Introduction To
UNIX Software Development

Revision Control with RCS

Why should one use RCS?

- Keep history of changes to program so can revert to older versions and generate patches
- Arbitrate file usage between multiple users

Basic concepts

- RCS repository: either foo,v or RCS/foo,v
- Checking files in and out
- Locks

Basic commands

```
ci
  checks in and removes existing copy

ci -u
  checks in and releases lock (file becomes read-only)

ci -l
  checks in and retains lock (file is rw)

c0
  checks out with no lock (read-only)
```
co -u
check out with no lock (read-only)
co -l
check out and obtains exclusive lock
rlog
look at log
readiff
compare revisions and create patches. Example:
    readiff -r1.4 -o foo.c
rcs -u
break locks
rcs
do random things to rcs file

Lots of other options

- History trees
- Merging trees
- Symbolic version names
- For info, read man pages:
  - rcsintro, readiff, ci, co, rcs, rcsmerge

Logs and headers in files

- Put strings into files and RCS will expand them:

  $Header$
  $Id$
  $Date$
  $Log$

- For example, $Header$ gets expanded to something like this:

  $Header: /mit/sisp-iap/unixsoftdev/www/ RCS/rcs.html,v 1.2 1999/01/13 23:1

Reverting to old versions without dealing with branch messiness
• Reverting to an old version by checking out the old version and then checking it back in would normally force you to deal with branches which can get messy.
• An easier way is to do something like:
  
ci -l foo # save old version before clobbering
co -rl.5 foo # revert to version 1.5
cl -u foo # check in the reverted file and unlock

---

**Diffs and patches**

- Creating diffs:
  
  ```
  diff file1 file2
  
  or
  
  rcddiff -rl.5 -c foo # changes from 1.5 to present
  ```

- Types of diffs:
  - Old style: `diff`
  - Context: `diff -c`
  - Unified (my favorite): `diff -u`
  - Ed (I’ve never seen them used): `diff -e`

- Examples:
- Applying diffs/patches:

  ```
  patch < foo.patch -p -E
  ```
  
  - Modifies the file with the old filename
  - Patch can have more than one diff in it
  - Pre-patch file is renamed to `file.orig`
  - Rejected parts of patch are renamed to `file.rej` in the proper directory

---

**Alternatives**

- SCCS - like RCS but not as available at MIT
- CVS - layer on top of RCS which allows for dealing with really big source trees
  - RCS deals with files
  - CVS deals with directory trees
  - For more information, `add gnu; man cvs` on Athena