Tools Talk

Project Management with RCS and CVS

Greg J. Badros
University of Washington
Computer Science & Engineering
12 November 1997

http://www.cs.washington.edu/homes/gjb/ToolsTalks

Outline

- What is PM and why care?
- Version control with RCS
  - Concepts
  - Commands
- Version control with CVS
  - Concepts
  - Commands
  - Secure Shell digression
- More information

What is Project Management?

- Grouping related files together
  - class project implementations
  - research papers
  - tools and data
  - configuration files
- Source code is valuable
- Tracking changes is useful

Make (another talk)

- Regenerateable targets can waste space
- Documenting build procedures is essential

Version control: RCS

- Revision control files
  - opaque
  - `,v` extension
  - `./RCS` directory if exists
- Working file
  - each person has a working file
  - only one person has a lock on theirs
- Version
  - single view of file, stored in RCS file
  - has a number, may be assigned a name

Putting a file under RCS control

- `mkdir RCS`
  `ci -u` <filename>
  or `rcs -i -kb` <filename>
- Remember `-u` option!
- `ci -l` <filename>
  keep lock on file
- What happens?
  - creates <filename>`,v`
    (uses `RCS` directory if exists)

Checking out working copies with RCS

- `co` <filename>
  - can use `,v` extension, but need not
    - e.g, `co RCS/*,v`
  - can use `-l` to lock,
    or `-u` to release lock
- Old versions?
  - `co -p1.4 | more`
  - `rcsdiff -r 1.1 -r 1.3 -u | more`
    - compare versions 1.1 and 1.3 using diff argument "`-u`"
  - `rlog` <filename>
- Keywords substituted `$Id$`, `$Log$`, `$Author$`, etc.

Adding a new revision

- Much like checking-in initial revision:
  `ci -u` <filename> # give up lock
  -or-
  `ci -l` <filename> # keep lock
- Prompts for change comments
  – high-level description
  – not diff! this can be computed

RCS Options

- Turn locking off
  `rcs -U` <filename>
  (`-L` to turn locking back on)
- Binary files?
  Turn off keyword substitution:
  `rcs -kb` <filename>
- Multiple users?
  Make `RCS` directory a symlink to a shared directory with write permissions for all project members

RCS Tip: Use Emacs!

- Version-Control (vc-mode) is built-in
- `C-x v v` = vc-next-action
  – Create RCS file
  – Check-in new versions
- Still need to:
  – Create `RCS` directory
  – use `rcs -kb` for binary files

RCS Limitations

- One person editing file at a time
- No extra grouping of files uses only directory structure
- `RCS` directory must be local
- Still useful for site-specific files; I use for `.aliases-site`, `.env`, etc.
- Useful primitives for CVS...

CVS - The Nouns

Concurrent Version Systems

- Repository
  – where `,v` are kept
  – may be remote, always sacred
- Module
  – named group of files under CVS control
  – multiple modules per repository
- Working (private) copy
  – a set of versions from a module
  – resides in a user's personal directory

CVS - The Verbs

Concurrent Version Systems

- checkout
  – create a working copy from a module
- update
  – integrate changes made to the repository into your working copy
- commit
  – publish changes in your working copy back to the repository (so others can update and use them)

Initializing a CVS Repository

once per repository

- `cvs -d ~/CVS-521 init`

once per module

- `cd current-fft-code`
- `cvs -d ~/CVS-521 import \ -m "CSE 521 FFT Project" \ fft gjb start`

module name — vendor name — release tag

- Ignores `*~`, `*.bak`, `core`, etc. use `-I !` to import everything
- Also does not keep symlinks!

```
find . -type l -printf "ln -s %l %p\n" \
    > recreate-symlinks-script
```


Checking out a module

- `cd ~/projects`
- `cvs -d ~gjb/CVS-521 checkout fft`

- Private working copy: `~/projects/fft/`
  - changes here not visible to other project members!
- checkout is recorded in repository, but otherwise has no other affect

Updating to see changes

- Execute when you want to integrate others' changes into your working copy

`cvs -d ~/CVS-521 update` **`-d`**

Look for new directories

U/**P**	=	Updated/Patched
M	=	Merged
A/**R**	=	Added/Removed
?	=	Extra file
C	=	Conflict!

- How do you know what's changed?

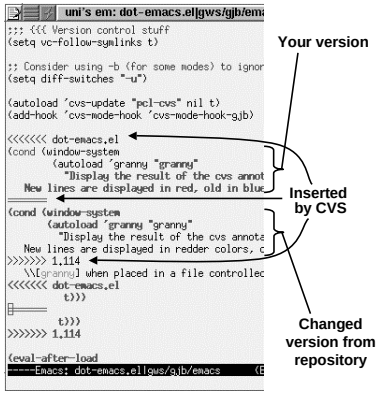
`cvs -n -q update` or `cvs diff`

Update example

```
% cvs -n -q update -d
M emacs.el
U proto.el
C dot-emacs.el
? generic-mode.el
```

```
% cvs update dot-emacs.el
RCS file: /homes/gws/gjb/.CVS/emacs/dot-emacs.el,v
retrieving revision 1.113
retrieving revision 1.114
Merging differences between 1.113 and 1.114 into
  dot-emacs.el
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in dot-emacs.el
C dot-emacs.el
```


Resolving a conflict



Undoing changes

- `rm dot-emacs.el`
  `cvs update dot-emacs.el`

- `cvs diff -u dot-emacs.el \ > dot-emacs.patch`
  `rm dot-emacs.el`
  `cvs update dot-emacs.el`
  apply patch by hand, or...
  edit dot-emacs.patch and then
  `patch <dot-emacs.patch`

Adding and Removing files

- `cvs add` <filenames>
- `cvs add` <directory>

- `rm fft-wo-templates.cc`
- `cvs remove fft-wo-templates.cc`

- Careful with globs!

```
set foo = *.o # (T)csh
foo=(*.o)     # {Z,Ba}sh
rm $foo
cvs remove $foo
```


Committing your changes

- Must be up to date on files you have changed
- `cvs commit` or `cvs commit` <filenames>
- Will be prompted for a log entry, as with RCS

CVS and Emacs

(pcl-cvs mode via M-x cvs-update)

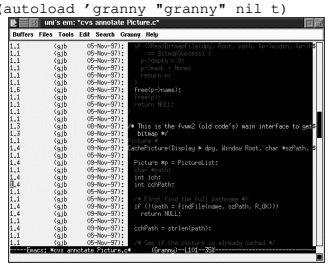
Using pcl-cvs mode

- add file
- remove file
- commit file
- diff file against repository
- Undo (discard) changes to local file
- Get (update) changes again
- (un)mark file
- (Un)Mark all files
- log for file

As always, C-h m to list mode help

CVS Options for Emacs

- First:
  `;;;` /uns/share/emacs/site-lisp in load-path
  `(autoload 'cvs-update "pcl-cvs" nil t)`
- Consider:
  `(setq cvs-update-optional-flags '("-d"))`
- Granny-mode
  `(autoload 'granny "granny" nil t)`



Other CVS Commands

- `cvs` **`rtag`** `turned_in fft`
- `cvs` **`tag`** `turned_in fft`
- `cvs` **`admin`** `-kb fft-binary`
- `cvs` **`diff`**
- `cvs` **`rdiff`**
- `cvs` **`log`**
- `cvs` **`release`** `[-d]`
- `cvs` **`status`** `-v`
- `cvs` **`history`**
- `cvs` **`export`**

Understanding CVS Commands

- `cvs [opt] command [cmd-opt]`

Global options relevant to multiple commands

-d <repository>
-n (no changes)
-e <editor>
-z <compression-level>

Options specific to the command

update -d
status -v
rdiff -r1.1 -r 1.5

- Environment variables
  - CVSROOT
  - CVSEDITOR
  - CVS_RSH

Remote repositories

- `cvs -z 9 -d \`
  `gjb@pahtoo:/homes/gws/gjb/CVS \`
  `checkout dots`
- Often faster than giving an NFS path on a linux box (perhaps omit `-z 9`, though)
- Works great from home, except for dynamic IP addresses
- Solution: Secure Shell (ssh)

What is the Secure Shell?

- Remote execution and file tools
  - ssh replaces rsh
  - scp replaces rcp
  - slogin replaces rlogin
- CVS uses some remote execution program to start the server on the computer hosting the repository
- `setenv CVS_RSH ssh`

Secure Shell vs. Rsh

	Ssh	Rsh
Encrypted passwords	✓	-
Token-based authentication	✓	-
Encrypted communication	✓	-
X-protocol forwarding	✓	-
AFS token forwarding	✓	-
Compression	✓	-

Setting up Ssh

- Need `sshd` running as root on target machine; already done!
- `ssh-keygen` generates private and public key pairs
- uses a "passphrase"
- copy generated public key to `~/.ssh/authorized_keys` (replaces `~/.rhosts` file)
- Now clients that have the private key can use the secure shell utilities on those hosts

Maintaining the private key

- Need to avoid re-entering the pass-phrase every time we ssh
- `ssh-agent` manages identities (name + private key) Permits all child processes to authenticate using the keys
- So, run your window manager with ssh.
  .xsession:
  .....
  `ssh-agent fvwm2 ....`

Adding identities

- ssh-agent only manages identities
- ssh-add confirms an identity, and gives it to the controlling ssh-agent
- Execute:
  `ssh-add </dev/null`
  from your window manager's startup routine

Customizing Ssh

- /etc/ssh_config
- ~/.ssh/rc
- List hosts from most specific to most general; first match wins!
- For home:

```
Host "monarch"
ForwardX11 yes
Compression no

Host "*"
ForwardX11 no
Compression yes
CompressionLevel 9
```


More information...

- RCS man pages
  - rcsintro
  - rcs, ci, co, rlog, rcsmerge, rcsdiff
- CVS Info pages
  - CVS 1.9
  - CVSClient - server protocol details
  - Pcl-CVS mode
- Secure Shell man pages
  - ssh, sshd, scp, slogin
  - ssh_keygen, ssh_agent, ssh-add

http://www.cs.washington.edu/homes/gjb/ToolsTalks