Electronic Submission

In CS 182, 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 182 is using a different submission system from the usual CS department submit system—be wary of any advice, help or tips you might be given about how submission works. Students who took CS 182 last did use this system, however, so their advice should be accurate. If you experience problems, please let us know by sending mail to cs182help@cs.hmc.edu.

1 Submission Model and Basic Commands

You will be editing a "working copy" of your files, but the definitive repository for your CS 182 files will be a master copy kept in the grader account area, where it can be accessed by the course staff and your programming partner. When you are working as part of a pair, both of you will access the same shared repository. You will not be editing your master copy directly—instead 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 `cs182submit` command. Your professors and graders will use the shared master copy when we help you debug your assignment, and will also examine it to determine whether you are making progress.

Because both members of the pair have access to the same shared repository, it means that you can both check out and review your code even when the other member of the pair is not around. There are severe limits on what you are permitted to do under the pair programming rules, but you can review the code and fix typos and minor bugs by yourself.

Your interaction with the submission system will be via two basic commands, `cs182checkout` and `cs182submit`. In addition, the `cs182resync` command is useful if your partner has submitted changes, and the `cs182checkin` command is useful for adding files to the list of files to be submitted.

1.1 Getting Files for the First Time

When you start a new assignment, there will be some files provided to you. The `cs182checkout` command gets these files and sets up the directory for an assignment. The syntax is

`cs182checkout` *assignment-name*

For example, if are in your cs182 directory and the first assignment was called `assign1`, you would type

```
cs182checkout assign1
```

to create a `assign1` subdirectory with a working copy of the files. If you already used your partner's account to check out the files, edited them and submitted your changes, when you run `cs182checkout` from your account you will get the version you both submitted, not the original unmodified files.

1.2 Submitting Your Files

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

```
cs182submit
```

(Optionally, `cs182submit` can take a list of filenames specifying which files to act on. You may wish to use this form if you only want to update one of the files in the master repository and *ignore* changes made to other files in your current working copy.)

When it is given no arguments, `cs182submit` 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. 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. (In the output from `cs182submit`, filenames preceded by the letter `T` are submitted, whereas filenames preceded with a question mark (`?`) are *not* submitted.

For example, if you create a new file called `pagetable.c` 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

```
cs182checkin pagetable.c
cs182submit
```

to add that file to your master repository and check it in.

Once a file such as `pagetable.c` has been added to your master repository, you can just type

```
cs182submit
```

from then on to check in the files in your assignment, including `pagetable.c`.

You should run `cs182submit` whenever you've reached a major milestone. It is wise to submit your 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 your code compiles and doesn't crash immediately when you run it, it's probably worth submitting.

1.3 Resynchronizing with the Master Copy

If your partner has made changes to their working copy of the files then submitted those changes using `cs182submit`, you can update your working copy to include your partner's changes by running

```
cs182resync
```

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.

If you get an error from `cs182submit` or `cs182checkin` telling you that you should run `cvs update`, it is a sign that you need to run `cs182resync` because your partner has checked in changes.

2 Advanced Features

The submission system has some other features that are sometimes useful, but you need not worry about in normal use.

2.1 Submitting individual files

You can give filename arguments to `cs182submit`, e.g., `cs182submit main.c`. This form of `cs182submit` is the form that explicitly specifies files to check in; thus the above command would only check the single file `main.c` into the master repository before marking the currently checked-in versions of everything in your repository as a submission.

2.2 Checking in without Submitting

Sometimes you may want to check in your files *without* it counting as a "submission" (e.g., because your submitted code *works*, whereas your latest modifications broke something).

To check in without submitting, run

```
cs182checkin
```

This command updates the master repository, but the modified files do not count as a submission.

2.3 Revisiting History

The submit system keeps a copy of *every* submitted version of your files. If you want to recover an old version of your assignment, you can use the command

`cs182checkout -O` *hours assignment-name*

This command will check out an earlier version of your assignment, from *hours* hours ago, into a directory named *assignment-name*`.old`. To avoid confusion, this directory is *not* a CS 182 working directory—you cannot submit from it. Instead you should copy any files you need from the old version into your working directory.

2.4 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. We will examine submitted 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:cs182/assign1 ~/mycoursework/
```

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

```
rsync -aC -e ssh ~/mycoursework/assign1 jsmith@turing.cs.hmc.edu:cs182/
```

After uploading them, you also need to log on to turing and run `cs182submit` (or `cs182checkin`).

2.5 Peeking at the Internals

The CS 182 submission system is just a wrapper around CVS, a 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. You will not be able to run these commands by hand, and you should not execute any CVS commands directly.

You are not allowed to attempt to subvert the CS 182 submission system directly using CVS commands, or in any other way.