2. CVS

Preliminaries

CVS =

1. a single database, called *repository*, the master (original) copies of files +
   all *revisions* to the files that have been submitted,
2. a workspace (or a set of them) where one does the work of modifying the originals, adding new files/components etc. The workspace contains copies of what's in the database.

Note: a particular file or directory in the repository is called a *module*

We *only* work on the copies in the workspaces and modify the originals through *cvs commands* (*never* touch repository directly).

We can have multiple workspace: one at school, one at home, etc. For each one, we need to

1. *check out* the copies *once*,
2. *synchronize* local copies with repository using *update* command
3. do some modification/addition/deletion on the local copies, and,
4. *submit* the changes back to the repository.

Environment Variables

1. **CVSROOT** tells CVS where to find its repository.
2. **CVS_RSH** tells CVS how to connect to remote machines.
3. **EDITOR** tells CVS what editor you want to use for log messages.

To set env vars,
with tcsh (cdf default): ***setenv CVSROOT <pathToRepository>***
with bash: ***export CVSROOT=<pathToRepository>***

Alternatively, we could use option: ***-d <pathToRepository>*** in the cvs command for CVSROOT

CVS Commands

All student repositories for this course are under: ***~studentid/cvs/cscb07f***
the module for the course contents: ***~studentid/cvs/cscb07f/contents***

(assuming CVSROOT has been set)

1. Starting a workspace: ***cvs checkout <module-name>***
   This creates a directory called <module-name> under the current directory, and populates it with

the latest versions of all documents in the module. E.g. a student might type:

mkdir b07

cd b07

cvs -d ~/cvs/cscb07f checkout contents

This will create "contents" under b07: ~/...../b07/contents/....

Normally, this would be done only once per workspace – students might want to do it on the ~/pc directory on their account: that directory is also visible on the lab PCs as their G: drive when they log in on campus. They might want to do it again on their home PC (more on how below). A major point to remember is that it *must* be done at least once; one *must not* modify any files in the repository itself.

2. Changing and checking back a change: ***cvs commit -m "Descriptive message" <module-name>***
   This creates a new version of the module in the database, and associates "Descriptive message" with it. If you forget –m "Descriptive message", cvs will open up some editor and wait for you to put the message there, and will not proceed without committing the change until you exit from the editor. Not knowing this can cause all kinds of pain and misunderstanding.... So, after doing some stuff for e01, you can issue:

   cd exer

   cvs -d ~/cvs/cscb07f commit -m "method xxx tested" e01

   or, if you want to check in just the one file that changed:

   cd exer/e01

   cvs -d ~/..../cscb07f commit -m "method xxx tested" Characterpair.java
3. Adding a file: ***cvs add <filename>***
   This alerts cvs to the existence of the file. It will respond with some message - but, in order to add the file to the repository, you must commit it at some point:

   cvs commit -m "Creating new file myfile" <filename>
4. Keeping the workspace(s) up to date: ***cvs update -d <module-name>***
   E.g., after a student has worked very hard on something at home on their hard drive, and has checked it back in, they want to retrieve it at school the next day - on fissure (students' mainframe – more about fissure:):

   cd ~/...../b07/ (i.e. the workspace)

   cvs - d ~/cvs/cscb07f/ update -d contents

   This updates the workspace so it has the same contents as the repository. If conflicts of the kind mentioned above are found, cvs reports them. Option "-d" means update all subdirectories of the module
5. Key Commands:
   a) ***cvs checkout [module-name]***
      Copy a module from a repository to the local directory. This command should only be run once.
   b) ***cvs add [-kb] [filenames]***
      Add one or more files or directories to the repository. If the file(s) should be treated as

binary (e.g. images), use the -kb flag.

c) ***cvs delete [-f] [filenames...]***

Delete one or more files from the repository. The files must also be deleted from your local machine; you can either do this before running the command, or by using the -f flag. Note that directories cannot be deleted (which is one of CVS's big weaknesses).

d) ***cvs update [-d] [filenames...]***

Update local files from the repository. If a file has changed in the repository, but has not been changed locally, the changes from the repository are copied to the local machine. If a file has been changed locally, but not in the repository, it will be marked as modified. If changes have been made in both places, CVS will signal a conflict, which must be resolved (by editing the file) before the local changes can be committed to the repository. The -d flag causes this command to work on sub-directories recursively.

e) ***cvs commit [-m message] [filenames...]***

Commit local changes to the repository. This command can only be run *after* cvs update (which compares local files to the repository), and only after conflicts between local changes and changes in the repository have been resolved. The -m flag can be used to provide a short log message on the command line. If it isn't used, CVS will launch the user's editor to get a log message.

f) ***cvs log [filenames...]***

Show the history of one or more files, including revision numbers, who made the change, and log messages.

Working from home vs at school:

1. At *school*:

   students have the option to use their fissure account - everything is simple. However, fissure does not support a decent IDE, and we want them to use one - so it's best if they use the

   IDE's on the lab pcs and work out of their G: drive workspaces. When it comes to checking stuff back in, (commit) they need to do it from fissure (from the ~/pc drive)

2. At *home*:
   a) use *ssh* to fissure
   b) work on local disk and use cvs remotely:
      i. *a ssh client* (e.g. the one from cygwin - or there may be a native Windows one) (NOT the standalone ssh terminal!)
      ii. set CVS_RSH to *ssh* (Control panel -> system -> advanced -> environment variables)
      iii. set CVSROOT to *:ext:myuserid@fissure.utsc.utoronto.ca:cvs/cscb07f*

Once these are in place, you can run the usual cvs commands as above, and must remember that the repository is now *remote*.

Notes for TAs

We now have the course material accessible on cdf. There are two repositories:

1. **csc207**: available to students, and,
2. **csc207instr**, for instruictors and TAs: here we have solutions, marking schemes, etc.

To use the contents of these remotely, do following

1. check out local copies:
   cvs -d :ext:youruserid@cdf.utoronto.edu:/u/gvwilson/repo/ co csc207
   and
   cvs -d :ext:youruserid@cdf.utoronto.edu:/u/gvwilson/repo/ co csc207instr
2. updates local copies (daily) to see what has changed:
   cvs -d :ext:youruserid@cdf.utoronto.edu:/u/gvwilson/repo/ update -d csc207
   cvs -d :ext:youruserid@cdf.utoronto.edu:/u/gvwilson/repo/ update -d csc207instr