VINAY.T

LINUX SCENARIO BASED Q&A

Q1. "The user can list (ls) the file but lacks the permissions to edit it."

Scenario: The user can list the file but receives a 'Permission denied' error when attempting to open it with an editor like vim or nano.

Root Cause: "They have read (r) permission but not write (w) permission on the file. To make edits, write access is required."

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 assign write permissions through group access rather than granting write access to others (everyone) on corporate systems."

Q2. "Accidentally deleted a configuration file, such as /etc/ssh/sshd_config."

Scenario: "If a critical system configuration file is accidentally deleted, any services that depend on it may fail to start or function properly after a system reboot."

Root Cause: "When a file is deleted, only its directory entry (metadata link) is removed. If the file is still held open by a running process, it may still be possible to recover its contents."

Solution:

1.Check whether any process is still using the deleted file:
lsof | grep deleted
2.Restore the deleted file: cp /proc/PID/fd/FD /path/to/newfile
3.If no processes are currently using the file:

a. "Restore the file from a backup."
b. "If no backup is available, use file system recovery tools (such as extundelete)."

Tip: Always set up automatic daily backups of the /etc directory to prevent system outages caused by configuration file loss.

Q3. Error: "No such file or directory" when executing a binary.

Scenario: A binary file exists, but running it results in a "No such file or directory" error.

Root Cause: This usually indicates either missing dependent shared libraries or the binary being compiled for a different architecture.

Solution:

1. Check the binary’s architecture: file ./binaryfile
2. Check for missing dynamic dependencies: ldd ./binaryfile
3. Install any missing libraries if necessary.

Tip: Always compile binaries on the same OS and architecture as your production environment to avoid compatibility issues.

Q4. "Text file busy" error when replacing a binary.

Scenario: Attempting to overwrite a binary that is in use results in a "Text file busy" error.

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

Solution:
1. Identify the process using the binary: fuser binaryfile
2. Terminate the process: kill -9 PID
3. Safely replace the binary.

Tip: To avoid runtime issues, use safer deployment strategies, such as versioned binaries and symlinks.

Q5. User can't access their own home directory.

Scenario: The user encounters a "Permission denied" error when attempting to cd into their home directory.

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

Solution:
1. Correct the ownership: sudo chown username:username /home/username
2. Adjust the permissions: chmod 700 /home/username

Tip: For security reasons, corporate servers often set the home directory permissions to 700 or 750.

Q6. Mistakenly set incorrect permissions on /etc/passwd, users can't log in.

Scenario: Users are unable to log in after incorrect permissions were set on the /etc/passwd file.

Root Cause: Critical system files like /etc/passwd must have the correct permissions to function properly.

Solution:
1. Boot into recovery mode.
2. Correct the permissions: chmod 644 /etc/passwd

Tip: Always use ls -l carefully when changing permissions on system files, as mistakes can disrupt authentication.

Q7. Cannot run a newly created shell script —> "Permission denied."

Scenario: The script was created, but attempting to run it results in a "Permission denied" error, even though the file exists.

Root Cause: The script is missing executable (x) permission.

Solution:
1. Add executable permission: chmod +x script.sh
2. Execute the script: ./script.sh

Tip: Always verify both the shebang (#!/bin/bash) and the permissions when troubleshooting script execution issues.

Q8. A symbolic link shows "No such file" after reboot.

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

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

Solution:
1. Check the symlink target: ls -l /path/to/symlink
2. Correct the target if necessary: ln -sf /correct/target/path /path/to/symlink

Tip: When creating symlinks for critical files, prefer using absolute paths to avoid issues after reboots.

Q9. Tar backup created but error occurs while extracting.

Scenario: Running tar -xvf backup.tar results in errors like "Unexpected EOF" or "corrupt archive."

Root Cause:
- The tarball creation was interrupted.
- The file was partially copied.
- There was a mismatch in the tar command or compression method.

Solution:
1. Check if the tar file is readable: tar -tvf backup.tar
2. If the file has minor damage, attempt to ignore bad blocks: tar --ignore-failed-read -xvf backup.tar

Tip: Always use compression methods like gzip or bzip2 (tar -czvf) for better integrity and reliability.

Q10. User created directory, but team members can't write into it.

Scenario: A user creates a project folder, but other team members are unable to create files inside it.

Root Cause: Missing group write permissions.

Solution:
1. Set the correct group ownership: sudo chown :projectgroup /path/to/projectdir
2. Add group write permission: chmod g+w /path/to/projectdir

Tip: Use the setgid bit (chmod g+s) on shared directories to automatically assign group ownership to new files created within them.