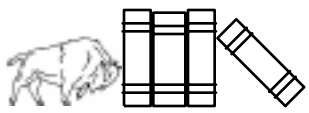
Computing and Information Technology

University at Buffalo
The State University of New York

Insights!

A series on special-interest pursuits



Introduction to Emacs

GEN-INS-003
Academic Services
Computing and Information Technology
Computing Center
University at Buffalo
Buffalo, New York 14260 USA

This document was prepared using *FrameMaker 5.5.6* on SunOS UNIX.

The following fonts are used in this document:

Helvetica Bold 10 points - Typeface used in the headers of each section.

Helvetica Bold 9 points - Typeface used in the menu, dialog box, and command button.

`Courier 10 points` - Typeface used to indicate output on the terminal's screen.

`Courier Oblique 10 points` - Typeface used to indicate user's typed input.

`Courier Bold Oblique 10 points` - Typeface used to indicate commands and is used inline in the document.

Times Roman 10 points - Typeface for the body of the document.

Times Italic 10 points - Typeface to indicate a name or title.

Helvetica Narrow Oblique 8 points - Typeface used for margin notes.

All commands demonstrated in this document should be followed with a carriage return key. Some keys are denoted by name, including ***`<Esc>`*** for the Escape key, ***`<Delete>`*** for the Delete key, ***`<Return>`*** for the Return key, and ***`<Space>`*** for the spacebar. When using these keys, you need press them only once to obtain the desired effect.

Revised: July 2001

Contents

Introduction

This document is designed as an introduction to *GNU Emacs*, a text editor available on UBUnix. To use *Emac*s on UBUnix, you must have a valid *UB IT Name* and password.

What is *GNU Emacs*?

GNU Emacs (or *Emacs* for short) is an advanced, feature-packed, customizable display text editor. Emacs allows you to edit several files simultaneously, open multiple session windows, define keyboard macros, and undo mistakes.

Initializing Emacs

To use Emacs on the UBUnix system, you must first set the terminal emulation. To do this, type the following at your UNIX prompt:

```
setenv term termtype
```

where *termtype* is the desired terminal type. For computers in Public IT Computing Areas, *termtype* should be set to ***vt100***. If you are a frequent Emacs user, you may wish to add the following line to your *.cshrc* file:

```
vt100
```

If you wish to set your default text editor to Emacs, add the following lines to your *.cshrc* file:

```
setenv EDITOR /util/bin/emacs
setenv VISUAL /util/bin/emacs
setenv ERROR_EDITOR /util/bin/emacs
```

Starting Emacs

To invoke Emacs, type the following at your UNIX prompt:

```
emacs
```

Emacs will start up and display the default help message. Type any key to clear the help message. You are now ready to create a file or modify an existing file.

Reading a file

If you issue the ***emacs*** command without specifying a file, Emacs will open up a file called **scratch**. This file cannot be saved, however. To save any work in Emacs, you must either edit an existing file or a new file.

You can also open an existing file from within Emacs by using the ***find-file*** command. This command is invoked by typing ***C-x C-f***. Emacs will display the current directory in the mode line. Provide the name and directory of the file you wish to edit and press ***<Return>***. You are now able to edit the text of the file.

If you wish to edit a different file, whether it is a new or an existing file, use the ***C-x C-f*** command again. The same command allows you to start a new file by entering a new filename. Emacs automatically saves the file that was open when you issued the ***C-x C-f*** command.

Alternately, you can read in a file when it starts up by typing:

```
emacs filename
```

where *filename* is the name of the file you wish to edit. If this file does not currently exist in the current directory, Emacs will create it for you.

It is also possible to start up Emacs at a particular place in a file by using the ***plus*** option. This option consists of a plus sign (+) followed by a number. This combination instructs Emacs to go to the specified line number in the file. The plus option is entered after the command ***emacs***, but before any file names. To start Emacs at line 150 of a file named *fn*, for example, you would type the following at your UNIX prompt:

```
emacs +150 fn
```

Special Keys and Commands

Emacs commands make use of the **<Control>** key and the **<Meta>** key (which is usually labeled **<Esc>**).

The following notation is used in this document:

Command	*Description*
C-<chr>	Hold down the **<Control>** key and press the keyboard character **<chr>**.
M-<chr>	Hold down the **<Meta>** key and press the keyboard character **<chr>**.

If there is no **<Meta>** key on your keyboard - and it is likely there isn't - type **<Esc>**, release it, then type the character **<chr>**. If there is no **<Esc>** key, then type **C-[** (the **<Control>** key, followed by the character **[**) instead.

The *undo* command

You can undo all sequential changes made in the text of a buffer. The commands **C-x u** and **C-_** both peform this function. The first iteration of this command will undo the last change. Repeating the command undoes earlier and earlier changes. Any command other than an **undo** command breaks the sequence of the undo commands.

The *keyboard-quit* command

Any Emacs command may be interrupted or aborted by means of the **keyboard-quit** command. Simply type **C-g** to interrupt the current command.

Repeating commands

It is possible to repeatedly execute a command in Emacs without explicitly entering the command multiple times. To do this, enter the following command from within Emacs:

```
C-u # command-name
```

where *command-name* is the name of the command you wish to repeat, and # is the number of times you would like to repeat the command. If you wanted to delete five characters after the point, for example, you would execute the command **C-u 5 C-d**.

NOTE: The screen-moving commands **C-v** and **M-v** may not be executed in this fashion.

The Emacs Screen

Emacs divides the session screen into several areas. The largest area contains the text you are editing. The cursor appears in the text, showing the *point*. Although the cursor seems to point at a character, the *point* should be thought to exist between two characters. It points to the character immediately to the left of the cursor.

Most terminal software programs have only one cursor, but Emacs may have several points of entry. If you are editing several files, each file has its own cursor location. When multiple text windows are open, each window has its own point location as well. The cursor shows the point in the selected window.

The last line in a text window is the *mode line*, which describes what is going on in that window. If your terminal supports it, this line is displayed in inverse video. The mode line display looks like this:

```
----:---F1  filename     (MajorMode)--Ln--%----
```

where *filename* is the name of the file you are editing, and *MajorMode* is the current editing mode.

The characters to the left of the mode line indicate whether or not you have made changes to the file's contents. A series of dashes indicates no changes have been made, while the presence of any asterisks in the sequence indicates that changes have been made to the file.

The characters to the right of the MajorMode indicator display the line (*Ln*) of the file upon which the cursor rests, while also indicating the depth (*%*) of the file, measured as the percentage of the file that lies above the current cursor position. If the entire file is currently displayed on the screen, Emacs will display `All`; if the current buffer contains the beginning of the file, it will display `Top`; and if if the current buffer contains the tail end of the file, the program will display `Bottom`.

The line at the very bottom of the screen is the *echo area*. It is used to display small amounts of text, including error messages and informative messages that explain the results of a command. It also serves as a minibuffer, allowing Emacs to read arguments for commands you have typed.

Modes

Emacs has both *major* and *minor* modes. The *major modes* are a mutually exclusive set of options, each of which configures Emacs for editing a certain sort of text. A *minor mode* is an optional feature of Emacs that can be switched on or off, independently of other modes in operation.

Major modes

A major mode changes the ways in which Emacs will interpret certain keys. These new interpretations help make Emacs more useful for editing a particular language. There are major modes for programming languages, English text, HTML code, and other formats. Some of the major modes available in Emacs are listed below:

fundamental
lisp-interactive
c-mode
lisp-mode
modula-2-mode
latex-mode
tex-mode
HTML
text-mode

The least specialized major mode is *Fundamental*, which is also the default Emacs mode. It has no variable settings, so each Emacs option is in its default state.

Minor modes

Minor modes operate independent of other modes, and are used to activate some of the more convenient features of Emacs. Some of the minor modes available in Emacs are listed below:

auto-fill-mode
auto-save-mode
overwrite-mode

Selecting modes

To select a mode in Emacs, type the following from within your Emacs session:

```
M-x modename-mode
```

where *modename* is the name of the mode you wish to invoke. To select the major mode *lisp*, for example, type the following from within your Emacs session:

```
M-x lisp-mode
```

You can select a major mode for the current buffer, but Emacs is capable of determining the optimal mode based on the name of the file or some text within the file itself. To turn on a minor mode, simply execute the desired mode command normally. To turn it off, execute the command again.

Saving and Exiting Files

The following commands are used to save your file and exit Emacs:

Command	*Description*
C-x s	Saves file without exiting.
C-x C-c	Exits Emacs. If you have made changes to a file but not saved them, Emacs asks if you want to save them.
C-z	Suspends Emacs.

Note that **C-z** suspends Emacs, but this does not end your session. To return to a suspended session, use the command **fg**.

Inline Help

The command **C-h** provides extensive help within Emacs. You can add options to this command to get specific kinds of help.

One help option is **C-h**, so typing **C-h C-h** gives help about using the **help** command itself. **C-h C-h** displays a list of possible options in the echo area. Emacs then asks you to type the character that corresponds to the type of help you desire. Typing in a third **C-h** displays an explanation of each option. At this point, Emacs will wait for you to enter an option.

There are four basic help functions in Emacs. These functions are:

Command	*Help Function*
C-h t	Tutorial
C-h c	Key documentation
C-h f	Command name documentation
C-h a	String documentation

These options are explained in the following sections. You are encouraged to try out the help options if you want to find out more about Emacs.

To clear a help window, use the command **C-x 1**. You can also exit help at any time by using the **C-g** command.

Tutorial

The command **C-h t** starts the interactive Emacs tutorial. The tutorial is actually a file that can be read just like any other file.

To exit the tutorial, use the command **C-x C-c**, just like you would to exit any other file. Emacs will give you the option to mark your place in the tutorial and save the tutorial in your home directory. You can also print the tutorial for later reference. See the *Printing* section for information on printing a file.

Key documentation

The most basic **C-h** options are **c** and **k**. The **c** option is used to obtain a brief description of a command, while the **k** option is used to obtain full documentation for a command.

To obtain a brief description for a specific key or key combination, use the following command from within Emacs:

```
C-h c key
```

where *key* is any keyboard key or combination of keys that constitutes an Emacs command.

The name of the command represented by *key* will appear in the echo area. For example, the following command:

```
C-h c C-f
```

will display the following in the echo area:

```
C-f runs the command forward-char
```

Because a command name illustrates the purpose of a command, ***C-h c key*** is a good way to get a very brief description of what a particular key or key combination does.

To obtain documented inline help on a specific key or key combination, use the following command from within Emacs:

```
C-h k key
```

where *key* is any keyboard key or combination of keys that constitutes an Emacs command. This command is similar to ***C-h c key***, but gives more information on the command indicated by *key*. Because this information is too large for the echo area, a second display window will appear. This window will contain the name of the command represented by *key*, as well as more detailed information on the command.

Command name documentation

The ***f*** option allows you to obtain information about a command that is not bound to a particular key. Commands that are not bound to a particular key are those that you would normally execute using the command ***M-x command-name***, where *command-name* is the name of the command you wish to execute. To obtain information on such a command, use the following command from within Emacs:

```
C-h f command-name
```

where *command-name* is the name of the command about which you desire more information.

In addition, the following command allows you to discover shortcuts for a command:

```
C-h w command-name
```

This command displays those keys that are bound by the command *command-name*.

String documentation (Apropos)

To obtain a list of commands for a particular topic, use the following command within Emacs:

```
C-h a string
```

where *string* is a topic about which you would like command information. To display a list of commands that contain the string *file*, for instance, use the following command within Emacs:

```
C-h a file
```

This command will display a list of command names that contain the string *file*, including ***copy-file*** and ***find-file***.

The ***a*** in ***C-h a*** stands for *apropos*. ***C-h a*** runs the *Lisp* function ***command-apropos***. In addition to each command name, Emacs will list a brief description of each command, including the keystroke combinations that will execute it.

NOTE: ***C-h a*** searches specifically for those functions that contain the specified string. This can cause problems if you have a command function in mind, but do not know the specific command name. For this reason, it is best to be as non-specific as possible. If you are looking for commands for killing text backwards, but ***C-h a kill-backwards*** does not reveal anything, try *kill* or *backwards* - or just *back* - in place of *kill-backwards*.

Navigating a File

The arrow keys allow you to move the cursor on the screen, but many shortcut commands allow you to navigate a file more quickly and easily than you could using the arrow keys alone.

The following commands are useful for navigating files and screens in Emacs:

Command	*Description*
C-n	Moves down to the next line.
C-p	Moves up to the previous line.
C-b	Moves backward over one character.
C-f	Moves forward over one character.
M-f	Moves forward one word.
M-b	Moves backward one word.
C-a	Moves to the front of the line.
C-e	Moves to the end of the line.
M-a	Moves to the beginning of the sentence, as defined by punctuation.
M-e	Moves to the end of the sentence.
M-<	Moves cursor to the beginning of the file.
M->	Moves cursor to the end of the file.
C-v	Moves forward one screen.
M-v	Moves backward one screen.

In addition to paging through a file manually, you can go to a specific line number. To do this, type the following command from within Emacs:

```
M-x goto-line
```

Emacs will prompt you with the following:

```
Goto-line:
```

Type the number of the desired line and press ***<Return>***.

Editing Text

Emacs is a text editor. Its primary function is the manipulation of text within a file. The following section details those areas linked to the powerful editing ability of Emacs.

Typing text

Anything you type is inserted directly into the text at point. If you wish to edit in *typeover* or *overwrite* mode, use the following command from within Emacs:

```
M-x overwrite-mode
```

Typed text will now be written directly over the existing text. Because overwrite mode is a minor mode, it is possible to return to insert mode by re-executing the command.

In Emacs, all characters you type are inserted into the text to the right of point. To delete text you have just inserted, use the ***<Delete>*** key, which deletes the character to the left of point.

To end a line and start a new one, press ***<Return>***. When this key is used in the middle of a line, the line is broken into two lines at that point.

If you wish to insert a new line before the existing one, the best way is to make a blank line first and type the text into it. The following are commands used to create and delete blank lines:

Command	*Description*
C-o	Creates a blank line in front of this one and moves point there.
C-x C-o	Deletes all but one of many consecutive blank lines.

If you add too many characters to one line without breaking it with a carriage return, the line will occupy two (or more) lines on the screen with a backslash (\) at the extreme right margin of all but the last line. The backslash indicates that the following screen line is not really a distinct line in the file, but is the continuation of a line too long to fit the screen.

If you would like Emacs to move automatically to a new line when the text in the current line approaches the right-hand margin, you must invoke the wrap function. To execute it, use the following command from within Emacs:

```
M-x auto-fill-mode
```

This function eliminates the need to type **<Return>** after each line has filled up with text.

Erasing text (Killing and deleting)

The following commands are useful for erasing text:

Command	*Description*
<Delete>	Deletes the character to the left of point.
C-d	Deletes the character to the right of point.
M-<Delete>	Kills the word immediately to the left of point.
M-d	Kills the next word to the right of point.
C-k	Kills from point up to the next carriage return.
M-k	Kills to the end of the current sentence, as defined by punctuation.

NOTE: If you type **C-d** at the end of a line, the carriage return will be deleted, and that line and the next line will be joined together.

Recovering text (Yanking)

Whenever you kill a section of text larger than a single character, Emacs stores it for you in a special buffer from which you may recall it later.

To yank the text back or recover it, use the command **C-y**. This command can be executed at any location in the file; you do not have to be in the original location of the text to use the **C-y** command. This is a good way to move text around in a file.

The **C-y** command brings back only the most recently killed text. Previously killed text is not lost, however. You can retrieve it by using the **M-y** command. After you have used the **C-y** command to recover the most recent kill, typing **M-y** will replace the yanked text with the previous kill. Typing **M-y** repeatedly brings back earlier and earlier kills.

After yanking back some text, you may notice that the first character in the yanked text is displayed in reverse video. This is because Emacs always sets a mark at the beginning of a yanked region. See the section *Setting a Mark (Blocking)* for more information.

NOTE: Killing and deleting are different actions. Killed sections can be yanked back, but deleted sections cannot. Deleted text can be recovered by means of the **C-x u** and **C-_** commands, though it may require some work to recover the deleted text.

Moving text

Killing text and yanking it to a new position is an easy way to move it. You can yank a section of killed text any number of times to create copies in different places.

Command	*Description*
C-k	Kills from the cursor postion to end of line.
C-y	Yanks the last section of killed text, inserting it into the buffer at the current position.
C-w	Kills from mark to current position.
C-g	Quits. This stops whatever Emacs is doing.

Sorting text

Text may be sorted either alphabetically or in reverse alphabetical order. In both cases, text is sorted by the first letter on each line. This is known as *line sorting*.

To sort lines, you must first block off a region of text (see the section *Setting a Mark (Blocking)* for information on blocking). After you have blocked a region, you can execute a forward line sort by using the following command from within Emacs:

```
M-x sort-lines
```

To execute a reverse line sort, block off the text and use the following command:

```
<ESC>-1 M-x sort-lines
```

This executes the *sort-lines* function with the first parameter, which is *reverse*.

Searching for a string

Like other editors, Emacs has commands that allow you to search for occurrences of a specific string. The search command is incremental: it begins to search for the string on a character-by-character basis as you type it. The commands to search are:

Command	*Description*
C-s	Searches forward.
C-r	Searches backward.

Each command reads in characters and positions point at the first occurrence of the indicated character string. If you type the **C-s** command and then follow it with *f*, for example, point moves to the right of the first *f*. When you add an *i* to this sequence, point will move to the right of the first occurrence of the sequence *fi*. To find the next occurrence of the indicated character string, type the search command again (in this case, **C-s**).

If you are in the middle of an incremental search and type **<Delete>**, the last character in the search string is erased, and the point position reverts to the last occurrence of the new string. This is a useful way to back up during a search. Alternately, one can use **C-s** to page forward through a document, and **C-r** during the same search to work backwards through the text.

A search is terminated by pressing **<Return>**.

Replacing text

Global search and replace operations are powerful tools in Emacs, as they allow you to replace instances of one string in a file with another string. To replace every instance of *foo* with *bar*, for example, you would type the following from within Emacs:

```
M-x replace-string
```

This command prompts you for `Replace string:` and `Replace string with:` strings. For example, you can type *foo* and *bar* respectively.

NOTE: This command will replace only those instances that occur after point, so if you want to cover the whole buffer you must move to the beginning of the file to execute this command.

If you wish to replace only some occurrences of *foo*, you cannot use an ordinary ***replace*** command. In addition to the replacement operation found in most editors, Emacs contains a ***query-replace*** option that allows you to replace one string sequence with another, prompting you for replacement at every occurrence of the indicated string. To execute this option, use the following command from within Emacs:

```
M-x query-replace
```

This command prompts you for `Query replace:` and `Query replace with:` strings. Since these may contain newlines, you must terminate each string with either ***<Esc>*** or ***C-d***. Once you have entered the strings, Emacs moves point to subsequent occurrences of *foo* and waits for you to respond. The commands you can type when you are shown an occurrence of *foo* are listed below:

Command	*Description*
<Space>	Replaces this occurrence of *foo*.
<Delete>	Skips to the next occurrence of *foo* without replacing this one.
<Esc>	Exits without replacing any further instances of *foo*.
.	Replaces this occurrence of *foo* and then exits.
!	Replaces all subsequent occurrences of *foo* without further prompting.
^	Returns to the previous occurrence of *foo* (or its previous location, if the occurrence was replaced).

It may be necessary to replace carriage returns in the text. Because ***<Return>*** is used to end a search condition, there is a control command used to represent the carriage return:

```
C-j
```

This command can be used at any point in the search string to represent ***<Return>***.

Setting a mark (Blocking)

It is often useful to work with one block of text at a time. Emacs allows you to set a *mark* at a particular point, and move up or down within the file in order to delineate a region of text. The area between the mark and point is called a *region*.

The following commands are used to mark regions:

Command	*Description*
C-<Space>	Sets mark at current point.
C-@	Sets mark at current point (alternative to ***C-<Space>***).
C-x C-x	Exchanges point and mark.
M-h	Marks paragraph.
C-x h	Marks entire file.

Spell checker

There is an interactive spelling checker in Emacs that is invoked according to a specific spell command. The spelling checker can check a particular word, a single region, or the entire buffer (file). The following commands invoke the spell checker from within Emacs:

Command	*Description*
M-x ispell-word	Checks word after cursor.
M-x ispell-region	Checks a marked region.
M-x ispell-buffer	Checks entire buffer (file).

The spell checker offers suggestions as well as the opportunity to edit errors.

NOTE: If the spelling for a single word is checked and corrected, Emacs substitutes the corrected spelling for the old one throughout the area checked.

File and Directory Operations

Emacs has extended commands used to perform many operations on files. Some of them are listed below:

Command	*Description*
M-x view-file fn	Scans or reads the file *fn* in sequential screens without editing the file. Type ***<Space>*** to see the next screen. Type anything else to execute the command and exit.
M-x write-file fn	Writes the contents of the buffer into the file *fn*, and then visits that file. Think of it as a way of changing the name of the file you are visiting or making a copy of the file you are editing.
M-x insert-file fn	Inserts the contents of the file *fn* into the buffer at point. Point is left unchanged, and a mark is set after the insertion. This command can also be invoked as ***C-c i***.
M-x write-region fn	Writes the region (the text between point and mark) to the file *fn*. The buffer is not changed.
M-x append-to-file fn	Appends the region to the file *fn*. The text is added to the end of the buffer (file).
M-x prepend-to-buffer	Prepends the region to the beginning of the specified buffer (file).

You can change the present working directory by means of the ***change directory*** command ***C-c C-d***. After you execute this command, Emacs will prompt you for a directory name. Once you have provided a directory name, your new directory will appear in the second line of the echo area. Some of the more useful directory commands are listed below:

Command	*Description*
C-x d	Displays a verbose directory listing (equivalent to the UNIX command ***ls -al***).
C-x C-d	Displays a concise directory listing (equivalent to the UNIX command ***ls***).
C-x C-f	Finds a file and changes to that file's directory.

To change to your login directory, simply press ***<Return>*** when prompted for a directory name.

Printing

There are two ways to print from within Emacs. You can print either a region of text or the entire buffer (file).

To print a region of text, first mark the region, and then issue the command:

```
M-x print-region
```

To print the entire buffer, issue the command:

```
M-x print-buffer
```

In order to use these commands, you must first set a variable called *lpr-switches*. To set this variable, use the following command from within Emacs:

```
M-x set-variable
```

When Emacs prompts you for the variable to set, reply with:

```
lpr-switches
```

Emacs then asks you what value to set for *lpr-switches*. Respond with:

```
'("-Pprinter-name")
```

where *printer-name* is the name of the printer to which you want your text printed. The default Public IT Service Area printer is *bell*, located in Bell 101. For more information on printing, see the CIT document *JumpStart: UNIX Printing*, available at any Public IT Service Area.

If you would these defaults set up automatically every time you run Emacs, add the following line to your *.emacs* file:

```
(setq-default lpr-switches '("-Pprinter-name"))
```

where *printer-name* is the name of the desired default printer.

Customizing Emacs

All features customized during an editing session are lost when the session ends. To make this customization permanent, you must create an initialization file named *.emacs* to initialize the customization in each session.

Emacs loads the *.emacs* file found in your home or login directory. The file, if it exists, should contain *Lisp* code. Each Lisp expression takes the following form:

```
(function-name A1 A2 ...)
```

where *function-name* is the name of the function, and A_1, A_2, etc., are *arguments* to that function. For example, **`(setq fill-column 63)`** calls the function **`setq`**, which is used to set the variable *fill-column* to 63.

The value of a variable can be a constant, another variable, or a function. Most variables are constants, which can be numbers, strings, characters, *True*, *False*, and other Lisp objects. Strings are one or more characters in double quotes, including newlines and special characters. For example, the following strings are used to represent common special characters:

String	***Represents***
`\n`	Newline
`\b`	Backspace
`\r`	Carriage return
`\e`	**`<Esc>`** key
`\\`	Backslash
`C-`	Prefix for a **`<Control>`** character
`M-`	Prefix for a **`<Meta>`** character

Characters are different from strings and are denoted by a question mark (*?*) followed by the character. For example, use the sequence `?^` to denote the character ^. In addition, `t` stands for true and `nil` stands for false. Other Lisp objects must be preceded by a single quote (`'`) to be considered constants rather than commands to be interpreted.

There are many Lisp expressions that one could include in a *.emacs* file. Some of the more useful of these expressions are listed below:

`(setq default-major-mode 'text-mode)`
Makes *text-mode* the major mode for new buffers.

`(setq text-mode-hook 'turn-on-auto-fill)`
Turns on *auto-fill* when *text-mode* is selected.

`(global-set-key "\C-c\C-tl" 'indented-text-mode)`
Defines a key to start the major mode *indented-text-mode*.

`(global-unset-key "\C-x\C-v")`
Unsets a key to use the key sequence for another command or a prefix.

Additional Help

Emacs is a powerful editor that supports a wide range of uses. This document is only a brief introduction to Emacs. It does not include information on using all its capabilities. Discovering the various possibilities offered by Emacs requires experience and practice, but the potential benefits are large. Emacs can be used to edit and send mail messages. It can be used to read newsgroups, and to program, debug, and compile programs. It is even possible to run your shell session entirely from within Emacs.

A wide variety of books cover more of the editor than does this document. If you are looking for one of these books, you should first check the UB Libraries. You can also purchase introductory Emacs books at most local bookstores with computing sections.

The *GNU Emacs Manual* is also available online as a reference at the following URL:

`http://www.gnu.org/manual/emacs-20.3/emacs.html`

Many newsgroups contain information that can expand your knowledge of Emacs, including:

comp.emacs
gnu.emacs.announce
gnu.emacs.bug
gnu.emacs.gnews
gnu.emacs.gnus
gnu.emacs.help

CIT also offers a number of tutorial and reference documents free of charge at the CIT Help Desk and other IT Computing Sites. Of particular note is the CIT document *Reference Card: GNU Emacs*, which contains a list of frequently used Emacs commands. These documents are also available online at the following URL:

`http://wings.buffalo.edu/computing/Documentation/`

For additional help, contact the CIT Help Desk, 216 Computing Center, at (716) 645-3542, or send e-mail to *cit-helpdesk@buffalo.edu*.