CS 403: Lab 5: Version Control with CVS

Getting Started

1. Boot into Linux.

2. Go to the course website[1], and click on the RAD1D link. This will take you to a page with a description of how to run the RAD1D program. Click on RAD1D.tar to download the source code, then untar it `tar xvf RAD1D.tar`.

CVS

1. First, make sure RAD1D works: build it with `make` and run the example problem (set up with basic.cmnd).

2. Now, we will set-up CVS. The ACCEL machines have CVS installed, but no repositories exist. Note: this may not be the case on your personal system. The way we'll set it up our repository should work on any system, but it will be difficult for other users to check-in or check-out modules from it (due to our file permissions). If you are working with a large group, they may have already set-up a CVS repository for everyone to use–use this if you can, especially if you are collaborating with others. For our purposes, each of us will set-up a CVS repository in our own directory. First, create a directory where the repository will reside: in your home directory, enter `mkdir CVSreposit`. CVS interacts with two *environmental variables* called EDITOR and CVSROOT. Unix allows you to define variables, just like variables in a program. Environmental variables are like global variables that can be seen from several programs. We could set these variables from the command line, but our lives will be much easier if we tell Linux to set them up every time. To do this, you should add the variable assignments to your `.bashrc` file which is in your home directory. This file is a set of UNIX commands that are automatically executed every time you open an new terminal. To edit, first, open up the file (e.g. `nedit .bashrc`), and add the following lines to the bottom of the file

[1] `www.cs.cornell.edu/Courses/cs403/2002sp`

```
EDITOR=nedit
CVSROOT=/home/NETID/CVSreposit
export EDITOR CVSROOT
```

The first two lines define the value of the variables. EDITOR determines which program CVS will open for your log messages (if you want to use another editor, put its name on this line), and CVSROOT tells CVS where to store its files (you'll need to substitute your netID for NETID). The last line makes the variables visible to outside the file (otherwise they would be destroyed when the system is finished with .bashrc). To activate these changes, you can either quite the terminal and open a new one, or type `source .bashrc`. To verify that the variables are set, type `echo $EDITOR` or `echo $CVSROOT`.

3. Whew! You're now ready for your first CVS command. CVS commands have the following form:

   ```
   cvs <command> <options>
   ```

   where <command> is a CVS command. The first command we need is

   ```
   cvs init
   ```

   This creates a repository in CVSreposit. Specifically, it creates CVSROOT inside CVSreposit and creates several files and variables inside there. This is where any modules (projects) you create will be stored.

4. We want to create a CVS module for RAD1D, including the source code (*.c and *.h) files, the Makefile and our sample problem (C.txt and basic.cmnd). cd into the RAD1D directory and make sure that it contains only what we want (in particular, make sure that all object files and the executable are deleted). To create a RAD1D module, type

   ```
   cvs import RAD1D cs403 start
   ```

   This tells CVS to input everything in the current directory and save it in a CVS module called RAD1D (it creates a directory RAD1D and stores the files in there in a special CVS form). CVS requires you to declare a *vendor tag* (cs403) and a *release tag* (start). These tags are more interesting in a corporate setting where the vendor tag would be the company name or division (e.g. Microsoft Product Theft Division). CVS will produce some output and will then open an editor (`nedit` or whatever EDITOR is set to). You should make a note in the editor explaining what you did (hint: you created a module for rad1d), then exit the editor. This comment will be saved with CVS. Now, cd .. and remove RAD1D (`rm -r RAD1D`).

5. Now, let's check RAD1D out and make some changes.

   ```
   cvs co RAD1D
   ```

   (co stands for Check Out). This will create RAD1D directory with the files. Make some changes to a couple of files (add some comments, whatever you want).

6. Now, check it back in:

   ```
   cvs ci
   ```

   CVS will look through your directory to see if you've modified any of the files. It will then open an editor where you should describe the modifications. Your changes have now been added to the CVS repository. If you're done, you should release the module:

   ```
   cd .. cvs release -d RAD1D
   ```

   The release command tells CVS that you are no longer using the module. This feature is especially useful if you're working in a large group where several people may be working on the same module. The -d flag tells CVS to delete the RAD1D directory.

7. Check RAD1D back out. Let's look at the log information for a file that you changed:

   ```
   cvs log <file you changed>
   ```

This will tell you the information on the copy you're currently working with and any other versions. The useful information is the version number of the current copy (line starting with "head" at the top), the version numbers of other revisions (line starting with "revision", following a line of dashes), and the description of the changes (the comments you entered when you imported and checked in).

8. A major advantage of CVS is that it allows you to go back in time and see what you did (possibly, resurrecting an old version). One useful command is "diff" which compares two versions. To compare latest version (should be 1.2) with earlier version (1.1) of io.c (or whatever file you changed):

   ```
   cvs diff -r1.2 -r1.1 io.c
   ```

   The -r flag lets you indicate which versions to compare. The output is similar to the UNIX `diff` command.

9. To show you that we really can go back in time, let's get the original version. First, check in and release the copy you're working with. Then, to check out version 1.1, cvs co -r 1.1 RAD1D Are the changes you made there? Check in and release this version when you're done.