Lab 01: Version Control
Post Lab

Introduction

This lab will get you familiar with the basics of git. You will need to sign in to https://git-classes.mst.edu and clone the repository for this lab. Make sure to clone with the HTTPS URL (unless you've set up SSH keys).

Scenario

Your friend made a git repository for the filter program from another lab. They want to collaborate with you to add some features to it.

Problem 0: Learning git

Git is one of the most important things you can learn, so we're going to go over some good tutorials for it. Complete all steps of both tutorials and save screenshots of the completion windows for submission with the assignment.

3.1 learngitbranching

Complete all parts of the `https://learngitbranching.js.org` tutorial and save a screenshot of the main screen showing you've completed all levels of the tutorial.

Problem 1: `.gitignorance` is bliss

Remember how you shouldn't commit generated files to a repository? Well, your friend managed to add `a.out` anyway. Oops.

1. Use `git rm` to delete `a.out` from the repo.

2. Check `git status` to see what this change looks like.

3. Make a `.gitignore` that ignores `a.out`. Add it to the repository and commit.
   (hint: try using `echo` and the `>` operator to generate the `.gitignore` file instead of a text editor)

4. Add the `.gitignore` file to your repo, commit, and then push to git-classes.mst.edu.

Note that your commit in step 4 should commit two changes:

1. Remove `a.out`

2. Add `.gitignore`

Problem 2: Features grow on branches

Filter.cpp program can filter lines that contain whitespace (tabs and spaces) before a `#`. Instead of hardcoding the comment character as `#`, we want to have it take a user input that specifies a comment character.

1. Make a new branch to develop your feature on.

2. Modify `filter.cpp` to support your new feature, add it to the repository, and commit.

3. Write your code, add it to the repository, and commit.

4. Push your branch to the remote repository. (You can check git-classes.mst.edu to make sure it's up there.)
   The first time you `push` a new branch, you have to tell git to make a new branch on the remote repo. `git push` will tell you the right command to run.

 Now, you'd like to merge your feature into `master` too!

1. Okay so merge it already.

2. Don't forget to push after you've merged!

Epilogue: Submitting

Your repo on git-classes.mst.edu is your submission, so whatever is up there is what we will grade. Make sure you've pushed both your feature branch and master. You can double-check on the Gitlab website to make sure your submission looks the way you want it to.
   We expect to see the following files on the master branch:

- `README.md`

- `.gitignore`

- `filter.cpp`

Hints

- Use `git help <command>` to learn more about a command.
  For example, `git help commit` to learn about `git commit`

- Use GitLab's Network view (under the Repository tab) to see a pretty graph of your commit history. You'll also see your branch names and their corresponding commits there.