Electronic Submission

In CS 70, we use an electronic submission system to handle programming assignments and most written work. The submission system handles getting the assignments, sharing the code between members of a pair-programming team, and submitting "working" versions of the code for testing or grading.

Note that CS 70 is using an entirely new electronic submission system this fall—be wary of any advice, help or tips you might be given about how the submission system works. Students who took CS 70 in prior semesters used an entirely different system. Also, be on the lookout for bugs. If you experience problems, please let us know by sending mail to cs70help@cs.hmc.edu.

Getting Files for the First Time

When you start a new assignment, there will be some files provided to you. The `cs70checkout` command gets these files and sets up the directory for an assignment. The syntax is

`cs70checkout` *assignment-name*

For CS 70, the assignment directory names are always of the form `cs70ass`*N*, where *N* is the assignment number. Thus, to start on the first assignment, you would type

```
mkdir -p ~/cs70
cd ~/cs70
cs70checkout cs70ass1
```

Checking in Your Files

You will be editing a "working copy" of your files, but the definitive repository for your CS 70 files will be a master copy kept in the grader account area. Your master copy can be accessed by both members of your programming pair and by the graders. You do not edit your master copy directly—you edit your working copy and then, whenever you have made a significant change (e.g., adding a new function or fixing a bug in an existing one), you should update the master copy using the `cs70checkin` command. You can and should check in your changes even if you aren't completely finished, *even if your code doesn't yet compile*. We will use this shared master copy when we are helping you debug your assignment, and examine it to determine whether you are making progress.

To update the master copy from your working copy, you should execute

```
cs70checkin
```

Optionally, `cs70checkin` can take a list of filenames specifying which files to check in. You may wish to use this form if you only want to update one of the files in the master repository.

When it is given no arguments, the `cs70checkin` only checks in files from the working directory that also exist in your master copy. Thus, if you create a new file, that file will *not* be added to the master repository unless you explicitly check it in by name, otherwise it will be ignored. In other words, by default, all files that didn't come with the assignment are assumed to be scratch files that you don't want to share.

For example, if you create a new file called `mytests.cpp` in your working copy of the assignment and you consider it to be part of your assignment, you should add it to your master copy by running

```
cs70checkin mytests.cpp
```

to add that file to your master repository and check it in. (Note that because you are using the form of `cs70checkin` where you explicitly specify which files to check in, this command will *only* check in the file `mytests.cpp` and no others.) Once the file is in the master repository, you can just type

```
cs70checkin
```

every time afterwards to check in the files in your assignment, including `mytests.cpp`.

If all of this seems too complex, you can just run

```
cs70checkin *
```

to check in everything in your working directory.

Resynchronizing with the Master Copy

If your partner has edited their working copy of the files, and then checked their changes into your (shared) master repository using `cs70checkin`, you can update your working copy to include your partner's changes by running

```
cs70resync
```

The resync process works *even if* you have also edited the same files, provided that you have not been editing the same lines. If you have both edited the same part of the file, the resync will produce a file that includes both parts, and you will have to merge the changes by hand. (Of course, if you are following the CS 70 pair-programming rules, the chances of this situation arising are remote.)

Remember that CS 70 requires that the vast majority of programming work to be done with both team members present and working together sharing a single computer or terminal. For example, writing new functions, making major changes, or tracking down an obscure crash, should all be done as a pair. By yourself, you are only allowed to make small fixes, correct or amplify comments, and generally polish the assignment.

Making a Submission

Checking in your files updates the shared master copy, but it does *not* count as a "submission" for testing or grading. To submit your files, you must run

```
cs70submit
```

You can think of `cs70submit` as making a copy of your files, exactly as they were in your master repository at the time of submission. (In fact, the master repository contains the entire history of your assignment, and can recreate any version of each checked in file—thus it just "tags" which version is the submitted version.)

Before performing the submission, `cs70submit` automatically runs `cs70checkin`. Any files you pass on the command line to `cs70submit` are treated as if they were arguments to the `cs70checkin` command.

You should run `cs70submit` whenever you've reached a major milestone. It is wise to submit code each time it seems to be "more-or-less working" even if it doesn't do everything it is supposed to (in case subsequent changes break something). In general, if it compiles and doesn't crash immediately when you run it, it's probably with submitting.

Some assignments support a special argument to `cs70submit` of the form

```
cs70submit -t
```

which submits your code and also runs tests on your submission and emails you the results.

Working Away from turing

We assume that you will be working on turing, either directly (on an NCD terminal) or over an `ssh` connection. It is possible to undertake the homework on your own machine(s), but you *must* keep your files on turing in sync with those on your machine. I will examine your files during the week of the assignment to see how much progress you are making—if you have not checked in any versions, I will have to assume that you have done no work. If you are using Mac OS X or Linux, the command

```
rsync -aC -e ssh source destination
```

may be a useful way to keep your files in sync (see the `rsync` manpage). For example, if you are `jsmith` on turing, you might run

```
rsync -aC -e ssh jsmith@turing.cs.hmc.edu:cs70/cs70ass1 ~/mycoursework/
```

on your personal machine to download files from turing, and then upload them after an editing session with

```
rsync -aC -e ssh ~/mycoursework/cs70ass1 jsmith@turing.cs.hmc.edu:cs70/
```

After uploading them, you would also need to log on to turing and run `cs70checkin` or `cs70submit`.

Peeking at the Internals

The CS 70 submission system is just a wrapper around CVS, a shared source-code control system widely used in team projects. If you are interested in the internal CVS commands used by the submission system, you can run any of its commands with `-v` as the first argument to print the CVS commands it executes.

Note, however, that you are not allowed to attempt to subvert the CS 70 submission system directly using CVS commands, or in any other way.