

Using the Zoo Workstations

Version 1.6: September 2, 2009

If you've used Linux before, you can probably skip many of these instructions, but skim just in case.

1 Logging on to the Suse Linux machines for the FIRST time

1. The Zoo is a collection of computers at the front of the third floor of AKW. There is a card-access area in room 300, and an open area at the south side of the building. Once you have an account, you may use the machines in person or remotely. In person is strongly recommended at first.
2. You must get a Zoo account for each Computer Science class that requires it. From the Computer Science Department web page, follow the "Course Web Pages" link; at the top of that page is a link "Apply for a Zoo Account." Follow that link and go through the sign up procedure for an account for Computer Science 201. Note that it may take up to an hour for your account to be created, so do this in advance (eg, now.)

It is important for computer science students to learn to use the Zoo machines. The best way to do that is to spend time in the Zoo when others are there, and to ask lots of questions. At first this will feel awkward, but soon you'll be one of those answering questions, and you may find you have some new friends. The Zoo (Computing Lab) web page, accessible from the Computer Science Department web page, has some useful information about the Zoo.

3. To log in once you have an account, find a machine that is not being used. It should display a message box inviting login. (Click with the mouse if the screen is dark.) Type your netid and password. Shortly you should see the desktop.
4. Right click on the desktop and select "Open in Terminal." This should open a terminal window with a command prompt something like `dca3@tick:~>`. You type your commands at this prompt, for example, `emacs` to start emacs and `mzscheme` to call scheme. You may open several terminal windows, for example, one for emacs and one for scheme.
5. To log out after a session, make sure to save all your work, and then click on "Computer" at the lower left corner of the screen. Select "Logout" in the menu, and select the option to "Log out."

2 Running Emacs

Emacs is a text editor. It will allow you to write and edit text files, including the scheme programs you will be writing for this course. To start emacs, type `emacs` at the shell prompt; if you are sitting in the Zoo, you will get a new emacs window; what happens with ssh will depend on how it is configured. You can learn about emacs by going through the tutorial (Select "Emacs Tutorial" under the "Help" menu in the Emacs window); the whole tutorial may take about an hour. Of course you can exit before finishing the tutorial.

Emacs uses the Ctrl and Alt (or Esc) keys a lot. Locate them before you get going. It also offers a lot of functionality in point and click style, if you prefer that. The concept of a "buffer" is that it holds a copy of your file while you edit it – the changes are not made permanent until you explicitly save. You can be viewing several files simultaneously in several buffers; copy text between them, etc. Emacs can recognize scheme syntax and can help you balance parentheses and indent code in a reasonable style (use `tab`.)

Here are some useful commands.

- `CTRL-x CTRL-c` (type them in succession) will let you exit Emacs, asking if you want to save any files that haven't been saved.
- `CTRL-x CTRL-s` will save the file in the currently active buffer (do this often when you start programming, so that you don't lose anything).

- **CTRL-x CTRL-f** will prompt you at the bottom of the Emacs screen to name a file. You can either type the name of a file that exists, in which case Emacs will load it into the current buffer, or you can type the name of a new file, in which case Emacs will create a new file by that name. You can keep loading more files into a single window—they will be hidden, but they are still there. To find other buffers, try:
 - You can use the pull down menu “Buffers” at the top left, *or*
 - **CTRL-x b** to switch buffers. This will prompt you to enter the name of a file that has already been loaded into Emacs.
- Use the arrow keys on the keyboard to move up, down, left, and right. You can also use **CTRL-p** to move up, **CTRL-n** to move down, **CTRL-b** to move left, and **CTRL-f** to move forward.
- The “Page Down” button or **CTRL-v** moves you a page forward.
- The “Page Up” button or **ALT-v** moves you a page back. (**ALT-x** is also written **M-x** in Emacs documentation).
- **CTRL-l** centers the page so that whatever line the cursor is on will be in the center of the window.
- The “End” button moves you to the end of a line, and **CTRL-End** moves you to the end of a file.
- The “Home” button moves you to the front of a line, and **CTRL-Home** moves you to the front of a file.
- **ALT->** moves you to the end of the file.
- **ALT-<** moves you to the beginning of the file.
- **CTRL-d** deletes the character under the cursor.
- “Delete” deletes the character before the cursor.
- “Insert” switches Emacs between “overwrite mode” and “insert mode.”
- **CTRL-k** kills everything at and after the cursor on that line (and sticks it in the clipboard).
- **CTRL-@** sets a “mark.” Once you set a mark, you can define a “region” by moving the cursor to the other end of the region.
- **CTRL-w** cuts a region (and puts it into the clipboard).
- **ALT-w** copies a region into the clipboard.
- **CTRL-y** yanks whatever is in the clipboard to wherever the cursor is (like pasting).
- You can also copy and paste regions by selecting them with the mouse. Press and drag with the left mouse button to select a region. Click on the middle button to paste it.
- **CTRL-_** (**CTRL-SHIFT-hyphen**) lets you undo the last operation. Pressing it twice in succession causes Emacs to undo the last two operations, and so on.
- **CTRL-x 2** (type “**CTRL-x**”, then “**2**” by itself) will split the Emacs window into two. You can have two different files displayed at once using this feature. To go back to one window, same command with 1 instead of 2.
- **CTRL-x o** will move the cursor into the other window, in the case that you have two windows.
- **CTRL-x 1** will hide the other window, in the case that you have two windows.
- **ALT-x** will prompt you to enter some command which is not necessarily bound to a keystroke, but which is still a command in Emacs.
- **CTRL-g** will stop any operation—if you get stuck in Emacs, it often helps to hit **CTRL-g** a few times.

If Emacs beeps at you, see what it says at the bottom of the window. Often, it will be prompting you for a “yes” or “no”.

3 Running scheme

Here's a simple way to run scheme and emacs – you'll learn more sophisticated ways. Create two shell windows, open emacs in one, and give the command `mzscheme` to start scheme in the other. In the scheme window you should see something like:

```
...> mzscheme
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
>
```

That last “>” is the mzscheme prompt where you can type scheme expressions. Try typing in:

```
> (+ 1 2 3) <---- you type this
6           <---- the interpreter responds with this.
```

In the emacs window use `CTRL-x CTRL-f` to load your current file (or create a new one if you don't have one yet). All of your scheme programs should be in files with `.scm` at the end of the filename. This enables a “scheme” menu in emacs that lets you comment and uncomment regions of your file; emacs will also help you match those pesky parentheses! Create a file `test.scm` containing the following procedure definition.

```
(define my-add          ;; This function adds 2 numbers.
  (lambda (x y)
    (+ x y)))
```

You'll learn how this simple function works in class, but for now, just believe that it does the relatively silly task of adding two numbers using scheme's predefined “+” function. After you've typed that in, press `CTRL-x CTRL-s` to save the file. In the scheme window type the following load command at the prompt.

```
> (load "test.scm")
>
```

Then try

```
> (my-add 5 6)
11
```

If you want to change your program, simply edit the file `test.scm`, save it, and load it into mzscheme again.

The text of each assignment will be available in the directory `/c/cs201/handouts`. We suggest that you copy the assignment file, for example, `/c/cs201/handouts/hw1.scm`, to your own directory as file `hw1.scm` and edit it to include your solutions. Don't forget to include your name, email address, and an estimate of how long the assignment took you. Starting with the second assignment, you will be required to submit your assignment electronically using your Zoo account. The file name for the second assignment should be `hw2.scm`, for the third assignment `hw3.scm`, and so on.

We will initially request that you hand in sample runs of your procedures. To do this, test your procedure in the scheme window, cut both the input and the outputs, and paste them in the `emacs` window below the code for the procedure being tested. Comment out the test lines by putting a semicolon at the beginning of each line, or by using the `emacs` comment capability. For problems that require answers in English, also comment out the lines you write. This will enable the TA to load your file into scheme and test the procedures you have defined.

To quit scheme, type

```
> (exit)
```

You'll be returned to the shell prompt in that window.

4 Manipulating files in the C-shell

A few useful shell commands are described below.¹ Many of the commands deal with directories. Directories, like folders in the desktop interface, organize your files in a hierarchical way. Directories can contain files and other directories. When referring to a directory within a directory within a directory..., you separate the names of the directories with a “/”. Thus, your “home” directory (the one you get placed in when you log on) is `/home/accts/your_name`. (This is called the “pathname” to your directory. Since it’s long, all of that is conveniently abbreviated by `~` e.g., files in your home directory would be called `~/file1`, `~/file2`, and so on).

Here are some useful commands.

<code>pwd</code>	See what directory you are in. (Upon login, you should be in the directory <code>/home/accts/your_name</code> .)
<code>ls</code>	List what’s in this directory.
<code>cd blah</code>	Change current directory to the one named “blah”. Note that “blah” must be a directory in the current directory for you to be able to cd to it.
<code>cd ..</code>	Change current directory to the one that contains this directory.
<code>mkdir foo</code>	Make a directory named “foo” in the current directory.
<code>rmdir bar</code>	Remove a directory named “bar” in the current directory (the directory must be empty before you can remove it).
<code>rm file1</code>	Remove the file called “file1”.
<code>cp file1 file2</code>	Make a copy of “file1” and call it “file2”. If “file2” is the name of a directory, a copy of file1 called “file1” will be created under the “file2” directory.
<code>mv file1 file2</code>	Rename “file1” to “file2”. If “file2” is the name of a directory, “file1” will get moved to the “file2” directory.
<code>more file1</code>	Show the contents of “file1” one screen at a time.
<code>man some_command</code>	Display the online manual pages for “some_command”.

Finally, you can print out a text file with the “`enscript`” or “`lpr`” command. Use this to print out your programs when you hand them in.

```
enscript file1
```

---OR ---

```
enscript -2r -Pzoo1 file1
```

The “`-2r`” will conserve paper (it fits two pages side by side on a single sheet), and “`-Pzoo1`” will print out on the main printer in the Zoo. There is a second printer called `zoo2`.

Since “`enscript`” sometimes has problems, if it doesn’t work, you can also print out your programs with the more primitive “`lpr`”:

```
lpr file1
```

or

```
pr file1 | lpr -Pzoo1
```

“`pr`” adds simple headers to `file1` and “`-Pzoo1`” will print out on the main printer in the Zoo.

¹Also see <http://www.tldp.org/LDP/gs/node5.html> for a quick overview of Linux. Take a look at “3.2 Basic Linux concepts,” “3.3 First steps into Linux,” “3.8 Wildcards,” and “3.10 File Permissions.”

5 Ready to depart the Zoo?

Make sure to save all work you've done in emacs. Click on "Computer" in the lower left corner of the screen and select "Logout" on the drop-down menu. The display should return to the login screen.

6 Administrative problems

If you have problems with one of the computers, or if you have problems with your account that are not CS 201 specific problems, first ask a friend for help. Then try sending e-mail to "dsac". (Dsac is a committee of undergraduates who are partly in charge of the Zoo.) If none of that works, send mail to "requests". E-mail sent to that address is handled by the CS Facility which is in charge of systems administration for the Zoo.

7 Final comments

This handout should tell you much of what you need to know to get you started in this course. Please go to the Zoo and ask lots of questions to fill in any gaps. There is a lot more to Linux and to Emacs, however, so you may want to play around with them a bit, go through the tutorials, and ask knowledgeable friends for help before you get your first real assignment.

The TA(s) will be happy to answer any questions you may have about anything related to the course (if the instructions here don't make sense or don't work, if lecture material is unclear, if you get stuck on a problem, etc.), but they might be less willing to deal with questions like "How do I change the color of my Emacs window?" and "Where is the Tetris game in emacs?"