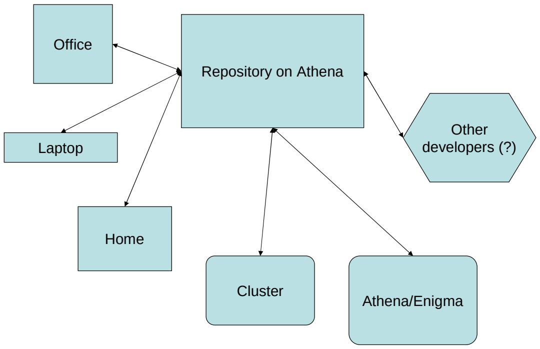
Version Control with CVS

Biostat 140.776



Office

Repository on Athena

Laptop

Other developers (?)

Home

Cluster

Athena/Enigma

Concurrent Versions System (CVS)

- Free software source code control tool
- Helps a developer (or multiple developers) track changes to files in a project
- “Nerd’s diary”
- Multiple developers?
  - You and some other people (using other computers)
  - You and yourself 3 months from now
  - You in the office and you at home (and your laptop, and….)

CVS as life management tool

- Repository and sandbox are redundant copies of a project (like a backup)
- Project files are in a central location so wherever you “are” you can always get the latest version
  - Great for syncing laptops with office computers when you’re running off to a conference
- Files can be rolled back to previous versions in case you make a really dumb mistake
  - Delete a file, incorporate a “brilliant idea”

CVS as project management tool

- CVS uses *repositories* and *sandboxes*
- Repository keeps the revision history of all files *checked in* to a project
- Sandbox is where you play – make changes to files
- Changes to files in the sandbox need to be checked in to the repository to be tracked
  - check ins can be accompanied by a comment
- A complete audit trail for each file

CVS caveats

- You need to be organized – changes don’t log themselves!
  - The rate at which changes are committed is person-dependent
- Files need to be stored in a central place accessible to all users/computers
- Sure, you get a backup, but now you need twice the disk space
- Not particularly useful for binary files
  - Word, PDF, data, images, etc.

CVS at Hopkins Biostatistics

- It’s useful to store the repository on a central server – the most logical choice is Athena (soon-to-be Enigma)
- Repository can be on your computer if a central server is not available
- Need ssh to use network client/server setup (i.e. repository on Athena)
- Home directory probably won’t have enough space – ask for a project directory

Some strategies for using CVS

- Use it for medium/long-term projects
- Best for software projects with lots of source files
- Also good for LaTeX projects
- Don't check in every file!

Basic CVS procedure

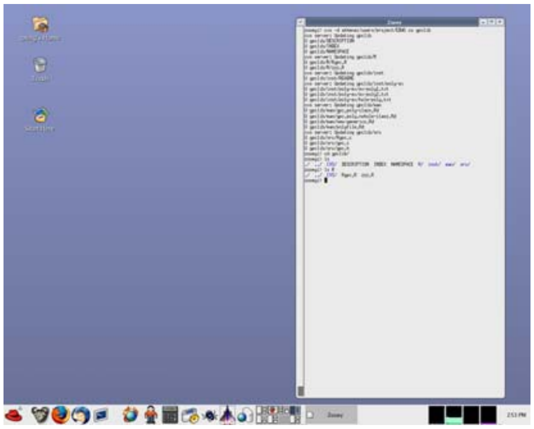
1. Designate a repository --- this can be a directory on your local computer or on a remote computer
2. Initialize the repository
3. Import a project --- all project files are now registered in the repository
4. Checkout the project files
5. Make changes
6. Check in all changes
7. Lather, rinse, and repeat 5 and 6

References

- http://www.cvshome.org/
- *Essential CVS*, by Jennifer Vesperman, ($40, or borrow mine)

CVS basics

- `cvs init` – create/initialize a repository (once)
- `cvs import` – import existing files to a new project
- `cvs co` – check out an existing project
- `cvs ci` – check in changes to files
- `cvs up` – update the sandbox with the latest changes (possibly made by other people)
- `cvs add` – add new files to the repository
- `cvs diff` – review differences in a file between versions
- `cvs tag` – mark a group of files with a single identifier



```
[illegible] cvs -d athena:/users/project/[illegible] co genlib
cvs server: Updating genlib
U genlib/DESCRIPTION
U genlib/INDEX
U genlib/NAMESPACE
cvs server: Updating genlib/R
U genlib/R/[illegible]
U genlib/R/[illegible]
cvs server: Updating genlib/inst
U genlib/inst/README
cvs server: Updating genlib/inst/[illegible]
[illegible]
[illegible]
[illegible]
cvs server: Updating genlib/man
[illegible]
[illegible]
[illegible]
[illegible]
cvs server: Updating genlib/src
[illegible]
[illegible]
[illegible]
[illegible] cd genlib/
[illegible] ls
./  ../  CVS/  DESCRIPTION  INDEX  NAMESPACE  R/  inst/  man/  src/
[illegible] ls R
./  ../  CVS/  [illegible]  [illegible]
[illegible]
```