CSE 303 Concepts and Tools for Software Development

Richard C. Davis
UW CSE – 11/15/2006

Lecture 18 – Version Control Systems

Administravia

- Choose partners by 11PM today
  - E-mail choice to Lincoln Ritter
  - E-mail Richard Davis if you can't choose
- Excellent Specification/Testing Tools
  - Doxygen - for Javadoc-like specifications
  - Unit testing systems
    - CUnit, CppUnit, JUnit, CsUnit
  - Links to all on CSE 303 Documentation Wiki
    - (At the very bottom)

Administravia

- Monday: Societal Implications
  - Digital Rights Management
  - Read paper and turn in *brief* summary
    - Only one or two sentences!
  - This reading is a bit longer
- HW6
  - Available this afternoon
  - Due one week from today
    - So you can have Thanksgiving off

Tip of the Day

- File changed on disk, Emacs doesn't know
  - Use M-x revert-buffer

Last Time

- Basic Software Engineering
  - Development Process
  - Specification
  - Unit Testing and Stubs

Today

- Version Control Systems
  - Subversion

From School to The Real World

- Programming in School
  - You usually work alone
  - Your programs survive a few weeks at most
  - Working in groups?
    - E-mail files or sit at same computer
- Programming in the Real World
  - Your team is large (possibly huge)
  - Programs stay around for many years
    - Often long after original developers are gone
  - How do you manage?

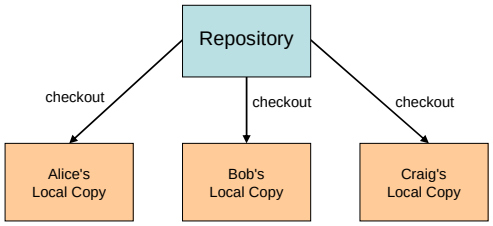
Code Management Problems

- Delete/re-arrange a large segment of code
  - Then decide you shouldn't have
- Many people working concurrently
  - Often working on the same files
  - Sometimes make incompatible changes
- Need to do maintenance on old versions
  - But can't stop working on new version

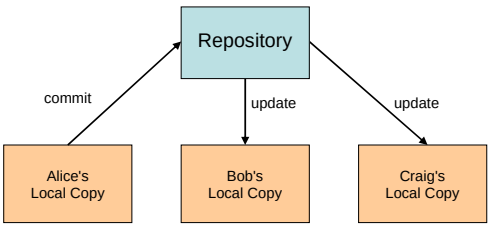
Version Control Systems

- Reliable backup of important information
  - Tools for "rolling back" to previous versions
- Manage concurrent changes
  - Possibly help resolve conflicts
- Many alternatives
  - Subversion - *we'll use this one*
  - CVS - *heavily used in this department*
  - RCS, SourceSafe, ClearCase
- *Note: These tools are not just for code!!*

Subversion: Basics



Subversion: Basics



The Repository

- There exists one Subversion *Repository*
  - Holds master copy of all files
    - Hopefully backed up
  - In new versions, only differences are kept
  - It's just a directory with special files
  - Can set up so it's accessible via Web/SSH
- Common tasks
  - Set up a repository: `svnadmin create`
  - Put a project into a repository: `svn import`
  - Check out project from a repository: `svn checkout`

What Goes Into a Repository?

- Keep ONLY files needed to build program
- Things to Keep
  - Source files, documentation, resources
  - `.c`, `.cpp`, `.h`, `Makefile`
- Things not to Keep
  - Files generated automatically
  - Your executable, `.o` files

Working With the Repository

- Examples
  - Set up a repository
    ```
    svnadmin create repos
    ```
  - Put a project into a repository
    ```
    svn import . file:///homes/rcd/repos/myproj
    svn import hw6 file:///homes/rcd/repos/hw6
    ```
  - Check out project from a repository
    ```
    svn checkout file:///homes/rcd/repos/hw6
    svn checkout file:///homes/rcd/repos/hw6 dirname
    ```


Developer Work Cycle

- Most common commands (`svn` command)
  - Update working copy
    - `update`
  - Make changes
    - `add, delete, copy, move`
  - Examine your changes:
    - `status, diff, revert`
  - Merge other's changes:
    - `update, resolved`
  - Commit your changes:
    - `commit`

Usage Examples

- Updating
  ```
  svn update
  ```
- Making changes
  ```
  svn add file.cpp
  svn move file.cpp newname.cpp
  svn delete newname.cpp
  ```
- Examine your changes:
  ```
  svn status
  svn diff file.cpp
  svn revert file.cpp
  ```
- Commit your changes:
  ```
  svn commit -m "Commit Message"
  ```


Conflicts

- Many people editing at the same time
  - Conflicts can occur
  - Subversion tries to merge automatically
  - If it can't, you must resolve the conflict

Dealing with Conflicts

- Discovering Conflicts
  - svn commit fails
  - Do `svn update` to get version in repository
  - "G" means the automatic merge succeeded
  - "C" means you have to resolve conflict
- Resolving Conflicts
  - look for `<<<<<` in file
  - run `svn resolved file`

Other Details

- Getting help with a command
  - `svn help command`
  - `svnadmin help command`

CVS is Subversion--

- CVS Problems
  - Has version numbers for individual files
    - What was version of foo.c when bar.c was 1.2?
  - Can't rename or delete files
    - Well, you *can*, but you lose the history
  - Doesn't handle binary files well
- Otherwise, very similar
  - `cvs` instead of `svn`, `cvs init` instead of `svnadmin create`
  - Different ways to specify repository
    - CVS uses `-d repository`
    - Subversion uses `file:///full/path/to/repository`
  - Missing cmds: `delete, copy, move, revert, resolved`
  - Modified cmds: `status`

Summary

- Managing code versions manually is hard
  - Adding dates to filenames?
  - Keeping multiple copies on disk?
- Version Control Systems
  - Key tool for managing versions
  - Store differences
  - Prevent problems that come from
    - Concurrent Access
    - Deleting Information
  - Can use for any type of information!

Reading

- Version Control with Subversion
  - Online at *http://svnbook.red-bean.com/en/1.2/*
  - Ch1: Introduction
    - Read section titled *A Quick Start*
  - Ch2: Basic Concepts
  - Ch3: Guided Tour
    - Read or skim for commands described in lecture

Next Time

- Build Scripting and Makefiles