A Programmer's Toolkit

How to use the OS X and Unix tools of CS136

Programmers use many tools in many environments. This handout introduces some of the most basic tools that are common across multiple platforms. You'll use these extensively in CS136. Although these are more difficult for some people to learn than the GUIs of Microsoft Word and the Mac OS Finder, most programmers find that once learned these tools are easier to use, more precise, and more powerful than their GUI equivalents. Once you have read the full document, you will probably want to refer to the Quick Reference sections at the end in the future.

1 The CS Lab

1.1 Log In

To start, you must log in to one of the Macintosh computers in our lab. Begin by entering the user name and password for your computer science account in the login window and click ``Log in". Your user name for this account will be the same as that used for computer accounts provided by the Office of Information Technology (typically something like 06ewr), but the initial password will be different. You must obtain this password from your instructor. Once you log in, the screen should display a window containing icons for some of the files you can access on the machine.

1.2 Change Your Password

Before proceeding, you should change the password we assigned to your account to something that only you will know and that will be easier for you to remember. Don't, of course, make your new password too easy to guess. In particular, at least use something that is a mix of letters and numbers. If you forget your password, the system administrator can give you a new password. To change your password you should:

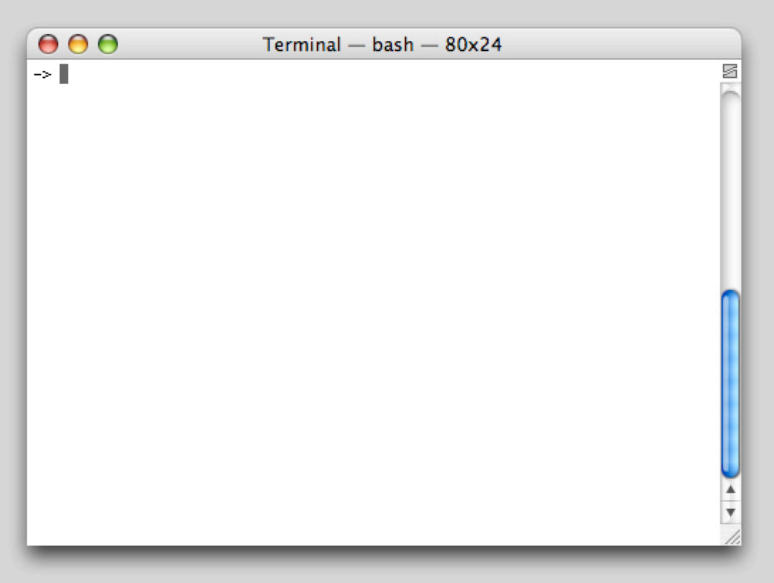
1. Point the mouse at the blue Apple icon in the upper left corner of the screen. Depress and hold down the mouse button to open a menu.
2. Select "System Preferences" from this menu.
3. Click once on the "Accounts" icon in the window that appears.
4. Change your password in this window.
5. Close the window by clicking in the red circle in the top left corner.

Now, log out and log in again with your new password. To log out select the last item in the menu that appears when you depress the mouse on the little apple image that appears in the upper left corner of your screen. Then click the "Log out" button that appears in the dialog box. The files you create for our course laboratory projects will not actually be stored in the computer on which you are working. Instead, they will be stored on our department's Macintosh file server, cortland. By entering the account information for your account, you have instructed the computer you are using to access your files on cortland. You can access your files on cortland from any computer in our lab by simply logging in to that computer.

1.3 Using the OS X Interface

Some of you may be new to Macintoshes, and the section highlights the most important elements of interface. The files on the hard drive can be viewed by clicking on the "Macintosh HD" icon in the top right-hand corner of the screen. To go to your home directory, click on the house icon in the Finder window.

All Mac windows look like the following window:



In fact, that particular window is the one that you'll be spending most of your time in. If you do not have an icon labeled "Terminal" that looks like this:

on the dock area at the bottom of your screen, then locate it in the Applications folder on the hard drive and drag it to your dock. When we say "launch a new terminal window", we mean either click on that icon or press Command-N if terminal is already running and active.

Here are some basic window management operations:

- To change the size of the window, click the mouse in the lower, right corner and drag the mouse with the button held down.
- To close a window, click on the red circle in the title bar.
- To iconify a window, click on the yellow circle in the title bar. The window should appear as an icon at the bottom of the display in the tool bar. Clicking on the icon in the tool bar will restore the window.
- To maximize the size of a window, click on the green circle in the title bar. Clicking again will restore a window to its previous size.

You can start to get the feel for the environment by clicking on the Safari icon to start a web browser. The safari icon is in the doc and looks like the following:

Go to the class web page at http://www.cs.williams.edu/~morgan/cs136 and find the electronic version of this page. Bookmark this page.

2 Entering Unix Commands

We sometimes refer to the terminal window as the "command line". More specifically, the line that you can edit, which ends with the blinking cursor is the actual "line" at which you issue commands. This is also casually known as the Unix prompt, terminal, and the shell or shell window. Almost every major operating system has one of these, and all are derived from an older operating system called "Unix."

The command line allows you to access all of the files and folders (which we call "directories" in this context) on your computer, and to run programs.

The Terminal Program will open a shell window in which will type Unix commands. For example, type the command "ls" in the shell window and press return. This command will list the files in your directory. A command consists of the name of a program, zero or more options preceded by '-', and zero or more arguments, which are themselves words. The command line is terminated with a carriage return.

If you make mistakes in typing on the command line, use the Delete or Backspace key to erase the previous character.

Here are other useful commands related to files:

ls	List the files in the current directory
cp file1 file2	Copy the file named file1 to a new file named file2
mv file1 file2	Rename the file named file1 to have the name file2. (mv is actually short for move, but it doesn't necessarily move anything!)
rm file1	Remove the file named file1
more file1	Display the contents of a file
lpr file1	Print the file named file1

One thing you will notice with Unix is that "no news is good news". Often if a Unix command is successful, it will produce no output! If it fails, you will receive an error message. Of the list of commands above, only ls and more produce output when they succeed.

The "more" command deserves some more comment because it is an interactive command. It displays your file one screenful at a time. You can try out "more" by typing[1]

```
more /usr/mac-cs-local/share/cs136/examples/alice.txt
```

After each screenful, it displays a prompt either consisting of the name of the file being displayed in reverse video (for the first screenful), (END) in reverse video (for the last screenful), or : (for all other screenfuls). The prompt appears at the bottom of the screen. At this prompt, you have several options, the most useful being:

space	Display the next screenful
b	Display the previous screenful
q	Quit more, returning to a shell prompt.

This will show you a file containing the first chapter of Alice in Wonderland. (One useful tidbit: hitting the tab key will autocomplete directory and file names. Try typing "/usr/mac" and then hitting the tab key.)

When you use ls to list your files, a directory will appear with / at the end of its name. At all times you have a current directory. Your commands are interpreted with respect to your current directory by default. So, for example, ls lists the files in your current directory if you give it no arguments. Here are some useful commands to manipulate directories:

mkdir dir1	Create a new directory named dir1.
rmdir dir1	Remove the directory named dir1. You can only do this

[1] Note: at the command line, the "tab" key provides completion for filenames. If you type just "more /usr/m" and press tab, it will suggest /usr/mac-cs-local, saving you some typing

	if the directory is empty.
cd dir1	Make dir1 be the current directory
pwd	Display the name of the current working directory

Filenames should consist only of letters, numbers, and periods, '.'. A filename is typically divided into a descriptive name and an extension, separated by '.'. Extensions are purely by convention but typically indicate the type of file. For example, "myfile.txt" would be a text file, while "myfile.java" would contain a Java program. Extensions are generally optional although some programs, such as the Java compiler, expect them to be there.

Filenames can be given as names that are relative to the current working directory or as absolute names from the special root directory. Thus far, we have assumed relative names that refer simply to files in the current directory. We can also use relative names to identify files in subdirectories. For example, "mydir/myfile" is the file named "myfile" in the directory named "mydir", where "mydir" is located in the current working directory. The special directory name "." Always refers to the current directory, and ".." always refers to the parent of the current directory. So "cd ../.." means go back up two directories in the hierarchy.

Each user also has a home directory. This is the directory that is the current directory when you start a new shell window. (You have already found your home directory through the standard Mac interface). No matter what the absolute pathname is to your home directory, you can always refer to it with the special relative pathname '~'. For example, no matter what your current directory is, if you type "ls ~", you will see a listing of the files in your home directory.

An absolute pathname always begins with / while a relative pathname never does. To find out the pathname that corresponds to your current directory, type pwd. Use pwd to find the absolute path for your home directory.

One last useful feature of the command line: Pressing the up arrow will give the last command you executed. Pressing return will run that command again. Try this. You can access older commands in your history by hitting the up arrow multiple times. To practice these commands, copy the alice.txt file to your home directory with the command

```
cp /usr/mac-cs-local/share/cs136/examples/alice.txt ~/alice.txt
```

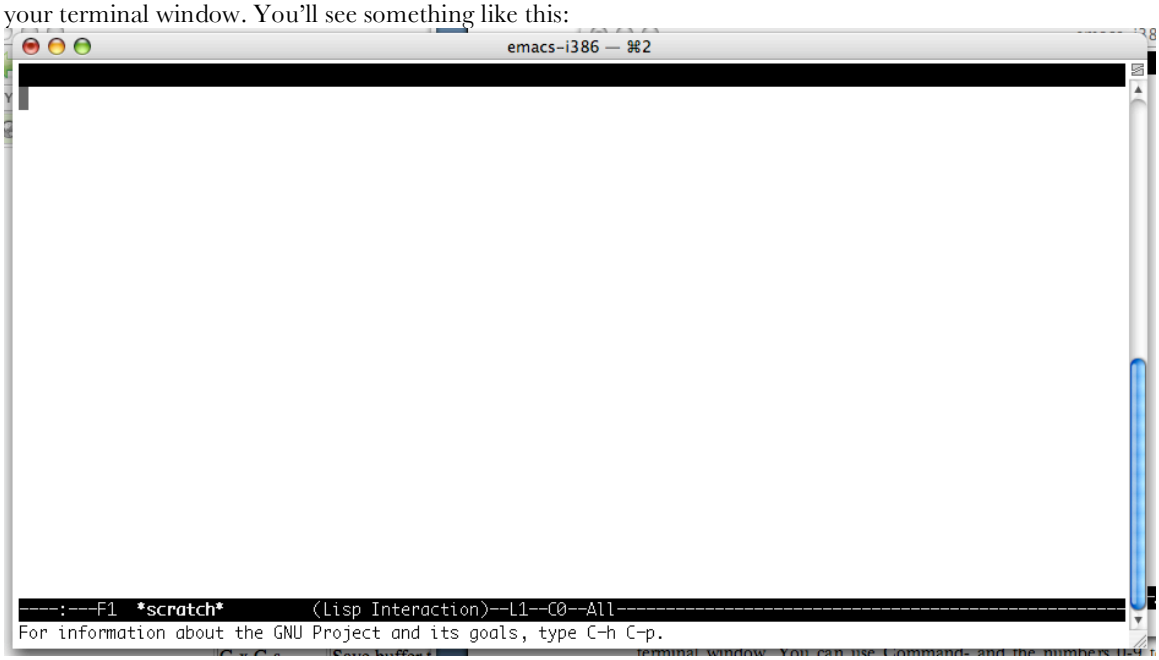
Rename the file, move it into a newly created subdirectory, and practice the other basic Unix commands as well.

2.1 Additional Terminal Windows

You can create additional terminal windows to work in more than one directory or on different tasks at the same time. When any terminal window has the focus (by clicking in it), the menus across the top of the screen should be "Terminal", "File", and so on. When this is the case, press Command-N to create another terminal window. You can use Command- and the numbers 0-9 to flip between open terminals. Press Command-W to close one terminal window and Command-Q to close them all.

3 Emacs

Almost all of your work in CS136 will be inside a program called Emacs. Emacs is a free open source code editor available for almost every operating system. It is controlled by a quirky set of keystrokes that quickly become second nature. To run emacs, type "emacs" at a terminal window. Emacs will then take you're your terminal window. You'll see something like this:



Note that Emacs is running in text mode. It is exclusively using text characters to build the user interface. This is handy because it can run in situations where graphics are not available, such as when you're remotely connected into a machine. (There's also a graphical version of Emacs called XEmacs. We're going to stick to the text version for now.)

Emacs relies heavily upon control-characters as a means of entering editing commands. For example, to create a new file, press Control-x followed by Control-f. That is, hold down the control key and press x and then f. At the bottom of the window, you will see "Find file: ~/" with a small black box following it. The box indicates where characters will go when you type. Type in the name of the file that you want to create and hit carriage return.

The last line of the Emacs window is where command prompts and messages appear. The embossed line immediately preceding it is a status line. It shows you the name of the file you are editing. Before that you should see -1:--. This means that the file is up-to-date. If it says, -1:**, it means the buffer has been changed since it was last saved. If it says -1:%%, it means that you cannot modify the file. After the filename is the current line number (L1 indicates the first line) and and indication of how far the current line is in the file in terms of percentage. "All" indicates that the entire file is showing. "Top" indicates that you are on the first screenful. "Bottom" indicates that you are on the last screenful. The final entry is is the "mode" that you are editing in. This should say "(Fundamental)", or "(Java)" when editing a Java file.

The area between the top of the window and the status line is the buffer that displays the contents of the file. Most of your typing goes directly into this buffer. Type some text. Notice the status line changes. Save the buffer by typing Control-x Control-s, typing a file name, and pressing enter.

Almost all Emacs commands are accessed by holding either Control (C-) or Meta (M-, which on Apple keyboards is the "Alt" or "Apple" key; you can also press and release Escape to type Meta). Often there is an initial sequence like "C-x" that corresponds to choosing a menu (in this case, the file menu) and then a second key sequence that corresponds to the actual command. See the appendix to this document for are some basic Emacs commands. Try them all out right now.

4 Java

The Java compiler and virtual machine are two of the programs that can be run from the command line, either directly in a terminal window or from a shell window inside Emacs. The name of the Java compiler is "javac" and the name of the Java virtual machine is "java". To do anything useful with these, you have to provide various arguments. To see all of the arguments for javac and most other Unix programs, just run the command with the "--help" argument. That's two minus signs and no quotes, like:

```
javac --help
```

To compile all of the Java files in the current directory, using the CS136 "bailey.jar" support files, you would type:

```
javac -cp .:/usr/mac-cs-local/share/cs136/lib/bailey.jar *.java
```

This compiles each of your source files like Date.java into a binary file like Date.class. If it encounters any errors they are printed to the screen and compilation stops. One of your classes is special; it has a public static void main(String[] args) method. Say that class is named Main (it doesn't have to be). Once compiled, you can run it as:

```
java -cp .:/usr/mac-cs-local/share/cs136/lib/bailey.jar Main arguments
```

Anything that you type for *arguments* becomes the array of strings passed to Main.main. You can generate nice HTML documentation for your classes using the command:

```
javadoc -classpath .:/usr/mac-cs-local/share/cs136/lib/bailey.jar -d html -tag pre:cm:"Precondition:" -tag post:cm:"Postcondition:" *.java
```

(that's all one line; it just didn't fit the layout of this document).

4.1 Configuring Your Environment

If you had to actually type that whole /usr/…/baily.jar every time you compiled it would quickly become annoying. Therefore we provide you with a script that modifies your environment to include that by default. The catch is that to run our script, you have to type a path that is almost as long. So you need to manually make a small change to your environment, and then can quickly run our script to make the remaining changes.

Open your ~/.bashrc file in Emacs and add the following line to it:

```
export PATH=.:/usr/mac-cs-local/bin:$PATH
```

Save the file and quit Emacs. That line tells your shell to look in the above path for programs whenever you try to run one from the command line. To update your environment with this change, at the terminal window type:

```
source ~/.bashrc
```

You don't ever have to make these changes again. Now, whenever you first open a terminal window, just type:

```
source 136
```

That command will run the script /usr/mac-cs-local/bin/136, which executes the following commands for you:

```
export MANPATH=/usr/share/man:$MANPATH
export PATH=.:/usr/mac-cs-local/bin:$PATH
export CLASSPATH=.:/usr/mac-cs-local/share/cs136/lib/bailey.jar
java -version
java structure.Version
echo "You're set up for 136."
```

The first command sets up some additional variables to allow you to view documentation at the command line. The second line ensures that the CS department programs are in your path (which they already should be, since you edited your .bashrc). The third line makes it so that you don't need the –cp argument for javac anymore. The remaining lines just print diagnostic information to help verify that you are running the right version of Java.

Now, you can compile and run the easy way using:

```
javac *.java
java Main arguments
```

You can generate documentation using the script that we provide:

```
javadoc136 *.java
```

Since you'll likely be executing those commands repeatedly, it helps to remember that in an Emacs shell you can re-execute the previous command by typing M-p, and that the up arrow performs the same function in the terminal.

5 Unix Reference Sheet

File Management

ls *directory*	List the contents of *directory* (which is the current one if not specified)
cp *source dest*	Copy *source* file to *dest* file
mv *source dest*	Rename or move *source* file or directory to *dest*
rm *file*	Delete *file* (permanently!)
mkdir *dir*	Create a new subdirectory
rmdir *dir*	Remove an empty subdirectory
cp –R *source dest*	Copy a whole directory tree
rm –rf *dir*	Remove a while directory tree
pwd	Print the current directory (handy if you get lost)

Looking at Files

emacs *file*	Open *file* in Emacs, or just run Emacs if not specified
more *file*	Print file on the command line, pausing between pages
clear	Erase the current contents of the screen (Command-K on OS X to really wipe everything)
open *file*	Open the specified *file* (which can be a directory) as if it had been double-clicked in the Finder. This is OS X-specific.

Java

javac *.java	Compile all of the source files in the current directory
java *Class args*	Run Class.main(), passing it *args* as an array of strings

Getting Help

man *command*	Print a description of a Unix command and its options, e.g., "man ls"

CS Department Scripts

source 136	Configure the CS136 environment; run every time you open a terminal
turnin –c 136 –d *directory*	Submit *directory* as your solution to an assignment in CS136. Will prompt you for your password.
javadoc136 *files*	Generate HTML documentation for *files* into the subdirectory "html"
cleanup136	Delete all .class and ~ files and any html subdirectory of the current directory; useful for cleanup up a solution to hand in.
cleanup_firefox	Delete old Firefox lock files that can make that program misbehave.

See also http://www.indiana.edu/~uitspubs/b017/ and http://sunsite.utk.edu/UNIX-help/quickref.html

6 Emacs Reference Sheet

File

C-x C-f	Open file or new file…
C-x C-s	Save
C-x C-w	Save as…
C-x C-c	Exit Emacs
C-x k	Close one buffer (but not Emacs)

Cursor Movement and Editing

C-f	Cursor forward/right (you can also use arrow keys)
C-b	Cursor back/left
C-n	Cursor dowN
C-p	Cursor uP
C-v	Page down
M-v	Page up
C-d	Delete
tab	Indent the current line appropriately for the current programming language
C-_	(Control-shift-minus) Undo
G-g	Cancel whatever command you were in the middle of

To jump to a specific line, type M-x followed by the words "goto-line" without the quotes. A new prompt will ask you for the line number. This is very handy when the Java compiler reports an error on a specific line of your program.

Notice that the scrollbar on the right side of your terminal does **not** scroll the Emacs buffer. It scrolls the terminal window, which is not useful to you.

Cut and Paste

You can paste from another OS X window into a terminal window using Command-V (not Control!). You can copy from a terminal window by selecting an area with the mouse and pressing Command-C. When copying and pasting within Emacs, do the following:

C-space	Place mark
C-w	Cut from previous mark
M-w	Copy from previous mark
C-y	Paste copy buffer
C-k	Cut the current line. If pressed repeatedly without other keys in between, this will cut a series of lines that can then be pasted.

Window Management

Emacs can show you views of multiple files (buffers) or multiple views of a single file at a time.

C-x b	Switch to a different buffer
C-x 2	Split the current view horizontally (you can execute this recursively!)
C-x 3	Split the current view vertically (you can execute this recursively!)
C-x 1	Collapse to a single view
C-x o	Move the cursor to the next view
C-x C-b	Show a menu of all available buffers

Shell

You can run other programs (including a new shell) inside Emacs. This makes it really convenient to compile without switching windows. To open a shell, type M-x and then the word "shell" (without the quotes). Within the shell buffer you can use all of the other Emacs commands, plus these:

M-p	Scroll backwards through previous commands that you typed in the shell.
C-c C-c	Kill the currently running program inside this shell.

See also http://cs.williams.edu/~freund/cs136/unix/emacs.html and http://www.hsrl.rutgers.edu/ug/emacs_qref.html