Time-Travelling File System

Amaan Khan

September, 2024

Following file specifies the instructions to setup/use the code and the assumptions made.

§1 Command Reference

- **INPUT:** The input is taken interactively with each command being executed in real-time. To terminate the program type the command `EXIT` or press `CTRL + D`.
- **CREATE FILE:** Use the command `CREATE <filename>`. This command creates a file with a root version (ID, 0), empty content, and an initial snapshot message. The root is marked as a snapshot. The code skips the command and displays an error message if a file with that name already exists, otherwise prints a message to confirm that the command has been executed successfully.
- **READ FILE:** Use the command `READ <filename>`. This command displays the content of the file's currently active version. The code skips the command and prints an error message if the file does not exist.
- **INSERT FILE:** Use the command `INSERT <filename> <content>`. This command appends the content to the file. If the file is a snapshot it creates a new version, otherwise modifies the version in place. The code skips the command and displays an error message if the file does not exist, otherwise prints a message to confirm that the command has been executed successfully.
- **UPDATE FILE:** Use the command `UPDATE <filename> <content>`. This command replaces the file's content. Same versioning logic as insert. The code also skips the command and displays an error message if the file does not exist, otherwise prints a message to confirm that the command has been executed successfully.
- **SNAPSHOT:** Use the command `SNAPSHOT <filename> <message>`. This command marks the active version as a snapshot, making its contents immutable. It stores the provided message and the current time. The code skips the command and displays an error message if the file

does not exist. If the file is already snapshotted, it displays a message signifying that as well, otherwise prints a message to confirm that the command has been executed successfully.

- **ROLLBACK:** Use the command `ROLLBACK <filename> [versionID]`. This command sets the active version pointer to the specified `versionID` and displays a message on successful execution. If no ID is provided it rolls back to the parent and displays a message regarding this as well. If the file doesnt exist it prints an error message. Otherise if the versionID is invalid/ doesnt follow input constraints prints an error message as well.
- **HISTORY** Use the command `HISTORY <filename>`. Lists all snapshotted versions of the file chronologically, which lie on the path from the active node to the root in the file tree, showing their ID, timestamp and message. If the file does not exist prints an error message.
- **RECENT_FILES** Use the command `RECENT_FILES [num]`. Lists the files in descending order of their last modification time restricted to the first `num` entries. If `num` doesn't follow the required input format the prints an error message. Otherwise if `num` > total_size of the heap then it lists all the entries and displays a message showing not enough elements present in the heap.
- **BIGGEST_TREES** Use the command `BIGGEST_TREES [num]`. Lists the files in descending order of their total version count restricted to the first `num` entries. If `num` doesn't follow the required input format the prints an error message. Otherwise if `num` > total_size of the heap then it lists all the entries and displays a message showing not enough elements present in the heap.
- **INVALID COMMAND:** If the command matches none of the defined commands in this section, then the code skips the command and prints an error message.

§2 Setup Instructions

- Enter the following commands in terminal

```
cd directory_of_project
chmod +x col106_run_assignment_1.sh
./col106_run_assignment_1
```

- Enter the commands in the input format specified in the previous section.

§3 Requirements

- `C++` compiler (e.g. `g++` or `clang++`)
- Install all the files locally and put them in the same directory before running the code.

§4 Important Notes

- Used `<ctime>` library for timestamps.
- On modifying a snapshotted version, the new version created is set as the active version.
- `INSERT, UPDATE, CREATE` are considered as modifications.
- `HISTORY` command prints all the timestamps.
- Assumed `filename` has no spaces.
- In command `BIGGEST_TREES` if there is a tie in `total_versions` then the file which was the last modified is printed first.
- In command `RECENT_FILES` if there is a tie in `timestamp` then the file which had a later modification statement is printed first.