Emacs editing environment, Part 1: Learn the basics of Emacs

Get going with this famous open source editor

Skill Level: Introductory

Michael Stutz (stutz@dsl.org)
Author
Consultant

20 Mar 2007

Master the Emacs editor and delve into the depths of its most advanced editing commands that have made it famous. The open source Emacs editor (one of the powerhouses of UNIX® computing) is a large, complex application that does everything from editing text to functioning as a complete development environment. It's rich in features and is unlike any other program you're likely to have encountered, especially in the way you specify and input commands. This tutorial, the first in a series, gets you going by providing a concise, hands-on introduction to the most important Emacs editing concepts and features.

Section 1. Before you start

Learn what to expect from this tutorial, and how to get the most out of it.

About this series

The Emacs editing environment is a favorite of UNIX® developers. It's known around the world as the king of editors, but many users find it has a bit of a learning curve. The Emacs environment doesn't seem intuitive at first glance, and it doesn't work like other editors and word processors. But learning Emacs doesn't have to be difficult. Once you get going, you'll see how intuitive it is and become more
comfortable with it after each use. This tutorial series shows you the way, taking you from the basics of Emacs, such as its features, philosophy, key-command layout, and methods for editing text, through many of its powerful editing features.

After completing this series, you'll be able to comfortably use Emacs for everyday editing, be well on your way to Emacs proficiency, and have a good feel for many of the advanced capabilities of Emacs.

About this tutorial

The first part of this tutorial focuses on the history and origin of Emacs and the later part explains how to:

- Start and stop Emacs
- Manipulate files and read them in for editing
- Use basic editing keys
- Move through documents
- Use the powerful Emacs text-marking routines
- Use the mouse for editing and selection

Objectives

The primary objective of this tutorial is to introduce you to the Emacs editor, give you a concise introduction to the application and its design philosophy, and show you how to be productive in this editing environment.

Upon completion of this tutorial, you'll have learned everything you need to know to perform basic text editing with Emacs.

Prerequisites

Previous knowledge of Emacs is not required to benefit from this tutorial; however, you should have a basic understanding of what text editors and word processors do. Although this tutorial is written for all levels of UNIX expertise, it's helpful if you have an understanding of the UNIX filesystem, including:

- Files
System requirements

This tutorial requires a user account on any UNIX-based system that has a recent copy of Emacs installed.

There are several varieties of Emacs; the original and most popular is GNU Emacs, which is published online by the GNU Project (see Resources).

You should have a recent copy of GNU Emacs -- one that is at version 20 or greater. Versions 20 and 21 are the most commonly available, and development snapshots of version 22 are also available. This tutorial works with any of these versions for Emacs. If your system is running something older, it's time to upgrade.

To know what version of Emacs you have running, use the GNU-style --version flag, as follows:

```
$ emacs --version
GNU Emacs 22.0.91.1
Copyright (C) 2006 Free Software Foundation, Inc.
GNU Emacs comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of Emacs
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
$ 
```

Section 2. Introducing the Emacs editing environment

The first thing you should know is that the two most popular editors in UNIX are Emacs and vi, and both of them are over 30 years old. That hardly makes them obsolete tools, however. As with UNIX itself, the basic design is decades old, but today these editors are universal and used in cutting-edge development.

Emacs is one of the first open source and free software projects in history. Its inventor, Richard Stallman, is the founder of the GNU Project and its parent
organization, the Free Software Foundation (see Resources). Even before he published the GNU Public License (GNU GPL) -- the license terms under which the majority of today's open source software is released -- Stallman published his Emacs source code under a similar free copyleft software license called the EMACS Public License.

The name Emacs has come to be a word of its own, but it initially was an acronym that stood for Editing MACros; Stallman's original implementation was a set of macros in the TECO language. Emacs is now written in Emacs Lisp, an elegant high-level programming language.

The GNU Project's short description of Emacs is the extensible self-documenting text editor. Emacs is extensible, meaning that its functionality can be added to and built upon. This is possible because it's written in Lisp, and you can write new Emacs Lisp routines to add new functions. They can run even while an Emacs session is still running. It's self-documenting because instant help is available for every keypress, even in the middle of typing a command. In that case, clicking the Help button pulls up a list of possibilities.

Emacs is said to be an editing environment because it's much more than just an editor in the usual sense of plain-text editing. Many administrators and developers on all platforms use it to compile and debug programs, manage e-mail, manipulate files on the system, run shell commands, and many other things. People even use it to communicate on Usenet news and to browse the Web. Extensions and built-ins handle everything from Internet Relay Chat (IRC) and messaging to network communications. An old UNIX joke goes something like, "The Emacs environment wouldn't be so bad, if only it had a good editor."

Because there’s a definite Emacs way to do things, it has its own vocabulary, which you’ll learn in this section as you step through an anatomy of a typical Emacs window. This section also covers how to start and stop Emacs and how to type various commands.

Starting Emacs

To start Emacs from a the shell, type:

```
emacs
```

When you start Emacs in a console or terminal, you should see it open to fill the entire terminal screen, as shown in Figure 1.

**Figure 1. Emacs in a terminal window**
If you're in X, Emacs generally opens in a window of its own, as shown in Figure 2. You can also specify that it open in a terminal window (like Figure 1) by giving the `-nw` option. At first glance, these two views of Emacs might seem like different applications, but the differences between the two are superficial. The only real differences are the default colors, the graphical logo that the X client displays, and the set of graphical icons near the top of the X client that contain shortcuts for some of the most common commands. The functionality of the text and X versions is exactly the same.

**Figure 2. Emacs, the X client**
Anatomy of an Emacs window

Some parts of the screen need explaining right away. Let's start from the top and work down.

The menu bar

At the top of the Emacs screen is a highlighted bar with some words on it. This is called the menu bar, and it lets you select common Emacs commands from menus. You can access the menus with the keyboard; with the X client, you can also pull down the menus with the mouse.

Experts usually configure their Emacs so that the menu bar is turned off, which gives a little more room on the screen for editing. But when you're learning Emacs, the menu bar is a nice way to familiarize yourself with its many features.

Additionally, if you opened Emacs in X, you should see a special set of icons at the top (see Figure 2); they're shortcuts to the most popular menu options.
To access the menu bar with the keyboard, press **F10**. Try it now and see how in the terminal a new window opens to show the list of menus, as in Figure 3.

### Figure 3. Selecting from the Emacs menu bar

You can use the up and down arrow keys to move between the menu choices and then press **Enter**. If there's a submenu for your selection, it's displayed on the screen. You can select an option from the new submenu the same way, until you pick a command to run.

If you want to abort the procedure at any time, press **Ctrl-G**. This special keystroke makes a beep and quits whatever command is currently being executed. If no command is executing, pressing this keystroke only emits a beep. Try it now.

You should see the Emacs window revert to how it was before you selected the menu.

**The window**

The large main area in the center of the screen is called the Emacs *window*, and it's
where you do your editing. When you open a file for editing, this is where its contents are displayed. When the contents of a file or document appear in an Emacs window, it's called a buffer. You can have multiple buffers open in Emacs at any one time, even if they're not displayed in the main window. During an editing session, you typically have many buffers open, even if only one buffer is displayed in the window.

In the X client, a scroll bar appears to the left of the window. (The terminal version has one as well, but it displays only when a buffer is open.) The scroll bar shows where the text in the Emacs window is in relation to the rest of the buffer, and it also shows the size of this text in relation to the rest of the buffer.

**The mode line**

The highlighted bar that run across the bottom of every Emacs window is called the *mode line*, and you can think of it as the status bar. It gives you important information about your Emacs session and about the current buffer displayed in the window above it, including whether your latest changes were saved to disk, what line number the cursor is at, how far into the buffer the bottom of the screen is (in percentage of the whole), and what Emacs features and settings are currently active.

**The minibuffer**

The small space beneath the mode line and the bottom of the screen or X client window is a single line called the *minibuffer*. This is where Emacs displays messages that are relevant to any operations. When Emacs asks you to input anything, such as a file name, it happens here.

Like the UNIX shell, the minibuffer prompt uses tab completion. Press the Tab key to get a list of possibilities.

**Learn how to type Emacs key bindings**

A particular Emacs key combination to call a given command is called a key *binding*. These can all be customized, but Emacs comes with default bindings.

At first glance, the large amount of default Emacs key bindings might seem confusing and complex, but remember that they're designed for speed and ease of memory. There's usually a mnemonic reason for every binding, such as using the S key for save. Think about these facts as you try them.

Broadly speaking, there are two major classifications of Emacs commands: those
prefaced by the Ctrl key and those prefaced by what's called the Meta key.

**Learn to type Ctrl key combinations**

Many Emacs commands are specified by a Ctrl key combination. In Emacs notation, Ctrl keys are written as "C-" followed by the character corresponding to the second key to press. For example, the popular Ctrl-X combination is written C-x in Emacs notation.

To enter a Ctrl key combination, press and hold the **Ctrl** key, press the **second** key, and then let go of both keys. Most commands have a Ctrl key combination followed by either a word or a second Ctrl key combination.

For instance, try running the C-x C-s command, which saves the current buffer to disk. Because you haven't made any changes, you don't need to save anything, but it's a good key combination to try. Try it now:

1. Press and hold the **Ctrl** key.
2. Press the **X** key once and let go of both keys.
3. Press and hold the **Ctrl** key again.
4. Press the **S** key and let go of both keys.

This keystroke runs the `save-buffer` command. In the minibuffer, Emacs reports, "(No changes need to be saved)."

Emacs users often keep the **Ctrl** key pressed in the second step when running these combinations -- this omits the third step altogether and makes it even faster to type. Try it.

**Learn to type Meta key combinations**

The second major type of Emacs keystroke is the Meta key; in Emacs notation, the Meta key is denoted by M-. If you've never heard of a Meta key, that's because most systems today don't have one. There are three ways to input Meta key combinations:

1. The Meta key is often bound to the Alt key, which is used like the Ctrl key. If your setup is like that, use it -- it's the easiest and most common method.
2. You can usually use the Esc key to give a Meta sequence, but you do it differently than when you enter a Ctrl sequence. Press Esc, let go, and then press the second key.

3. You can use Ctrl-[ as a substitute for the Esc key. This is handy if you’re running an Emacs session over a network line where both Esc and Alt don’t work.

Try typing the M-b command, which moves the cursor back one word, in three different ways:

1. Press and hold the Alt key, and then press B once.
2. Press Esc, let go, and then press B once.
3. Press and hold Ctrl, press [, let go of both keys, and then press B once.

Every Emacs command you run is a function, defined in Emacs Lisp, and has a function name. Even the command to move the cursor to the previous line with the up arrow key is a function (previous-line). You can run any function by using the M-x command followed by the name of the function.

Try it:

1. Press and hold the Alt key.
2. Press the X key.
3. Let go of both keys and notice how "M-x" appears in the minibuffer.
4. Type previous-line and press Enter.

When you do this, the cursor moves up another line. Of course, you normally wouldn’t run this function that way, because using the up arrow key is so much easier, but it’s a good example. There are many, many functions and, because Emacs is extensible, you can write your own to extend its capabilities.

Like the up arrow key shortcut for previous-line, many functions are given keyboard shortcuts. When you type such a key or key combination, that function runs. You can also run the previous-line function using C-p. Try it now:

1. Press and hold Ctrl.
2. Press the P key.

3. Try the arrow key, too, and bring the cursor to the top of the window.

To recap, Table 1 lists the main types of key prefixes you can use in Emacs key bindings. Like anything else in Emacs, these can be customized and redefined.

**Table 1. Common default Emacs key prefixes**

<table>
<thead>
<tr>
<th>Key prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-c</td>
<td>Commands particular to the current editing mode</td>
</tr>
<tr>
<td>C-x</td>
<td>Commands for files and buffers</td>
</tr>
<tr>
<td>C-h</td>
<td>Help commands</td>
</tr>
<tr>
<td>M-x</td>
<td>Literal function name</td>
</tr>
</tbody>
</table>

**Stopping Emacs**

To exit Emacs, type:

\[ \text{C-x C-c} \]

If there are any unsaved buffers, this command gives you a chance to save them first.

This is the usual way to exit Emacs. Try it now, and then start Emacs up again by typing `emacs` at the shell prompt.

---

**Section 3. Buffers and files**

In this section, examine the most important Emacs buffer and file commands: how to load files into buffers, how to save buffers to files, how to switch between buffers, and how to "kill" them.

**Make a new file from scratch**

When you start Emacs in the usual way, it opens to a special buffer called the
scratch buffer, whose purpose is described in a message at the top of the buffer:

```lisp
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f, and then enter the text in that file's own buffer.
```

Toward the left side of the mode line is a place where the name of the current buffer is always displayed. Here, you see the name of this buffer, which is *scratch*. The special buffers that Emacs makes automatically have asterisks on either end of their names.

Try typing a line of text: This is a practice file. and then press Enter to move the cursor down to a new line. As soon as you begin to type, you should see two asterisks appear at the left side of the mode line -- that means the current buffer has text in it that hasn't been saved to disk.

One way to make a new file is to write the scratch buffer to a file. The key combination C-x C-s runs the save-buffer command -- it saves the contents of the current buffer to a file. As you saw before, when you run it in a buffer that doesn't need saving, Emacs tells you as much. But when you run it in a buffer that has unsaved changes and the buffer corresponds to a particular file on disk, the contents of the buffer are written to that file. When you run it in a new buffer or in a buffer that isn't yet associated with a file (like the scratch buffer you're in now), Emacs prompts for the name of file you'd like to save the buffer in.

Try it now: Type Ctrl-x Ctrl-s. See how in the minibuffer asks you to give the name of the file to save in, as in Figure 4. Now type practice.

Figure 4. Saving the scratch buffer
After you press **Enter**, Emacs reports in the minibuffer that a new file named practice has been written to disk. Exit Emacs by typing `C-x C-c`, and then look in your directory to see your file there:

```
$ ls
practice
$
```

There's one more thing to remember about the scratch buffer. When no other buffers are open in Emacs, there's always a scratch buffer -- and when it's the only buffer open, it can't be closed.

**Starting Emacs with filenames**

To start Emacs with the contents of a given file in a new buffer for editing, give the name of the file as an argument. If you give multiple files, each file is opened in a buffer of its own. If any of the file names you give don't exist on disk, Emacs makes a new buffer for that file and indicates (on the mode line) that it's a new file. This is
what you do when you start Emacs and edit a new file. When you save the buffer to disk, Emacs writes to the file whose name you gave as the argument.

Try starting Emacs with the name of your practice file:

```
$ emacs practice
```

You still see the Emacs welcome screen, but if you press a key like c-g (or wait long enough), you see your file in the Emacs window. Yes, the welcome screen is another configurable option.

Open a file

Emacs never works directly on the contents of files. It reads a copy of the file's contents into a buffer that you edit. Use C-x C-f, the find-file command, to open a buffer with the contents of a file.

Try opening your file with this command now:

1. Type C-x C-c to exit Emacs.
2. Start Emacs again, but this time don't specify your file:

```
$ emacs
```

3. Type C-x C-f and give the name of your file (practice) when you're asked to do so in the minibuffer.

Visit a buffer

The C-x b command switches from the current buffer to another buffer that you specify. A default option is always given in the minibuffer. If you press Enter without giving a buffer name, that's the buffer you switch to. The default is usually the last buffer you visited. If you haven't visited a previous buffer, the default is usually the scratch buffer.

Type C-x b now and notice how the scratch buffer is suggested in the minibuffer.
Press **Enter** -- your practice file is gone and the familiar contents of the scratch buffer are in the window. (By default, the scratch buffer contains that three-line message, but sometimes the scratch buffer is empty.) Type `C-x b` again, and press **Enter** to switch back to the practice buffer.

If you want to make a new buffer when you're in Emacs, switch to a buffer with the name you want to give it.

Type `C-x b` again, but give the name mybuffer for the buffer. Notice how the window is empty -- this is a brand new buffer. Type a line of text, and press **Enter**:

```
On what wings dare we aspire?
```

Save this buffer to disk by typing `C-xC-s`. Notice how now Emacs asks you for a name of the file. When you make a new buffer that doesn't correspond to a file on disk, you have to also give it a file name if and when you decide to save it. You don't have to give it the same name as the buffer -- call it practice.b and notice how Emacs changes the name of the buffer to correspond with the new file.

**Kill a buffer**

Use the `C-x C-k` command to *kill* a buffer or close it out of your Emacs session. You're prompted to type the name of the buffer to kill; the current buffer is the default and is killed if you press **Enter**.

Try killing your old practice buffer now: Type `C-x C-k` and, when Emacs asks for a buffer name in the minibuffer, type `practice`, and press **Enter**. The practice buffer is killed and the practice.b buffer remains in the window as before.

**Save buffers to disk**

You already know how to save the contents of a buffer to disk by running the `C-x C-s` save-buffer command. You need to know a few more things about this process.

The first few characters in the mode line describe the status of the buffer. Type another line of text in your practice.b buffer now and press **Enter**:

```
What the hand dare seize the fire?
```
Notice how two dashes in the mode line have turned into two asterisks. The straight dashes meant that the contents of the buffer were the same as what was on disk, and the asterisks indicate that the buffer has unsaved edits.

Save your changes and exit Emacs: Type `C-x C-s C-x C-c`. You should have several new files in your directory:

```
$ ls
  practice
  practice.b
  practice.b~
```

The practice and practice.b files are the ones you created, but practice.b~ is a file automatically created by Emacs. This is an Emacs backup file, which is created any time you edit a preexisting file in Emacs. When you save your edits to the file, Emacs makes a backup copy by writing to a new file whose name is the same as the original but with a tilde appended to it. The old practice file doesn't have a backup, because you never edited it after you created it. If you ever edit it again in Emacs, Emacs writes a corresponding practice~ backup file.

Emacs also writes another kind of file, called an autosave file, which has the same name as your file but with a pound sign before and after the name. Emacs writes an autosave file at set intervals for as long as you're working on a buffer. The default is to write an autosave after every 300 new characters. You normally never see the autosave file when you kill the buffer. This happens because the autosave file corresponding to that buffer is deleted. Autosave files are handy in case your system crashes or you lose the connection in the middle of an editing session -- this gives you the option of recovering your lost edits by looking for such a file in that directory.

Summary of buffer and file commands

Table 2 contains a list of some of the most common buffer and file commands that you'll use as you learn Emacs. The table gives both the keystroke the command is bound to and the function name of the command. Remember that you can always run a command by using its key binding as well as by giving its function name as an argument to `M-x` (see Learn to type Meta key combinations).

<table>
<thead>
<tr>
<th>Binding</th>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-x C-s</td>
<td>save-buffer</td>
<td>Save current buffer to disk.</td>
</tr>
<tr>
<td>C-x s</td>
<td>save-some-buffers</td>
<td>Ask about saving all unsaved buffers to disk.</td>
</tr>
</tbody>
</table>
Section 4. Edit text in Emacs

Entering and changing text in a buffer as well as navigating through that text are the most important things you do in Emacs, whether you're editing an existing file, creating a new file, or just perusing a file.

In this section, learn the basic key sequences and commands for doing these things: how to enter text in a buffer, how to navigate through the text, and how to do basic edits on that text, such as deleting characters and words.

Type in and move through a buffer

As you've already seen in the Make a new file from scratch section, entering text in an Emacs buffer is easy -- you just start typing. You can enter any alphanumeric character in a buffer by typing that character.

Emacs is sometimes called a modeless editor, often in comparison with a modal editor such as vi. This means the behavior of the editor and the keystrokes and commands you can type remain constant in your session, as opposed to in an editor like vi, where keystrokes have certain meanings depending on whether you're in command mode or insert mode. Emacs has no such modes; however, it does have a different kind of mode that can change its behavior or extend its capability -- but that's a topic for the next tutorial in this series.

As with plain typing, you need to keep a few things in mind.
**Insert text at point**

In Emacs, an important concept called *point* denotes where character insertion occurs. It's an imaginary spot in the buffer that's just between the character where the cursor is located and the character immediately preceding it.

Whenever you type text in a buffer, it's inserted at point. By default, all the text on the same line but after point is pushed over to the right to make room for what you insert. Press `Enter` to move to the next line and press `Enter` again to insert a blank line.

Try inserting a paragraph of text at the beginning of your practice.b buffer:

1. Start Emacs with that buffer:

   ```
   $ emacs practice.b
   ```

2. Type a paragraph by pressing `Enter` at the end of each line, and press `Enter` again at the end to make a blank space after it:

   ```
   Tyger! Tyger! burning bright  
   In the forests of the night,  
   What immortal hand or eye  
   Could frame thy fearful symmetry?
   ```

**Note:** This is the first stanza from a poem called *The Tyger!* by William Blake. The author references several stanzas of this poem throughout the tutorial. Now your practice.b buffer should look like Figure 5.

**Figure 5. Inserting a paragraph**
Move point

You see how the cursor follows your typing and how point remains the same:
Everything you type is inserted before the letter O from the first sentence you typed.

To move point, you can use the arrow keys and, as you'd expect, all the other cursor motion keys work, such as the PgUp, PgDn, Home, and End. But Emacs has its own bindings for moving the cursor and, because you don't have to move your hands away from the home row of the keyboard to use them, you'll find that they're much more useful when you're typing.

As you saw in the Learn to type Meta key combinations section, you can use C-P to move the cursor up to the previous line; similarly, C-n moves the cursor down to the next line. C-f moves forward to the next character and C-b moves backward to the previous character.

To move up and down by a screenful without using the PgDn and PgUp keys, use the C-v and M-v keystrokes, which do the same thing. To go to the beginning of the current line, use C-a; and use C-e to go to the end of the current line.
Usually, a Meta key is bound to a command that's similar to its corresponding Ctrl key and, for movement, this general rule applies. When used with Meta instead of Ctrl, the F and B keys move forward and backward by the *word* instead of by the character, whereas the A and E keys move to the beginning and ending of the current *sentence*. (In the default configuration, the $M-n$ and $M-p$ keystrokes are not defined.)

Table 3 lists the various movement and navigation keys, giving their function names and descriptions. Try using them to move to the beginning of the buffer, end of the buffer, and places in between.

### Table 3. Useful Emacs keystrokes for movement and navigation

<table>
<thead>
<tr>
<th>Keystroke</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-p, UpArrow</td>
<td>previous-line</td>
<td>Move point up to the previous line.</td>
</tr>
<tr>
<td>C-n, DownArrow</td>
<td>next-line</td>
<td>Move point down to the next line.</td>
</tr>
<tr>
<td>C-f, RightArrow</td>
<td>forward-char</td>
<td>Move point forward to the next character.</td>
</tr>
<tr>
<td>C-b, LeftArrow</td>
<td>back-char</td>
<td>Move point backward to the previous character.</td>
</tr>
<tr>
<td>M-f</td>
<td>forward-word</td>
<td>Move point forward to the next word.</td>
</tr>
<tr>
<td>M-b</td>
<td>backward-word</td>
<td>Move point backward to the previous word.</td>
</tr>
<tr>
<td>C-v, PgDn</td>
<td>scroll-up</td>
<td>Scroll the text upward by a screen.</td>
</tr>
<tr>
<td>M-v, PgUp</td>
<td>scroll-down</td>
<td>Scroll the text downward by a screen.</td>
</tr>
<tr>
<td>Home</td>
<td>beginning-of-buffer</td>
<td>Move point to the beginning of the buffer. (On some versions, this key is defined by default to move to the beginning of the current line.)</td>
</tr>
<tr>
<td>End</td>
<td>end-of-buffer</td>
<td>Move point to the end of the buffer. (On some versions, this key is defined by default to move to the end of the current line.)</td>
</tr>
<tr>
<td>C-a</td>
<td>beginning-of-line</td>
<td>Move point to the beginning of the line.</td>
</tr>
<tr>
<td>C-e</td>
<td>end-of-line</td>
<td>Move point to the end of the line.</td>
</tr>
<tr>
<td>M-a</td>
<td>beginning-of-sentence</td>
<td>Move point to the beginning of the sentence.</td>
</tr>
<tr>
<td>M-e</td>
<td>end-of-sentence</td>
<td>Move point to the end of the sentence.</td>
</tr>
<tr>
<td>C-{</td>
<td>beginning-of-paragraph</td>
<td>Move point to the beginning of the paragraph.</td>
</tr>
</tbody>
</table>
Use overwrite mode

As you’ve seen, the text you type in a buffer is inserted at point. But you can overwrite existing text, too. Press the **Ins** key once; this toggles overwrite mode, which is off by default. Look at the mode line -- you should see **Ovwr** to tell you that this mode is now active.

Use the commands for movement as described in **Table 3**, and move the cursor so that it's on the *w* in *we*, and then type an *h* character.

Try going to the top of the buffer: Type **M-a M-a C-f** to move the cursor above the *y*, and then type an *i* character. Do the same for the next *y* by typing **M-f C-f C-f C-f i** so that your buffer looks like **Figure 6**.

**Figure 6. Overwriting characters in a buffer**

Press **Ins** again to turn overwrite mode off.
Quoted insert

You're not limited to the alphanumeric keys for entering text. You can enter literal control characters, and you can enter characters based on their character code. Do this by performing a quoted insert, which is bound to \texttt{C-q}; follow it with a literal keypress, such as a Ctrl combination, to enter that key at point. You can also give a character code value, in octal, followed by the Enter key.

Move to the end of the buffer, type \texttt{Page break here}, and press \texttt{Enter}. Now type a page break, which is a literal Ctrl-I character, followed by an escape character, which has an octal value of 033: \texttt{C-q C-l C-q 033 Enter}.

Delete, undo, redo

It's time to try the Emacs facilities for getting rid of existing text and for undoing (and redoing) what you've already done.

Delete text

Use either the Backspace or Del key to delete the character to the left of point. Try using both to delete the two control characters you just entered.

To delete the character at point, use \texttt{C-d}; similarly, \texttt{M-d} deletes from point forward to the end of the word. You can go backward, too -- both \texttt{M-Del} and \texttt{M-Backspace} delete everything from the left of point to the beginning of the word.

Try using these commands to delete the sentence \textit{Page break here} that you just typed, and then delete the blank line the words were typed on.

Move the cursor to the last line in the file (which should be the sentence "What the hand dare seize the fire?"), and delete it with several taps of \texttt{M-d}.

Undo and redo

Oops, what if you didn't mean to delete that last sentence? You can get it back by running the undo function. This is bound to \texttt{C-_}, which you type by holding Ctrl and using the Shift key to type an underscore. Try it now to get back the last line, tapping it once for each word to come back.

Tap it a few more times so that \textit{Page break here} comes back.

Now you decide that you regret this last undo -- you really do want \textit{Page break here}
to go away. You can redo your undo by typing C-g, which cancels any more undoing, and then type C-_ enough times so that the words are gone again.

Run a command multiple times

C-u is the universal argument command, which you follow with a number and then a command. It runs the given command that number of times.

Try it: Move to the beginning of the second verse, type In what distant d, and then type C-u 2 e -- two e characters should be written. Finish the verse by typing:

ps or skies
Built the fire of thine eyes?

Type C-x C-s to save your buffer; it should now look like Figure 7.

Figure 7. Character insertion with the universal argument buffer
With no numeric argument, \texttt{C-u} assumes a value of 4. If you give \texttt{C-u} itself as the argument, it multiplies that default 4 by itself to make 16; you can build on this as much as you like before you give a command -- for example, \texttt{C-u C-u C-u A} types the letter \texttt{A} 64 times at point.

Try moving backward with \texttt{C-b} once, and see how the cursor wraps back to the line before it. Now try it with the universal argument by typing \texttt{C-u C-b}; notice that the cursor moves back four characters instead of one. Try it again with \texttt{C-u C-u C-b}, and see how the cursor moves 16 characters to the left. Type \texttt{C-u C-u C-u C-b}, and then give it an argument of 1000: \texttt{C-u 1000 C-b}. The cursor moves to the beginning of the buffer, but notice that Emacs beeps -- that means it reached the top of the buffer (and could move back no further) before it went through all the requested iterations.

**Table of editing keys**

Let's recap what you've just learned. Before you advance to the next section, take a glance at Table 4, which lists important editing commands and the default keystrokes to use them.

**Table 4. Common Emacs editing commands**

<table>
<thead>
<tr>
<th>Keystroke</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{Ins}</td>
<td>overwrite-mode</td>
<td>Toggle overwrite mode (default is off).</td>
</tr>
<tr>
<td>\texttt{Backspace}, \texttt{Del}</td>
<td>delete-backward-char</td>
<td>Delete the character before point.</td>
</tr>
<tr>
<td>\texttt{C-d}</td>
<td>delete-char</td>
<td>Delete the character at point.</td>
</tr>
<tr>
<td>\texttt{M-d}</td>
<td>kill-word</td>
<td>Delete the characters from point forward to the end of the word.</td>
</tr>
<tr>
<td>\texttt{M-Backspace}, \texttt{M-Del}</td>
<td>backward-kill-word</td>
<td>Delete the characters from point backward to the beginning of the word.</td>
</tr>
<tr>
<td>\texttt{C-_}</td>
<td>undo</td>
<td>Undo your last typing or action.</td>
</tr>
<tr>
<td>\texttt{C-q} \texttt{character} or \texttt{XXX}</td>
<td>quoted-insert</td>
<td>Insert, at point, the literal \texttt{character} keypress or the character whose octal value is \texttt{XXX}.</td>
</tr>
<tr>
<td>\texttt{C-u} \texttt{command}</td>
<td>universal-argument</td>
<td>Execute \texttt{command} a total of \texttt{number} (default 4) times in succession.</td>
</tr>
</tbody>
</table>
Section 5. The mark and the mouse

You're well on your way to grasping all the basics of Emacs editing, but there are a few more important concepts to learn: how to mark sections of text and perform operations on them and how to use the mouse.

Mark, yank, kill

Emacs has a facility for marking an area of text so that you can edit it as a whole. These commands are described here and outlined in Table 5.

Mark a region

Move to the top of the buffer -- the beginning of the first stanza -- and type \texttt{C-Space} by pressing and holding Ctrl and then pressing the spacebar. This is called \textit{setting the mark}; a message in the minibuffer tells you that the mark has been set.

Everything between point and the place where you've set the mark is called the \textit{region}.

Make the whole first stanza the region by moving point down to the blank line right after the stanza.

Kill and yank text

There are special commands that work on the region, including \texttt{C-w}, which kills it.

Type \texttt{C-w} to kill the region you just defined.

Whenever you kill text, it's saved in the Emacs \textit{kill ring}. You can get it back by yanking it at point with \texttt{C-y}. Move to the end of the buffer, press \texttt{Enter} to insert another blank line, and then yank the stanza back.

You're not limited to killing regions; use \texttt{C-k} to kill all the text from point to the end of the line. Move up to the line beginning with \textit{Tiger}, and kill it with a \texttt{C-k}. Notice that doing so doesn't take out the blank line; that requires a second \texttt{C-k}. Do it, and then use \texttt{C-y} to yank the whole line back. If you kill with multiple kill commands typed in succession, they're added together, and a single yank returns them.

You can yank a line as many times as you want. Move to the top of the buffer, and yank the line back again with another \texttt{C-y}. 
You can also kill multiple lines with successive \texttt{C-k} keystrokes, and they're yanked together. Try killing several lines at once, and then yank them right back as they were with a single \texttt{C-y}.

\textbf{Copy text}

You don't have to kill the region in order to copy it. To save the region in the kill ring without killing it, use \texttt{M-w} instead of \texttt{C-w}.

Try it:

1. Move point to the beginning of the second line of the bottom stanza.
2. Type \texttt{C-Space} to set the mark.
3. Move point to the line beneath the stanza.
4. Type \texttt{M-w} to copy these three lines in the kill ring without killing them.
5. Move point up to the blank line beneath the first line in the buffer.
6. Yank the three lines with \texttt{C-y}.

After this, your buffer should look like \textbf{Figure 8}.

\textbf{Figure 8. Yanking multiple lines}
Table 5. Emacs functions for marking and killing text

<table>
<thead>
<tr>
<th>Keystroke</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-Space</td>
<td>set-mark-command</td>
<td>Set the mark at point.</td>
</tr>
<tr>
<td>C-k</td>
<td>kill-line</td>
<td>Kill all text from point to the end of the line.</td>
</tr>
<tr>
<td>C-w</td>
<td>kill-region</td>
<td>Kill the region.</td>
</tr>
<tr>
<td>M-w</td>
<td>kill-ring-save</td>
<td>Save the region in the kill ring, but don’t kill it.</td>
</tr>
<tr>
<td>C-y</td>
<td>yank</td>
<td>Yank text from the kill ring.</td>
</tr>
</tbody>
</table>

Mark and move with the mouse

Although Emacs is designed for fast keyboard use, you can use the mouse -- and doing so sometimes comes in handy in text operations.

To move point anywhere in the buffer, move the mouse pointer to where you want to
go, and click the **first mouse button** once. Try using the mouse to move to the space between the second and third stanzas, and type a new stanza:

```
When the stars threw down their spears,
And watered heaven with their tears,
Did he smile his work to see?
Did he who made the Tiger make thee?
```

Double-clicking a word with the **first button** selects that word. Double-click the *Tiger* you just typed -- it should become highlighted -- and press **Del** to delete it. Now type *Lamb* to insert that word at point.

To select a whole line with the mouse, triple-click the line with the **first mouse button**. Try it on the top line, and press **Del** to delete it. Type `C-` to bring it back.

Click the **first button** in the middle of the buffer, and type two more paragraphs to complete the poem:

```
What the hammer? what the chain?
In what furnace was thy brain?
What the anvil? what dread grasp
Dare its deadly terrors clasp?

And what shoulder, and what art,
Could twist the sinews of thy heart?
And when thy heart began to beat,
What dread hand? and what dread feet?
```

You can click the **first mouse button** and drag the pointer to select a region. You can also select a region by clicking the **first button** once (which sets point there) and then clicking the **third button** somewhere else (which sets the mark there). When you do that, the text you've selected is placed in the kill ring without killing it -- you can use `C-y` or the middle mouse button to yank the text at point. To place the highlighted region in the kill buffer and kill it, double-click with the **third mouse** to set the region.

Try switching the order of the two stanzas you just typed:

1. Click the **first mouse button** on the beginning of the blank line between them.
2. Double-click the **third mouse button** below the second stanza you just typed.
3. Click the **second mouse button** at the beginning of the blank line before
the stanza that begins with *What the hammer?*

Type `C-x C-s` to save your buffer to disk. Your Emacs session should now look like Figure 9.

**Figure 9. Inserting with the mouse**

Table 6 describes the various Emacs mouse operations.

**Table 6. Mouse operations in Emacs**

<table>
<thead>
<tr>
<th>Mouse command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>This command sets the point; drag B1 to set the region.</td>
</tr>
<tr>
<td>B1–B1</td>
<td>This command marks a word.</td>
</tr>
<tr>
<td>B1–B1–B1</td>
<td>This command marks a line.</td>
</tr>
<tr>
<td>B2</td>
<td>This command yanks the text.</td>
</tr>
<tr>
<td>B3</td>
<td>This command sets and highlights the region, and then places it in the kill buffer without killing it. If a region is already highlighted and set, the</td>
</tr>
<tr>
<td>B3–B3</td>
<td>This command sets and highlights the region and then kills it. If a region is already highlighted and set, the end of the region moves to the place where you click, and then the region is killed.</td>
</tr>
</tbody>
</table>

---

### Section 6. Summary

#### Wrap-up

Congratulations! You've gone through the first tutorial in this series on learning to use Emacs. You've learned all the most important basics of the Emacs editing environment: how to open new buffers and save them to files; how to move through buffers, enter and edit text in a buffer, and mark and operate on regions of text; and even how to use the mouse for text operations.

Although this tutorial covered a lot of ground, you have a lot to learn. Further tutorials in this series take you through the complex editing features of Emacs and show you how to use them in your work.
Resources

Learn

• "Use free software within commercial UNIX" (developerWorks, February 2006): Learn how to install GNU Emacs on the IBM AIX operating system or another commercial UNIX.

• Emacs commands: The IBM DB2® Universal Database Information Center has a quick summary of basic Emacs commands.

• "UNIX tips and tricks for a new user, part 2: The vi text editor" (developerWorks, November 2006): This tutorial provides an introduction to vi.

• GNU Project: Learn more about the GNU Project and the Free Software Foundation from its Web site.

• Check out other articles and tutorials written by Michael Stutz:
  • AIX and UNIX zone
  • Across developerWorks and IBM

• Search the AIX and UNIX library by topic:
  • System administration
  • Application development
  • Performance
  • Porting
  • Security
  • Tips
  • Tools and utilities
  • Java™ technology
  • Linux®
  • Open source

• AIX and UNIX: The AIX and UNIX developerWorks zone provides a wealth of information relating to all aspects of AIX systems administration and expanding your UNIX skills.

• New to AIX and UNIX: Visit the New to AIX and UNIX page to learn more about
AIX and UNIX.

- **AIX 5L Wiki**: A collaborative environment for technical information related to AIX.
- **Safari bookstore**: Visit this e-reference library to find specific technical resources.
- **developerWorks technical events and webcasts**: Stay current with developerWorks technical events and webcasts.
- **Podcasts**: Tune in and catch up with IBM technical experts.

**Get products and technologies**

- **Latest GNU Emacs**: Download a free copy of the latest GNU Emacs for your operating system and architecture free of charge from the GNU Project Web site.
- **IBM trial software**: Build your next development project with software for download directly from developerWorks.

**Discuss**

- Participate in the **developerWorks blogs** and get involved in the developerWorks community.
- Participate in the AIX and UNIX forums:
  - **AIX 5L -- technical forum**
  - **AIX for Developers Forum**
  - **Cluster Systems Management**
  - **IBM Support Assistant**
  - **Performance Tools -- technical**
  - **Virtualization -- technical**
  - **More AIX and UNIX forums**

**About the author**

Michael Stutz

Michael Stutz is author of *The Linux Cookbook*, which he also designed and typeset using only open source software. His research interests include digital publishing and the future of the book. He has used various UNIX operating systems for 20 years. You can reach Michael at stutz@dsl.org.
Trademarks

IBM, AIX, AIX 5L, and DB2 are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.