Revision Control for Sysadmins

Presented at the
MUUG General Meeting on 2011-Sep-12

Adam Thompson, athompso@athompso.net

Who

- Designed for developers:
  - Version control, which led to
  - Software configuration management, which led to
  - Build management
- System administrators can also use:
  - Version control (almost anywhere)
  - SCM (complex situations)
  - Build management (large-scale deployment)7

What – Version Control

- Designed to keep track of changes to a file
- Most allow multiple people to simultaneously work on multiple versions of a file, then merge their changes later
- For sysadmins, allows you to track changes to config files
- Can be used to back up and synchronize files, too

What – Software Config. Mgmt.

- More complex than revision control
- Manages entire filesets at a single time
- Includes path information
- Includes metadata
- Includes build-environment information
- Usually client-server, designed for teams

What – Build Management

- Evolution of SCM
- Includes intelligence about building the project
  - Deploys the product, e.g. to a J2EE app server
  - Now includes "Application Lifecycle Management"
- Team-oriented
  - Covers workflow for sign-offs, approval, testing, etc.
- Usually complex
  - Supports "Continuous Integration"

Scope of presentation

- I'm only going to talk about Revision Control software
- Everyone can find a use for it
- Easy (easier, anyway) to understand, set up and use

Where

- ASCII configuration files
  - Unicode and National Character Set files are also generally supported
  - Binary files are typically unsuitable, but are generally supported without diff(1) capability
- /etc/*
- ~/.bashrc, et al.
- ~/Music
  - Yes, really: use CVS/SVN/git for synchronization!

When

- Manually, before and after each edit
- Manually, after testing
- Automated, from cron(8)
  - Daily snapshots
- Automated, from startup scripts
  - "Last-known-good" snapshots

Why

- What changed yesterday?
- What changed on 2010-Feb-06?
- Who changed it?
  - This is more complicated to set up
- Easy rollback of changes
- Easier to test new changes and apply them selectively

How - Origins

- SCCS – Source Code Control System
  - First shipped with AT&T System III PWD in 1972, can still sometimes be found in SVR4-derived OSes.
  - File format still used in other products today
- RCS – Revision Control System
  - Developed for BSD UNIX® in 1982, can be found almost everywhere. Part of the base system for many OSes.
  - Command invocation syntax is still used by many systems today

How – Evolution & Revolution

- CVS – Concurrent Version System
  - Client-server version of RCS
- SVN – Subversion
  - "CVS done right"
- git – completely *New* & *Different*!
  - Written by Linux Torvalds to manage the Linux kernel source

How - Others

- There are many other version control systems
- Many are commercial, many are cross-platform
- Some prominent systems:
  - Mercurial
  - Bazaar
  - BitKeeper
  - Visual SourceSafe
  - Rational ClearCase

RCS

- The only one I'm going to explain is RCS
  - Trivial to set up
  - Easy to use
  - Provides concepts necessary for understanding CVS, SVN, etc. (but not git)

RCS - Repository

- RCS tracks one file at a time. Period.
- RCS creates a "revision group", contained in a file named "`filename,v`".
- If a subdirectory called "`RCS`" exists, the "`,v`" files will be placed inside it.
- Each "`,v`" file stores the latest version of the file and all the reverse-deltas.
- Easy to recover some of the file even with a damaged repository

RCS - Initialization

- Strongly recommend using an `RCS/` directory
- The first `rcs(1)` command you run on a file will initialize the revision group.
- Manually do so with "`rcs -i`"
  - Avoids being prompted for the file description during checkin/checkout

RCS – Basic Concept

- Check In / Check Out: kind of like a Coat Check at a restaurant or concert
- Checked In:
  - You don't have it
  - You don't see it
  - You have to check it back out to use it
- Checked Out:
  - You're responsible for it

RCS - Locking

- A checked-out file can be locked or unlocked
  - Locked (for modification): read/write
  - Unlocked (for other use): read-only
- Leave files unlocked normally
  - Only lock them when you need to make changes
  - Prevents accidental, untracked changes

RCS – Basic commands

- "ci -u <filename>"
  - Checks IN a new version of the file, then immediately checks OUT an unlocked copy.
- "co -l <filename>"
  - Checks OUT a locked copy for modification

- That's all you need to know!

RCS - differences

OK, there's another command you need to know...

- `rcsdiff(1)`
  - Shows the differences between any two revisions of the file
  - With no options, `diff(1)`s the current working file against the last-checked-in-version. (i.e. "What have I changed so far this time?")

RCS - Tags

- If you put the magic string "`$Id$`" and "`$Log$`" into your file, RCS will automatically fill them in with useful information
- See `co(1)`, under KEYWORD SUBSTITUTION for more details

Typical uses

- /etc/httpd/httpd.conf
- /etc/openldap/slapd.conf
- /etc/postfix/main.cf
- /etc/*, really...
- /usr/local/bin/my-custom-script.pl

Limitations

- RCS only handles one file at a time
  - No notion of "projects"
  - Revision history is (deliberately) vulnerable to tampering
- Poor scalability
  - I/O scales as O(N) with file size, # of revisions, size of reverse-diffs
  - I/O scales as O(N^2) with # of active branches
- Single-user-writable model

Demo

- Maybe this will work...

Q&A

- I have to leave quickly tonight
- Several of our audience members are familiar with version control systems (mostly CVS and SVN, though)