Assignment 4. Basic change management

Useful pointers

- Scott Chacon, Pro Git, especially §§1–3 (basics and branching), §5 (distributed version control), §7.1 (revision selection), §7.5 (searching), and §7.10 (debugging).
- Linus Torvalds, Jun Hamano *et al*., Git - local branching on the cheap
- Jacob Gube, Top 10 Git Tutorials for Beginners

Laboratory A: Exploring a linear development history

This lab uses the development repository for the Time Zone Database (tzdb).

1. Use GitHub from a browser to compute the difference between the previous and current commit to this repository. Save the resulting web page as a file `prevcur.html`.
2. Use GitHub from a browser to compute the difference between tzdb releases 2020f and 2021a. Save the resulting web page as a file `2020f-2021a.html`.
3. Clone the tzdb development repository, in Git format.
4. Write a shell or Python script `justone` that displays the difference from the previous and current commit, assuming the repository is what an ordinary Git command would use. Use your command on the just-cloned repository, and put the output of your command into a file `justone.out`.
5. Write a shell or Python script `compare-releases` that displays the difference between two tzdb releases given as arguments to the command. For example, `compare-releases 2020f 2021a` should output the difference between tzdb release 2020f and tzdb release 2021a. Put the output of this particular invocation into a file `2020f-2021a.diff`.
6. Suppose we're interested in the number of commits from each time zone. Write a shell or Python script `tzcount` that postprocesses the output of `git log` and outputs a simple report of time zones and number of commits from that time zone. Each line of output should look something like "`-0500 1802`", meaning there were 1802 commits from the -0500 time zone. Sort the output numerically by its first (numeric timezone) column. Run the command `git log 2021a | ./tzcount` using the tzdb repository, and put its output into a file `tzdb-2021a.tzcount`.
7. Suppose the maintainer of tzdb is being sued for copyright infringement because one of the source files contains the following statement: "Even newspaper reports present contradictory information." Also suppose the plaintiff claims that this statement was improperly copied from the plaintiff's book. Use Git and other commands to find out how this statement was introduced to the tzdb files. Create a text file `who-contributed.txt` that describes what commands and/or scripts that you used, and what the result of your investigation was.

Laboratory B: Exploring nonlinear development histories

There is a copy of some version of the GNU Emacs git repository's master branch on SEASnet in the directory `~eggert/src/gnu/emacs`.

1. Find the mergepoint M at 7dd52bfd8e503316b4aa9c5767850d3985626b26 (2020-10-17) and the commit C at c00606171f88be0df2c19346fa53f401ea71c71f (2020-10-10) and describe the relationship between C and M, by drawing a graph containing all paths from C to M. Your diagram need not list every commit in all the paths, but you should label and list every commit with more than one parent, or with more than one child. For example, your graph should have a node labeled C and M because C has multiple childen and M has multiple parents, and the graph's legend should say that C is

c00606171f88be0df2c19346fa53f401ea71c71f and that M is 7dd52bfd8e503316b4aa9c5767850d3985626b26.

2. Clone the GNU Emacs git repository yourself from Savannah, and briefly describe the differences between your repository and the one in `~eggert/src/gnu/emacs`. (Hint: look at the output of `git branch`.)

Put your descriptions into a text file `emacs.txt`. Put your diagram into a PDF file `emacs-graph.pdf`.

And now for a more open-ended search. The Git 2.30.1 Release Notes (2021-02-08) say "Newline characters in the host and path part of git:// URL are now forbidden." Which code change or changes actually made this happen, and who authored the change? Do not worry about subsequent administrative changes such as merge commits; look for the original changes that actually fixed the problem in question. Give the SHA1 commit IDs for the relevant changes, and explain how you discovered them, all in a text file `git.txt`. Also generate patch files for the relevant changes, in `git format-patch` format.

Submit

Submit the following files at the top level of a gzip-compressed tarball `gitlabs.tar.gz`. Your tarball may contain other files if you think it necessary.

- `prevcur.html` web page
- `2020f-2021a.html web page`
- `justone` script
- `justone.out` output file
- `compare-releases` script
- `2020f-2021a.diff` output file
- `tzcount` script
- `tzdb-2021a.tzcount` output file
- `who-contributed.txt` infringement report
- `emacs.txt` descriptions
- `emacs-graph.pdf` descriptions
- `git.txt` descriptions
- The output files generated by `git format-patch`
