Unix Tools	**Lecture 4**
CS114 Fall 2002	**Monday, October 7, 2002**

Finally, after a week of dreary file and directory manipulations, we get to actually create and modify files, a process known as *editing*.

What's a file? A (text) file is just a sequence of characters split into lines.

Early editors were line editors. One such editor in Unix which is still available is `ed`. The idea is simple: you load a file from the filesystem into memory, you modify the version of the file in memory, and when you are done, you save your changes back to the filesystem. Until and if you save your changes, the original file on the filesystem remains unchanged.

In many ways, `ed` was a pain: you could give commands such as "display lines x to y", "replace line x by what I'm going to type next", "delete line x", "insert the following line before the current point" (a point is just a moving pointer in the file). In other words, you have to have some kind of idea of the layout of the file in your head to get work done.

(Nonetheless, `ed` is still useful once in a blue moon. For example, if you can't get any of the fancier editors going because you have messed up one of your configuration files, you can always use `ed` to change the configuration files. But this is becoming a rarer and rarer occurrence. Another use of `ed` is when you want to instruct the machine to make changes to a file. You can build a file containing essentially `ed` commands on how to change the file. This approch is used for example by `diff`.)

It is much easier to do things visually, that is to have the text on the screen and modify it directly by for example moving the cursor to the point of the text where you want to make changes. One such editor, built on top of an editor such as `ed`, is `vi`. This editor is very popular, and in fact comes in many versions.

Introduction to Emacs

I will not describe `vi` in this lecture, but rather another very popular text editor called Emacs. Invoking Emacs to edit a file is a simple matter of calling `emacs` *`file`*.

When you start Emacs on a file, you get a full screen display, split into a large *edit area* where your file appears, a separator line at the bottom of the screen containing information such as the name of the file, etc, called the *mode line*, and a smaller area at the very bottom of the screeen, called the *minibuffer*.

If you type a "normal" key (such as a letter, punctuation, or a number, it gets inserted at the

position of the cursor in the edit area. To get Emacs to do something interesting, you can press various key combinations. We use the following notation: `C-x` means pressing the CONTROL key and pressing the key `x` while still pressing CONTROL; the notation `M-x` means pressing the ESCAPE key (sometimes, the ALT key, but that doesn't always work depending on your system), and the pressing the `x` key.

To move the cursor around, the following keys are used:

- `C-f` (forward) to move the cursor right by one character
- `C-b` (back) to move the cursor left by one character
- `C-p` (previous) to move the cursor one line up
- `C-n` (next) to move the cursor one line down
- `C-v` to move the cursor one page down
- `M-v` to move the cursor one page up

(The arrow keys also often cause cursor movement, but again, it may depend on your system.) Other simple key combinations are quite useful:

- `C-l` refreshes the display, and brings whatever text is at the cursor in the middle of the screen
- `C-s` initiates a search for a piece of text, from the current cursor position The minibuffer will ask you to enter the text to search for.
- `C-r` initiates a search for a piece of text, backwards from the current cursor position. Again, the minibuffer will ask you to enter the text to search for.
- `C-g` interrupts the current operation (for example, if you press `C-s` but decide not to search for anything, you get out of the minibuffer by pressing `C-g`)

Other key combinations are double key combinations. You first press a *key prefix*, such as `C-x`, and Emacs will wait for you to press another key to actually execute the corresponding command. If you change your mind after pressing `C-x`, you can press `C-g` to get out. The following combinations are useful:

- `C-x C-s` saves the file to disk, under the same name it was loaded
- `C-x C-w` saves the file to disk, under a different name (it asks for the new path and name in the minibuffer)
- `C-x C-c` quits Emacs. If there are any unsaved changes, it will asks you whether you want them saved.

Emacs buffers

You can edit more than one file at a time. this introduces the notion of a *buffer*. A buffer is typically associated with a file you are editing. (There are other ways to get buffers, however.)

Once Emacs is started, you can edit a new file by doing:

- `C-x C-f` loads a file from disk (or creates it if it doesn't exist)

This creates a new buffer to edit that file. The previous file you were editing is still in Emacs; it has simply been hidden. Every buffer has a name. When the buffer is associated to a file, the name of the buffer is just the name of the file. To change to another buffer (perhaps to continue editing one of your previous files), the following keys are useful:

- `C-x b` changes to another buffer. Emacs will ask you in the minibuffer for the name of the new buffer to go to, or just ENTER if you accept the default choice.
- `C-x C-b` creates a new buffer containing the list of all the buffers Emacs currently has.
- `C-x o` when more than one buffer is on the screen, this key cycles the cursor through all the buffers appearing, so you can edit different buffers on the screen.
- `C-x 1` when more than one buffer is on the screen, this makes the buffer where the cursor into a full screen buffer.
- `C-x k` kills a buffer. If the buffer contains unsaved data, Emacs asks you if you want to save the content.

Functions

Everything in Emacs is ultimately done by commands, also known as functions. At startup, every key is bound to a function. For example, the key `C-f` is bound to the function `forward-char`, while the key `C-x C-w` is bound to the function `write-file`. There are many more functions than key combinations, and it is possible to change the function to which any given key is bound.

It is possible to invoke a command by name:

- `M-x` asks for a function to call in the minibuffer.

Since this is so common, we use the notation `M-x function-name` to represent the call of a function `function-name` from Emacs. Note that Emacs has function name completion: if you start a function name and press TAB, Emacs will try to complete the function name, or give you a list of possible completions in a buffer.

Some useful functions to know about and to explore are:

- `M-x describe-key` asks for a key (by pressing it), and Emacs creates a new buffer containing information on that key, including the function it is bound to, if any.
- `M-x describe-function` asks for a function name, and creates a new buffer containing information on that function.
- `M-x apropos` asks for text to search for in all the function descriptions, and returns the function name for which there is a match.
- `M-x help-with-tutorial` an extremely useful tutorial on the use of Emacs that I strongly advise you to go through at least once.