CS50 Version Control and Collaboration

Overview

Collaboration is an integral part of computer science; part of programming is constantly sharing and collaborating with peers. Sites like Facebook and Google are not written by one person. Rather, they require thousands of engineers, all working in teams to build specific pieces of the site. Furthermore, one team might rely on the code of many other teams. Effective collaboration is crucial.

Key Terms

- abstraction
- documentation
- comments
- Git
- commit

Abstraction

The saying "too many cooks spoil the broth" alludes to the idea that too many people working on the same thing is counter-productive. In cooking, like in programming, one can largely avoid this issue through **abstraction**, by building large projects out of a set of smaller, self-contained sub-projects and assigning these sub-projects to different people.

In a restaurant, for instance, one person might make the appetizer, another might make the main course or dessert, and still another might wait the table. In code, a project might rely on many individual programs, with programs' functions calling on still more library functions. The harmony that arises from the working together of many distinct parts is truly a beautiful thing!

Documentation and Comments

If abstraction requires breaking projects into smaller, independent pieces, documentation and comments allow engineers – and readers, more generally – to fit the pieces together. The man pages are one example of **documentation**: to use the library function `strlen()` from `string.h`, one shouldn't need to look at its actual code. Instead, the library's documentation acts as an executive summary, describing how and when to use a product, in this case `strlen()`.

There are cases in which looking at original code is useful or necessary. For these, **comments** in the code explaining what it does are incredibly important. These can make debugging one's own code or that of others much easier. Similarly, if we wanted to create a slightly different product based on someone else's code, clear and comprehensive comments would also be valuable. To promote consistency and clarity, best practices often instruct the use of a common style within a project or an organization.

Version Control and Git

There are many tools that enable coding collaboration, the most popular of which is a file tracking system called **Git**. The Git workflow is divided into three stages. First, we work on files in our working directory. Then, we pick what changes we want to store and add those to our staging area (also known as index). Finally, we **commit** those changes, which means that a "snapshot" of our project is stored in our repository (.git directory). Git also features branches – copies of a master project – that allow programmers to experiment with changes without actually affecting the original project.

Many version control systems save data as changes relative to the original files. Git works differently by saving "snapshots" of the entire project every time we commit. The diagram at right shows these "snapshots," or versions, over time. So version 2 represents our repository after our first commit. From this "snapshot," we can see we only made changes to files B and C. When files have not changed, Git links back to the previous file commit. That's denoted here with a lighter gray color, as is the case with file A in version 2.