Version control systems

Finch
The CAT

April 16, 2010

Outline

What is VCS?

- A Version Control System, or VCS, is a way to track changes to files
- Allows you to manage code or documents as you edit it
- Makes collaborating on a document/code significantly easier

Why should you care?

- Track changes to your files over time
  - Try out experimental ideas to code without risking and permanent harm
  - Use VCS on your homework to track your changes between drafts
  - Turn a VCS into a backup system!
- Locate the revision that a bug was introduced
- Manage changes to a single file across multiple users
- VCS can prevent you from really buggering a project
  - “Wait, you **didn’t** want that file deleted?”

Why should you care?

- Any sane developer/project uses VCS
  - sourceforge
  - code.google.com
  - github/gitorious/git.or.cz
- The CAT utilized VCS in many locations
  - agendas - RCS
  - CRACK - Git
  - intranet - Git

Outline

Before VCS

- Living dangerously!

```
vim foo.txt
```

- *three hours later*

```
rm *.txt
```

. . .

- Whoops.

Do it yourself VCS

- Users replicated VCS tools by doing everything by hand

```
vim foo.txt
cp foo.txt foo.txt.1 && vim foo.txt
cp foo.txt.1 foo.txt.2 && vim foo.txt && cp foo.txt foo.txt.1 && vim
foo.txt
...
```

- Obviously, this is not a very fun idea

The beginning of VCS

- Initial idea of VCS came from engineering
- Engineers would create blueprints, and save earlier revisions
- If a version was not liked, it was trivial to roll back
- Revision control was also applied in business, law
- Any place where you may need to backtrack on a document, VCS will show up

Concerns with VCS

- Any VCS has to deal with some basic concerns
  - Merging changes
  - Locking files
  - Atomic commits
- Each VCS handles these concerns differently

Outline

Source Code Control System (SCCS)

- Arguably the first VCS system available
- Developed by Bell Labs in 1972
  - For some comparison, C was written in 1972 at Bell Labs as well
- SCCS was the dominant VCS until the advent of RCS
- Generally considered obsolete, and only mentioned for historical purposes
- Except for the storage method - still used today

Revision Control System (RCS)

- Released in 1982
- Created as an evolution of SCCS
- Stored changes of files as a series of diffs
- Very popular, still used today

Pros and Cons of RCS

- Advantages
  - Dirt simple
  - Only need to know a few commands
  - Changesets stored as series of diffs, so easy to view
  - File locking and branching supported
- Disadvantages
  - Single files only
  - Cannot store entire projects
  - No security mechanisms - anyone can tamper with diffs
  - Branching sucks - everybody just locks the file
    - co -l agenda.dog
    - vim agenda.dog
    - ci -u agenda.dog

Outline

Modern VCS

- While early VCS systems were an improvement, they were still lacking
- No real support for multiple users
- No support for project wide version control
  - Make changes in foo.c and bar.c, and changes in one break the other
  - No way to look at both of the changes without black magic and/or shell scripts
- Technologies started coming out to deal with this

Concurrent Version Systems (CVS)

- Behold! Another evolution!
- CVS was the first client-server VCS
- Instead of just hacking on one file in a specific location, you check out a project
- Support for project wide VCS
- Support for multiple users simultaneously working on one project

Pros and Cons of CVS

- Advantages
  - . . .
  - I got nothing.
- Disadvantages
  - It was the best of times, it was the worst of times
  - No moving/renaming of directories
  - No versioning of symbolic links
  - No unicode
  - No atomic commits
    - It'll commit enough changes to break everything
  - CVS was used because it was the first, not because it was good.

Outline