AAIT Institute of Technology

Group members. ID No. GitHub account

Fitsum

Mussie workneh. ATE/9096/15. Mo-me-0

Mihretu Tesema. ATE//15

Peter Koru. ATE//15

Yonathan Tatek ATE//15

MiniGit: A Custom Version Control System

Overview

MiniGit is a simplified, local-only version control system developed in C++. It is designed to replicate the core functionalities of Git, including version tracking, branching, committing, merging, and viewing logs. The objective of this project is to deepen our understanding of how modern version control systems work under the hood using fundamental data structures and file operations.

Features

1. Initialization (init)

- Creates a hidden `.minigit/` directory.

- Initializes internal data structures for commits, branches, and object storage.

- Sets the default branch to `master`.

2. Adding Files (add <filename>)

- Checks if a file exists and stages it for the next commit.

- Reads the file content and hashes it using a custom hash function based on it's content.

- Stores the content in `.minigit/objects/<hash>` directory.

3. Committing (commit -m <message>)

- Captures a snapshot of all staged files.

- Records metadata such as timestamp, commit message, and parent commit.

- Stores commits in objects and updates the current HEAD accordingly.

- Adds each commit to a linear commit list for traversal.

4. Viewing Log (log)

- Displays the commit history in reverse chronological order(from recent to older commits).

- Lists commit hash, message, and timestamp for each commit.

5. Branching (branch <branch-name>)

- the branch command has tow functionalities.

Wiht no argument taken, it displays the branchs highlighting the current branch.

With argument, it creates a new branch pointing to the current commit.

6. Checkout (checkout <branch-name> or checkout <commit-hash>)

- Switches the working directory to the specified branch or commit.

- Updates the HEAD reference accordingly.

7. Merging (merge <branch-name>)

- Finds the lowest common ancestor between the current and target branches.

- Performs a three-way merge based on file versions.

- Identifies and reports conflicts, if thete are any.

8. Diff viwer (diff <file1> <file2>)

- takes two files and compare them lene by line to identify differences

- displays the different parts indicated with there line number

Internal Data Structures

Component

Description

DSA Concepts

Commit

Stores hash, message, timestamp, parent, and file references

Hash map, Struct

CommitList

Maintains a linear list of commits

Linked list, Class

Blob

Represents versioned file contents

Hashing, File I/O

Branches

Maps branch names to commit hashes

Hash map

StagingArea

Temporarily holds files before committing

Hash table

How to Use

Compile

g++ minigit\_fileutils.cpp fileutils.cpp -o minigit

Run

./minigit <command> <argument/s>

Commands

- `init` - Initialize the repository.

- `add <filename>` - Stage a file.

- `commit -m "message"` - Create a new commit.

- `log` - Show commit history.

- `branch <name>` - Create a branch.

- `checkout <name/hash>` - Move to a branch or commit.

- `merge <branch>` - Merge a branch.

-'diff <file1> <file2>' - Identify the differences between two files

Limitations

- No remote repository support.

- Basic conflict handling in merge (only message output).

- Linear commit history maintained; no visualization.

Future Improvements

- Implement more file diffing tool for `diff` command.

- Add a GUI wrapper for visualization.

- Persist commit metadata to disk.

- Improve merge conflict resolution with markers.

Repository

fstream for file i/o handling as well as .

iostream for managing cmdl interface.

unordered\_map for key value pair interactions.