CMSC 23000	**Operating Systems**	**Handout 2**
Autumn 2006		**September 26**

Programming assignment tips

This handout provides an introduction to some of the tools you will use to complete the programming projects.

Getting Started

You will need an account on the CS machines (this is different from your harper account). If you do not already have one, you can request one at

`www.cs.uchicago.edu/info/services/account_request`

You may work on your projects using either Linux or MacOS X. Both Linux and MacOS X machines are available for use in the CS Instructional Computing Laboratory (MacLab) located on the A-Level of Regenstein Library.

Doxygen

For this course, it is not sufficient to just write working code, you code should be well organized and documented. To help with the latter, we will use the Doxygen tool (`doxygen.org`) to generate documentation from source code. Specifically, we expect you to document each file, data structure, and function, as well as other significant definitions (*e.g.*, macros). You will be expected to hand in a hardcopy of your documentation as part of the project submission.

Makefiles

For each of your projects, you should include a makefile in your submission. We will provide a skeleton makefile for you, but you are responsible for maintaining it. If you have not used **make** before, you should take a look at the documentation. Information about make is available at `www.gnu.org/software/make` and online documentation can be found at `www.gnu.org/manual/make/html_chapter/make.html`.

Gforge

We are using a new system to keep track of projects called gforge. A server has been set up with hostname cs230.cs.uchicago.edu. You can access it using your web browser at that address.

Before you can have a project, you need to register yourself as a user. Do this by pointing your web browser to http://cs230.cs.uchicago.edu and clicking on the link "New Account" on the top right corner of the page. Follow the directions (you only need to fill in the starred fields) and submit the form. In a few seconds you should receive an email with a link to confirm your registration. Click on it and log in and you should have an active account. When this is complete, email the TA at `jriehl@cs.uchicago.edu` with the account name you just set up. As soon as I can, I'll set up your first project and email you when I do so. It will be called project1user, where user is your gforge username. Future projects will follow the same naming scheme. In order to be declared a member of the project, you must log in and go to the project's web page: http://cs230.cs.uchicago.edu/projects/project1user/. There is a link that says 'Request to join'. Follow it and click the submit button for the form on the linked page. I will email you when I approve you for the project and you will then be able to administer your project and use the Subversion repository.

Using Subversion

Once a project is created for you, it will have a Subversion repository on the server. You are expected to keep the source code of your projects in the repository. To *checkout* a copy of a project called project1user, run the following command:

```
svn checkout svn://cs230.cs.uchicago.edu/project1user
```

On your first checkout, you should be prompted for your password. It will assume you are using the username of the account executing svn. If your gforge username is different than this name, just press enter on the password prompt and it will then ask you for your username first and then your password. If everything checks out, a directory called `project1user` will be created in the current directory. All the files related to your project should live in this directory.

Now suppose you create a file called `main.c` in your `project1user` directory. In order for Subversion to keep track of it, it needs to be added to the repository. You do this using the following command:

```
svn add main.c
```

You should see a message like:

```
A         main.c
```

This command records the fact that `main.c` has been added to the repository, but the file will only be added when you commit your changes. To do so, type the following command:

```
svn commit
```

to add the file permanently to the repository. You will be prompted to enter a log message in an editor. You can avoid editors altogether by typing your log message on the command line with the `-m` flag:

```
svn commit -m "added files"
```

After you have entered your message, you will see a message like the following:

```
Adding         main.c
Transmitting file data .
Committed revision 1.
```

Changes you make to your files are recorded in the repository every time you do a `svn commit`. Before you make changes to your files, you can ensure that you have a current version, by running `svn update`. This fact is not of tremendous significance for individual projects, but matters when more that one person can modify the same files.

Not all the files in your project directory need to be in the repository. For example, you should not put your executable files in the repository — these can always be recreated (hopefully!) by compiling the source.

The "`svn diff`" command is for comparing differences between versions. If no files (or options) are specified, all working files are compared to their last committed versions, otherwise only the specified files are compared. There are also flags to compare other versions, see the man pages or the online manual for details.

Useful resources

Information about make is available at `www.gnu.org/software/make/` and online documentation can be found at `www.gnu.org/manual/make/html_chapter/make.html`.

The Doxygen home page is at `http://doxygen.org` and it includes an online manual.

The Subversion home page is at `http://subversion.tigris.org/`. Official documentation is at `http://svnbook.red-bean.com/`.