Lab #6
Version Control and RCS
Due In Lab, October 6, 2004

Name: _____

Lab Time: _____

Grade: _____/10

Basic RCS Commands

1. Go into your `~/csci3308/tmp` directory, and create a new directory called `lab06`. `cd` into this new directory. You will be using this directory to play with the basic RCS commands. We will be asking about changes that you see in the state of files in the directory. The easiest thing to do is use `ls -l` after each command to examine the contents of the directory.

2. Create a file named `versioned` containing the line `$Header$`. What are the size and permissions on `versioned`?

3. Perform a check-in on the file with the command `ci versioned`. (You can enter anything you like for the text of the description. Typically, the description is used to document the purpose of the file. Type a period on a line by itself to end the description.) What revision number was assigned to it?

4. What file is now in the directory? What are its permissions and size?

5. This new file is the version control file. Its contents are different than the original file `versioned`. The version control file is a plain ASCII text file.

You can look at its contents with a pager such as `more`, or a text editor such as `emacs`. Look at the contents of the file now.

Because it is an ASCII text file you could also edit the version control file with a text editor. In fact, the format of this file is specified in a man page. Type `man rcsfile`. If `man` cannot find this manpage try the command:

```
man -M/tools/cs/rcs/man rcsfile
```

You do not have to read this man page all the way through, and I don't recommend that you edit version control files by hand. However, it is interesting that once you understand the format, you can edit the file to make changes—such as removing a lock—without going through the `rcs` program. This has certain security implications for RCS files. In what sort of environment do you think RCS was intended to be used?

6. Perform an unlocked check-out with the command `co versioned`

7. A working file has now been created. Where did it come from, and what are its permissions?

8. Look at the contents of the working file. What did RCS do with the string `$Header$` that you put in the file?

9. Look again at the contents of the *version control file*. Does RCS perform substitution on `$Header$` when you check in the original file, or does it do the substitution when you check out a working file?

10. Try to perform a check-in with the command `ci versioned`. What happened and why? (Hint: look in your directory.)

11. Why at a conceptual level does RCS behave this way?

12. Remove `versioned`. Why is it safe to do so?

13. Perform a locked check-out with the command `co -l versioned`. What are the permissions on the working file.

14. Look inside the version control file. What line(s) tell RCS that a lock is being held on the file?

15. Perform a check-in on the file. What happened and why? What file or files are currently in the directory?

16. Check out the file with a lock again. Edit the working file to add some text. Now perform a check-in on the file. What revision number was assigned?

17. Now check out version 1.1 with a lock. You can do this with one of the following commands

```
co -r1.1 -l versioned
co -l1.1 versioned
```

18. Now edit the working file and check it back in. What version number was assigned?

19. Below, draw the version graph of the file at this time.

The RCS Directory

20. Create a directory named `RCS` in the current directory. Move the version control file `versioned,v` into that directory. Stay in the current directory and try checking `versioned` out and back in a few times.

21. Create a new file in the current directory named `control`. Check it in with the command `ci control`. Where was the version control file created?

22. Type `co RCS/*`. What happened?

23. It is important to be able to recognize what component performs what functions of a command. In this case the shell expands the pathname `RCS/*` into `RCS/control,v RCS/versioned,v` and then executes

```
co RCS/control,v RCS/versioned,v
```

then the `co` command finds the version control files and performs the checkout.

24. Delete both working files in the current directory. Type `co control`. Even when you do not specify the RCS directory the RCS program knows to look there.

25. Delete the working file again and type `co *`. Why doesn't this work? Shouldn't `co` look in the RCS directory for the version control files?

Mediating Among Multiple Developers

In this part of the lab everyone will be editing the same file. This simulates a situation where there are multiple software developers trying to edit the same code. Depending on what others are doing you may or may not gain immediate access to the shared file. If not, try again a short while later.

26. Go to your `~/csci3308/tmp` directory and type

    ```
    ln -s ~csci3308/RCS
    ```

    This creates a symbolic link to an RCS directory that you will be sharing with the other "software developers."

    NOTE: This RCS directory is currently "world writeable" for the purposes of this assignment. Even though you have the ability to place other files in this directory, DO NOT DO SO! In addition, do not erase or delete the CommentBook,v file.

27. Check out, with a lock, the file `CommentBook`. Write the command that you used below.

28. What is the current version of the file?

29. Type `rlog CommentBook`. When was that version checked in?

30. Who was the user that checked it in?

31. Edit the file by adding a short comment.

32. Check the file back in. *BE SURE TO RELEASE THE LOCK!* Write the command that you used below. Use the rlog command to verify that the file is unlocked.

33. What version number was given to the file?

34. Examine the version history of the CommentBook with the rlog command. Which user created version 1.1 and when?