Assignment

What command is used to stage all changes in the working directory for the next commit in Git?

The command used to stage all changes in the working directory for the next commit in Git is:

```
git add .
```

Explain the difference between 'git add' and 'git commit' commands in Git.

'git add' is used to stage changes in the working directory, preparing them for the next commit. It moves modifications from the working directory to the staging area.

'git commit' creates a new commit with the changes staged in the repository. It permanently stores the changes and adds a commit message to describe the modifications.

What is the purpose of using the '.gitignore' file in a Git repository?

The '.gitignore' file in a Git repository is used to specify files and directories that should be ignored by Git. It helps exclude irrelevant or sensitive files, such as build artifacts, temporary files, or configuration files, from being tracked and committed to the version control system.

How can you undo the most recent commit in Git, keeping the changes in the working directory?

To undo the most recent commit in Git, keeping the changes in the working directory, you can use the following command:

```
git reset --soft HEAD^
```

Explain the concept of 'stashing' in Git and how it can be useful while working with files.

'Stashing' in Git is a feature that allows you to temporarily save changes in your working directory that are not ready to be committed. This is particularly useful when you need

to switch branches or perform other operations without committing incomplete work. The 'git stash' command is used to save changes, and 'git stash apply' is used to reapply those changes later. Stashing helps maintain a clean working directory and facilitates smoother context switching between tasks.

What are the differences between soft reset, mixed reset, and hard reset in Git?

In Git:

1) Soft Reset (git reset --soft HEAD^):

Resets the HEAD to the previous commit.
Keeps changes staged (changes are not lost).
Changes remain in the working directory.

2) Mixed Reset (git reset --mixed HEAD^ or simply git reset HEAD^):

Resets the HEAD to the previous commit.
Unstages changes (changes are not lost).
Changes remain in the working directory.

3) Hard Reset (git reset --hard HEAD^):

Resets the HEAD to the previous commit.
Discards changes both in the staging area and working directory.
Changes are lost (irreversible).

How can you view the changes between two commits for a specific file in Git?

To view the changes between two commits for a specific file in Git, you can use the following command:

```
git diff <commit1>..<commit2> -- <file-path>
```

Replace <commit1> and <commit2> with the commit hashes or branch names, and <file-path> with the path to the specific file.

What is the command to discard changes in the working directory for a specific file in Git?

The command to discard changes in the working directory for a specific file in Git is:

```
git checkout -- <file-path>
```

Explain the concept of Git hooks and how they can be used to manage files in a repository.

Git hooks are scripts that can be triggered at specific points in the Git workflow, such as before or after a commit, push, or merge. They provide a way to automate tasks and enforce policies within a Git repository.

Git hooks can be used to manage files in a repository by performing actions like:

1) Pre-commit checks: Enforcing coding standards, running tests, or checking for syntax errors before allowing a commit.

2) Pre-receive and update hooks: Enforcing policies on the server-side, such as restricting certain file types or checking commit messages.

3) Post-merge or post-checkout hooks: Automatically updating dependencies, rebuilding projects, or performing other actions after a branch is merged or checked out.

By customizing Git hooks, developers can streamline workflows, maintain consistency, and ensure that certain conditions are met before or after specific Git operations.

How does Git handle file permissions and line endings across different platforms?

1) File Permissions:

Git tracks executable permissions but may not fully preserve complex permission settings on files.
Git uses a simplified permission system (executable or non-executable) that accommodates differences across platforms.

2) Line Endings:

Git can handle different line ending conventions (CRLF for Windows, LF for Unix) automatically.
The 'core.autocrlf' configuration can be used to automatically convert line endings based on the platform.
The '.gitattributes' file allows specifying line ending preferences for specific files or paths.

What is the purpose of the 'git mv' command in Git, and how does it affect file tracking?

The 'git mv' command in Git is used to move or rename files and directories. It is a convenient way to perform these operations while maintaining proper file tracking within the Git repository. Instead of using separate 'mv' and 'git rm'/'git add' commands, 'git mv' accomplishes the move or rename operation and simultaneously stages the changes, preserving the file's history.