Tracker

Tanner Winkelman and Lucas Winkelman

We have found that Git is an extremely poorly designed piece of software. So we want to write a replacement for it for Git.

Tracker is to be a replacement for Git. It will track “states” of the code in a tree data structure. Now, Git uses a DAG. But Tracker will not allow merging, only the equivalent of rebasing. The root will be the empty state. States will be immutable and indestructible, like commits are in Git. We think the following commands will encompass the functionality of Git:

tracker show unsaved changes

* Like git diff

tracker list [number] (active)? states

* This will list [number] of the most recent states, in order of recency. If the active option is included, states that were used to generate other states will not be listed.

tracker save as [name]

* This will create a new state by [name] with the current state of all tracked files.

tracker move to state [name]

* This will ensure that all files that are tracked in state [name] are exactly the same in the state [name] and the file system. It will have a warning if unsaved changes are present.

tracker apply changes from [name] to [name2]

* This will attempt to apply all changes from state [name] to state [name2] onto the file system. This will not modify the state tree, only the file system. Conflicts are possible. If conflicts occur, the user will be prompted for what they want to do. Here are some options we might have:
* Put a bunch of trash in the files like Git does (<<<<<<<<<<<<<).
* Use the type of the code files to put comments in the files instead of trash.

So for cpp, java, and js files it would have //

For py and sh files it would have #

Prompt the user for what to do with unknown file types.

* Use one state over the other.

tracker upload (current state|state [name]) (to [servername])?

* This will upload a state to a server. May fail if a state with the same name already exists on the server.

tracker download state [name] (as [name2])?

* This will download a state to the local repository. This may fail if a state with the same name already exists on the local repository.

We might have the user make a file called the trackerignore (notice that it does not begin with a ‘.’, so users can see it). It would determine which files are not tracked.

This is a lot of code and we probably will only be able to implement some of this. Also, we may find that this set of commands has some flaws, so we might use a slightly different set of commands.