TELER: A Streamlined Version Control System
Project Report

Hadley Luker Zachary J. Susag

May 15, 2018

1 Project Overview

During their education, most computer science students must learn to use a version control system (VCS). Although these systems see wide use in research and industry for software development and production, in educational contexts they typically service a narrower use case: saving, submitting, and tracking changes in programming assignments. This limited use case causes many students to become overwhelmed when confronted with the numerous and powerful commands available to modern VCSes such as Git and Mercurial.

For this reason, we decided to develop TELER: a streamlined version control system for the student use case. Our system exposes exactly five commands: repository initialization, pushing changes with a required commit-like summary to a remote, pulling changes from a remote, reverting changes to a specific version, and viewing version history. This interface condenses the common add-commit-push / pull command sequence at the expense of requiring all files to be pushed or pulled at once. However, this has positive effects: it encourages users to push frequently and document important changes as they are made.

Evaluation summary.

2 Design & Implementation

The design of our program is centered around four main components: the command line interface; a hash table, which contains commits, directories, and files; a tree, which organizes the hashed objects by their position in the file directory; and compression for items within the tree. This system draws heavily from the approached used by Git. ***Describe the workflow.***

2.1 Command-Line Interface

The command-line interface is a simple argument parser made using GNU C's `argp` interface. This interface automatically generates help and syntax information for display on the command line through the assignment of certain global variables and "option vectors", a structure which organizes and configures command line arguments. However, our implementation is slightly modified from the GNU standard of C argument morphology. The GNU practice is to precede all arguments on the command line with a number of hyphens: one for single-character arguments (e.g. `-v`) and two for long arguments (e.g. `--verbose`). We chose to let the first argument be hyphen-less. (For differentation's sake, we refer to this ”first argument” as the ”command”.) This reflects more closely the practices used by Git for their command line interface, which we suspect will be more familiar to students. This decision caused us to parse the first argument through an if–else if chain and instead use `argp`'s parser function to parse arguments given to the command itself.

The commands provided are `init`, `push`, `pull`, `history`, and `revert`. `init` creates a new TELER repository. `push` saves every new change made in the repository, while `pull` extracts the data from the latest saved change and updates the repository proper. `history` lists all saved changes to the repository in most-recent-first order. Finally, `revert` allows a user to specify a change to which they would like to revert back to for debugging purposes.

2.2 Hash Table

The hash table indexes all files within the repository, in addition to certain directory information and commits. Any data structure could feasibly be used for this, but since its primary purpose is for lookup, a hash table was chosen.

2.3 Tree

The tree organizes hash table entries by their file system structure. It enables `teler` to save or reconstruct the version of the repository that was saved in the latest commit.

2.4 Compression

As TELER saves every version of a file that was added in a change, it becomes important to utilize every means of space-reduction we have available to us. Therefore, all files and commits within a TELER repository are compressed. We utilize the `zlib` library to accomplish this. *** How it do and how we use ***

3 Evaluation