In Interview

MASTERING UNIX INTERVIEW QUESTIONS: A COMPREHENSIVE GUIDE

Unix has stood the test of time as a powerful, multi-user operating system that forms the backbone of countless servers, development environments, and embedded systems. For professionals aiming to secure roles in system administration, software engineering, DevOps, or IT support, a solid grasp of Unix is non-negotiable. Interviewers often use targeted **Unix questions asked in interview** settings to assess a candidate's practical knowledge, problem-solving abilities, and familiarity with core concepts. This guide explores the most frequently encountered topics and provides the depth you need to answer with confidence.

Whether you are a fresh graduate preparing for your first tech interview or an experienced engineer brushing up on fundamentals, understanding the types of questions that come up and the

rationale behind them can set you apart. Let’s dive into the essential areas you must master.

Core File System and Navigation Commands

A foundational area of **Unix questions asked in interview** revolves around file system navigation and manipulation. Interviewers want to confirm you can move efficiently through directories, manage files, and leverage the command line for everyday tasks.

- **Explain the difference between absolute and relative paths.** An absolute path starts from the root directory (`/`) and specifies every directory down to the target file, while a relative path is based on the current working directory. Example: `/home/user/docs` vs. `./docs` or `../docs`.
- **What is the purpose of the `ls` command and its common options?** `ls` lists directory contents. Key options include `-l` (long format), `-a` (show hidden files starting with a dot), `-h` (human-readable sizes), and `-t` (sort by modification time).
- **How do you find files by name or content?** Use `find` for file names (`find / -name "*.log" 2>/dev/null`) and `grep` for content search (`grep -r`

`"error" /var/log/`).

- **Explain the cd command and special directories.** `cd ~` goes to the home directory, `cd -` toggles to the previous directory, and `cd ..` moves up one level.

Interviewers may also ask you to create a symbolic link, move files with wildcards, or use `du` and `df` to check disk usage. These questions test your daily fluency.

File Permissions and Ownership

Security is a pillar of Unix, so expect several **Unix questions asked in interview** about permissions. Understanding read, write, and execute bits in both symbolic and octal notation is essential.

- **What do the permissions `rwxr-xr--` mean?** Owner has read, write, execute (7); group has read and execute (5); others have read only (4).
- **How do you change permissions recursively?** Use `chmod -R 755 directory`. Be cautious with scripts – the setuid/setgid bits require special attention.
- **Explain the difference between chown and chgrp.** chown changes file owner and optionally group (`chown user:group file`), while `chgrp` only changes

group ownership.
- **What is the sticky bit? Give an example.** The sticky bit (`chmod +t /tmp`) prevents users from deleting files owned by others within a shared directory, even if they have write permission. Common on `/tmp`.

You might also be asked how to view permissions of a script to ensure it is executable (`ls -l script.sh`) or how to use `umask` to set default permissions for newly created files.

Process

Management

Unix's process handling is another high-frequency topic. Interviewers look for your ability to monitor, control, and debug processes.

- **How do you list running processes?** Use `ps aux` for a detailed snapshot, or `top` / `htop` for real-time monitoring. `ps -ef` is also common.
- **What is a zombie process? How do you handle it?** A zombie is a child process that has terminated but its exit status has not been read by the parent. If the parent does not call `wait()`, the process remains as a zombie. Killing the parent usually allows

init to adopt and clean them up.

- **Explain foreground and background jobs.** A foreground job occupies the terminal; press Ctrl+Z to suspend it, then use `bg` to send it to background. Use `jobs` to list background tasks and `fg` to bring one back.
- **How do you kill a process by name or PID?** Use `kill -9 PID` (SIGKILL) to force stop, or `kill -15` (SIGTERM) for a graceful shutdown. To kill by name, `pkill process_name` or `killall process_name`.

Be prepared to explain process states (running, sleeping, stopped, zombie) and how signals work, especially SIGINT, SIGHUP, and SIGKILL.

Shell Scripting Basics

Shell scripting is a staple in almost every **Unix questions asked in interview**. Even non-administrator roles expect you to automate tasks with simple scripts.

- **What is the difference between `#! /bin/bash` and `#! /bin/sh`?** The shebang determines the interpreter. `bash` offers more features (arrays, extended syntax), while `sh` is POSIX-compliant. On some systems `sh` is linked to dash or bash in POSIX mode.
- **How do you pass arguments to**

a script? Inside the script, `$1`, `$2` represent positional parameters; `$#` is the count, `$@` all arguments, `$0` the script name.

- **Explain variable assignment and quoting.** `name="John"` without spaces around =. Use double quotes to allow variable expansion (`"Hello $name"`), single quotes for literal strings (`'Hello $name'`). Backticks or `$()` for command substitution.
- **Write a simple script to check disk usage and send an alert if over 90%.** Typically use `df -h` combined with awk to parse percentages, then an `if` condition with `mail` or a log entry.

Expect questions on loops (`for`, `while`), conditionals (`if-then-elif-else`), and error handling (`set -e`, `trap`).

Input/Output Redirection and Pipes

Efficient use of the command line relies on redirection and piping. These concepts appear in many **Unix questions asked in interview**.

- **Explain stdin, stdout, stderr.** File descriptors 0 (stdin), 1 (stdout), 2 (stderr). Redirection: `command > file` (overwrites stdout), `command >> file` (append), `command 2>&1`

(redirect stderr to stdout), `command < file` (read stdin from file).

- **What is the purpose of the pipe (|)?** It connects stdout of one command to stdin of another: `ls -l | grep "\.txt" | wc -l` counts text files.
- **How can you suppress error messages?** Redirect stderr to `/dev/null`: `command 2>/dev/null`.
- **Give an example of using `tee`.** `ls -l | tee file.txt` writes output to both stdout and file. Useful for logging while viewing.

You might also be asked to combine `find` with `xargs` or to use `awk` and `sed` for text processing within a pipeline.

Networking Configuration and Tools

Unix systems are often network-facing, so knowledge of networking basics is part of many **Unix questions asked in interview**.

- **How do you check network interfaces and IP addresses?** Use `ifconfig` (older systems) or `ip addr`/`ip link` (modern). `hostname -I` shows IP addresses.
- **Explain the use of `ping`, `traceroute`, and `netstat`.** `ping` tests connectivity,

`traceroute` shows the path packets take, `netstat` (or `ss`) displays open ports and connections.

- **What does `ssh` do? How do you copy files securely?** Secure Shell for remote login; `scp` and `rsync` for file transfers. Example: `scp file.txt user@host:/path/`.
- **How do you check which ports are listening?** `netstat -tulpn` or `ss -tulpn`. The `-n` option shows numeric addresses.

You may also encounter questions about DNS resolution (`nslookup`, `dig`), firewall rules (`iptables` or `ufw`), and troubleshooting connectivity with `telnet` or `nc`.

System Administration and Troubleshooting

For more senior roles, interviewers pose real-world **Unix questions asked in interview** that require diagnostic thinking.

- **A user cannot log in. How do you debug?** Check if the account is locked (`passwd -S username`), home directory exists with correct permissions, disk is full (`df -h`), or the shell is valid

(`cat /etc/passwd`). Also check `/var/log/auth.log` or `/var/log/secure`.

- **The system is running very slowly. What steps do you take?** Use `top` to find CPU/memory hogs, `iostat` for disk I/O, `free -m` for memory, `vmstat` for overall system performance. Check for runaway