Introduction to GNU Emacs

Abstract

This document explains how to use GNU Emacs to create and edit text files, use RMAIL to read mail, and read Usenet news with GNUS.

For a reference card of GNU Emacs commands, see document Unix 5.01.

Table of Contents

Description of the Emacs Editor

An Introduction

GNU Emacs (hereafter just “emacs”) is a powerful, flexible editor available from the Free Software Foundation. Emacs has lisp-like editing macros which allow for extensive customization. It is available on Unix, Macintosh computers, and for Windows (3.1, 95, and NT). At Rice, however, it is installed only on Unix workstations.

This document begins with opening and closing a window, moves on to editing files, and finally covers more advanced topics.

Special Keys

Emacs utilizes certain modifier keys to execute commands. The two that you will encounter most are **M-**, or **ESC-**, (Meta and Escape keys) and **CTRL** (Control key).

If a terminal has a Meta key, you will be able to use the Meta characters. Different keys serve as the Meta key depending upon the type of terminal you are using. The following chart is a list of terminal types and their respective Meta keys:

Terminal Type	Key Sequence
Sun SPARCstation	Diamond(◊) Key, ESC, or “Alt”
Macintosh or PC (remote login)	ESC

TABLE 1.1. Meta Keys on Different Terminals

The Diamond (◊) key works like a Shift key—it is held down as the following character is pressed. The “Alt” and Alternate keys also work like a Shift key.

If you are using the Escape (“ESC”) key, press ESC and release, then type the key sequence. You can use either ESC or the Alt key for Meta on most X terminals, but from a PC or Macintosh you’ll need to use the Escape key.

The Emacs Screen

What you see when you call up an emacs window is a region of space into which you may edit several buffers of text. The window consists of four parts:

- **pull-down menu bar** is the top line of the screen, containing mouse-activated versions of the most commonly used emacs commands
- **text region** is the area of the screen where text is entered

- **mode line** is the line just below the text that appears in inverse video (if supported by your terminal) and contains the filename you are editing, the mode you are in, and where you are in the file
- **echo area** (also “minibuffer window”) is the space below the mode line where information or error messages appear; it will often tell you the status of your last command, or tell you that a command has just been completed

A *buffer* is a window in which you are working. A feature of emacs is that it can handle any number of buffers at once, and you may switch among them. Whenever you “visit” a file and do not exit it, it remains in the buffer list until you quit emacs or exit that buffer. Later in this document, you will learn how to switch among buffers both manually and by using the Dired buffer.

Characters in Emacs

There are certain special function keys that you will use regularly in emacs for different classes of commands. The following conventions will be used throughout the remainder of this document:

- RETURN indicates the Return key
- TAB indicates the Tab key
- DEL indicates the Delete key
- SPC indicates the Space bar
- M- indicates the Meta (and Escape) key; explained on page 4
- ESC indicates the Escape key; often used in place of Meta on some terminals
- C- indicates the Control key
- C-*character* is executed by holding down the Control key while pressing the character named by *character*; ex: C-d, hold down the Control key, and press “d”

How Do I Start?

If your terminal supports X Windows, on a system supported by Information Technology (such as Owlnet) type **gnuemacs &** in an xterm to begin emacs as a separate process. On systems not administered by Information Technology, you can usually start emacs with the command **emacs &**. If you have not modified the configuration of your window manager, emacs will also appear in the “Applications” menu which appears when you hold the right mouse button down. Selecting gnuemacs from that menu will also start up an emacs window on your screen.

When logging in via a terminal without X Windows graphics capabilities, type **gnuemacs**, and your shell window becomes an emacs window. While you are in emacs, you cannot type any shell commands unless you exit or suspend emacs. When started without an initial filename, emacs provides a buffer titled “*scratch*”. The scratch buffer does not detect that changes have been made to it, thus it will not prompt to save the buffer if you exit emacs without saving the scratch buffer’s contents. If you attempt to save the scratch buffer, it will prompt for a file name to save it under. Saving text will be discussed at the end of this section.

You can type **C-h t** (Tutorial in the Help pull-down menu) to access the emacs tutorial and/or type **C-h i** (Info from the same menu) for an introductory browser menu. The tutorial is thorough, providing some exercises and useful commands. The introductory browser includes information on basic concepts as

well as complex customizations; it includes the entire emacs manual and manuals for other popular GNU products (the “GNU” prefix indicates products written by members of the Free Software Foundation).

Moving Around in a File

When emacs has been started, you can enter text by typing in the text buffer. Once you have finished typing a few lines of text, try using any of the commands in Table 1.2 to move around in the file.

Command	Description
C-p **C-n**	moves cursor back one line moves cursor forward one line
C-a **C-e**	moves cursor to the beginning of the line moves cursor to the end of the line
M-a **M-e**	moves cursor backward to the beginning of the sentence moves cursor forward to the end of the next sentence
C-b **C-f**	moves cursor backward one character moves cursor forward one character
M-b **M-f**	moves cursor backward to the beginning of the previous word moves cursor forward to the beginning of the next word
M-[**M-]**	moves cursor back one paragraph moves cursor forward one paragraph
C-v **M-v**	scrolls down one page scrolls up one page
M-< **M->**	moves cursor to the beginning of the file moves cursor to the end of the file

TABLE 1.2. Basic Commands

In X Windows, if you click the cursor on a point in the text, emacs begins inserting characters at that point. Text will be entered wherever the cursor is in the emacs window.

Basic File Commands

To save a file, type:

C-x C-s

Choosing Save Buffer from the File menu will also save the currently selected buffer. Since you are editing the scratch buffer, emacs will prompt you for a filename under which to save the text. Normally, emacs will save the file as the current filename. To find a file and load it into the current emacs buffer, type:

C-x C-f *filename*

You can also use Open File from the File menu. Once your text has appeared, try scrolling through the text using **C-v** and **M-v**. Emacs always keeps a few lines of text from the previous page when advancing to the next page to give you a reference point.

If you find yourself stuck at any point while issuing commands, you can type **C-g** to abort the most recent command you entered, which allows you to re-issue commands. Sometimes you will need to abort several commands in order to return emacs to a state where it will accept input normally.

Remember:

- **C-x C-s** will save; if a filename needs to be supplied, emacs will ask for one
- **C-x C-f** will search for *filename*
- **C-g** will abort the most recent command

Options to Edit Text

Once you have saved a file, what if you want to make changes? In this section, you will build upon the basic commands that you already know and learn some of the more advanced commands.

Time-Saving Commands

Repeating Commands

At times, you may want to repeat a command. This can be easily accomplished by using the command:

C-u (*# of times to repeat command*) (*command to be repeated*)

For example, to create a row of several n's in your document, place the cursor where you want to begin, then type **C-u 15 n** and fifteen n's will appear.

Another simple way to repeat a command is to use the Meta key:

META (*# of times to repeat command*) (*command to be repeated*)

You may encounter some commands that do not repeat in the same way that they operate as a single command. For instance, **C-u 4 C-v** will not scroll 4 screens, but will instead only scroll 4 lines. In this case, you may choose to write your own keyboard macros or map keys to functions, which are explained in the section called *Basic Customization of Emacs*.

A Useful Command: Goto Line

If you are writing a program in emacs or just looking at text files, you may want to refer to a particular line in the file. Emacs has this command: **M-x goto-line** RETURN, which will respond with `goto-line:` in the echo area. Type the line number, press RETURN, and the cursor will be placed on that line.

Automatic Word-Wrap

Sometimes you may find it useful to have emacs automatically word-wrap your sentences for you. When you type **M-x auto-fill-mode**, your fill column is set to 70, and any text over the 70 character limit will be placed on the next line. If you alter the middle of a sentence however, your paragraph will not be correctly word-wrapped. To correct that, you need to type **M-q** within that paragraph.

Another useful command is **C-x f** which sets the fill column to wherever the cursor is located. Be careful to have your cursor where you wish your "right margin" to be before invoking this command- it can and will be set to 0 if that is where your cursor is located.

Moving, Deleting, and Making Blocks of Text

Using the Mouse

Other important skills in emacs are moving, copying, and deleting text. In X Windows, you can use the mouse to rearrange text by clicking the left button to mark one end of the text to be moved, then moving the mouse to the other end of that text and clicking the right mouse button. Dragging with the left mouse button held down selects a region of text that can be pasted at the cursor location. If you wish to extend a previously selected region, move the mouse to the end of the extended area and click the right button. Be careful, though; clicking the right button twice at the end of the selection will kill the region thus selected. Then you can copy this text anywhere by positioning the cursor at any particular spot and clicking the middle mouse button to insert the text at that point.

You can also use emacs keyboard commands to move text. **C-SPC (space bar)** sets the mark -- in other words, marks the beginning of the text to be moved. Then, move the cursor to the end of the region and type **C-w**. This moves the text into the *kill ring*. Your text will disappear, and you can place it anywhere you want to put it. The following table lists some command options available when using emacs to move text.

Command	Description
C-SPC	sets the text mark
C-@	also sets text mark
C-w	marks end of text selected, cuts text and places it in the kill ring
M-w	marks end of text selected and copies text to the kill ring, leaving original text intact
C-x C-x	sets the mark at the cursor's current position, then moves the cursor to the previous mark
C-y	inserts the text from the kill ring wherever the cursor is located
M-y	yanks the previously killed text from the kill ring. (Normally, the last 20 deleted text blocks are in the kill ring.)

TABLE 2.1. Commands for Moving Text

The Edit pull-down menu includes commands to cut, copy, and paste text selected with the mouse or by **C-SPC**. Emacs can also exchange text with other X applications. Killing or copying text in emacs makes that text available for pasting into other applications (see document UNIX2, *Introduction to the X Window System*). The emacs yanking commands insert the latest selection set by other applications, and add the text to the kill ring.

Editing Text in Emacs

Incremental Search

Emacs can easily search for a particular string or a word from any point in the text. Use:

- **C-s** to search forward through text
- **C-r** to search backward through text

Emacs will scroll to the next occurrence of the string or word as you type each character. Repeat the command (**C-s** or **C-r**) to search for the next occurrence. To abort the search and return to where you were before beginning the search, type **C-g**. However, if you have used any command that changed the position of the mark, you will be returned to the latest mark set, not your original position.

The Search option on the Edit pull-down menu is not an incremental search. It searches forward from the point to the specified string and sets the point at the end of the occurrence found. Search Backwards will perform the same kind of search from the end of the file to the beginning.

Search and Replace

There are two ways to replace multiple occurrences of a word or expression.

- **M-x replace-string** RETURN *old* RETURN *new* will replace every occurrence of *old* with *new* without prompting.
- **M-x query-replace** RETURN *old* RETURN *new* RETURN prompts at each occurrence of *old*, asking whether it should be replaced with *new*. **M-%** also activates query-replace.

Responses that query-replace will accept include:

Command	Description
SPC or **y**	replace this match, go to next match, prompt
DEL or **n**	do not replace this match, skip to next match
.	replace first match and exit query-replace
,	replace this match, do not move immediately
!	replace all remaining matches without further prompting
^	back up to previous match
ESC or **q**	exit query-replace

TABLE 2.2. Valid repsonses in Query-replace

Inserting and Deleting Text

Emacs automatically allows you to insert text wherever the cursor is placed. If you need to make deletions at some point in the text, you can use the commands in the following table.

Command	Description
DEL	deletes the character just before the cursor
M-DEL	deletes text from the cursor to the beginning of the word
C-d	deletes the next character after the cursor
M-d	deletes from the cursor to the end of the word
C-k	deletes from the cursor to the end of the current line
M-k	deletes text through the end of the current paragraph

If, in the course of rearranging text, you find that you need to retrieve a portion of text you just killed (sent to the kill ring), you can "yank it back" using the command **C-y**. If you have text that was killed earlier, you can still recall it by typing **M-y** (after typing **C-y**), which will bring back a series of your earlier kills. On the Edit pull-down menu, several selected regions can be stored in the Choose Next Paste list; highlight this command to paste a previously marked selection.

What If I Made a Mistake?

There is a difference between "killing" and "deleting." When you kill text, you move it into memory and can call it back later. When you delete text, you cannot "yank" it back, but you can still resort to the "undo" command to bring it back. To do this, type **C-x u**. This command, and **C-_**, will cancel the effect of the command immediately previous. Alternately, choose Undo from the Edit menu. You may find these commands very useful!

Modes in Emacs

Emacs has several modes that can be used when entering different types of text. Fundamental mode is the least specialized mode. It has no mode-specific rebindings and all variables are set to their default options. Text mode can be used for English text, TeX mode is used for TeX and LaTeX, nroff mode is used for nroff input, and outline mode is used to write outlines for documents. Emacs will automatically put you in a particular mode according to the suffix of your file. The default mode for the scratch buffer is Lisp Interaction; to change modes, type **M-x [name-of-mode]-mode**. This is a toggle command; if a mode is on, it will be turned off, and if it is off, it will be turned on. This document assumes the use of text mode; however, you can find information about the other modes by looking through the info browser.

A Word About TeX Mode

TeX mode is enabled when you call up a file that is suffixed with ".tex" or when you activate the mode. TeX and LaTeX are popular word processing/typesetting languages. There are three variants of this mode: **M-x plain-tex-mode** which activates the TeX mode, **M-x latex-mode** which activates the LaTeX

mode, and **M-x slitex mode** is for SliTeX input. **M-x tex-mode** will attempt to guess whether you need SliTeX or LaTeX. Its features include checking balance of delimiters, smart quotes, and whether to use TeX, LaTeX, or SliTeX on all of the file.

Help While in Any Mode

If you need help in a particular mode, type **C-h m** to open a help buffer in that window.

Programming Modes

If you use emacs for writing programs, you may find it convenient to use emacs programming modes. They have features that include indenting lines, tabbing, and balancing parentheses. Programming modes are automatically activated by emacs as it reads the suffix to the filename. To activate any of these modes, you call up the file and emacs will put the file into whichever mode pertains, depending on the suffix of the file. A file that ends in " **.f** " will be Fortran, a file that ends in " **.c** " , **" .C** ", or " **.cc** " will be C, etc. For more information about these modes, you can refer to the info browser in any emacs window by typing **C-h i**, or selecting the appropriate item from the Help menu.

Rectangles

This section deals with yanking or adding rectangles of space or characters to text in your buffer. Emacs allows you to define a rectangle of any dimension (within the bounds of the window) and either add or delete that from the buffer.

To define a rectangle, set the mark (using **C-SPC**) at the upper left corner and the point at the lower right. A rectangle thus specified is called the "region-rectangle" because you control it in the same way in which you control a region.

The commands for moving rectangles are listed in Table 2.4.

Command	Description
M-x delete-rectangle	delete the rectangle defined, moving text toward the left edge of the region-rectangle
M-x kill-rectangle	same as above, except that the region killed is saved as the "last killed rectangle"
M-x yank-rectangle	yank the last killed rectangle at mark specified
M-x clear-rectangle	clear the rectangle by replacing it with blank space

TABLE 2.3. Rectangle Commands

Managing Emacs

Buffers

You can have several buffers in one emacs window. While you are working on one file, you can call up another in the same window, and another, etc., although usually you will not have more than two buffers showing per window.

Table 3.1 lists commands to use in managing buffers:

Command	Description
C-x b	toggle among buffers
C-x 0	causes the buffer with the cursor to disappear
C-x 1	causes all buffers without the cursor to disappear
C-x 2	vertically splits the buffer into two
C-x 3	horizontally splits the buffer into two

TABLE 3.1. Buffer Control Commands

The Buffers pull-down menu contains a list of all current buffers; you can display the list of buffers in a new buffer called Buffer List by choosing List All Buffers. See the section, *Dired for Buffers*, below for more information on how to edit the buffer list. Select an individual buffer by highlighting its name.

If you decide to exit the window at any point, and haven't saved any of the buffers in that window, emacs will ask if you wish to save each buffer before closing it. This prevents accidental erasure of any updating you may have done to your files. If you decide to save your changes, emacs will save the original file as a backup to the newly saved copy. The default extension for these backup files is ~.

The New Frame option under the File menu allows you to open multiple windows for your buffers. When opened, a new frame will display the currently selected buffer in the old frame. Starting RMail or GNUS will automatically open a new frame for those processes.

Using Command Completion

Some of the emacs commands may seem too long to be typed or memorized. To compensate, emacs has a feature called command completion. While typing a long, hyphenated command, you can retrieve the rest of the command set by hitting SPC or TAB at intervals to cause the command to complete.

- SPC completes to punctuation marks
- TAB completes as far as possible

For example, if you must use the command **M-x goto-line**, you can use either SPC or TAB to complete it. After typing **M-x g**, type "**o**", press SPC or TAB, and the command will complete through **M-x goto-**. When you press SPC or TAB again, a mini-buffer will appear with the possible completions for the

command. In this case, there are only two, and by typing an "**l**" for **line**, you can then type either TAB or SPC to reach the command you want without needing to type the entire command.

To use command completion, type the first few letters of the command, press SPC (or TAB) then type the next letters in the command followed by SPC or TAB, and you'll eventually arrive at the appropriate command without having to remember the entire sequence. If you've forgotten part of the sequence, a mini-buffer appears with possible completions based on the letters you've typed of the command sequence so far. You can select the proper completion from this list by clicking the middle mouse button on it.

Using Dired

Dired is the emacs menu system that allows you to edit several files at once, by helping you switch from buffer to buffer in one window. It is a file manager/DIRectory EDitor, giving you a list of files and directories from which to choose files to open in the window.

To load, type: **C-x d**, **M-x dired**, or load a directory using **C-x C-f**. Once you have opened a dired window, you can use it to switch between it and other emacs buffers.

The Dired buffer will appear as a list of all files in a particular directory, and you may enter a command by each listing. Some Dired commands are listed Table 3.2.

Command	Description
SPC or **n**	equivalent to C-n
p	equivalent to C-p
d	delete; flag this file for deletion
DEL	move up and remove deletion flag
g	re-initialize buffer from actual disk directory and update any changes made to files
u	undo; remove the deletion flag on this line
x	delete all files flagged for deletion
#	flag all auto-save files for deletion
~	flag all backup files for deletion
.	flag all excess backup files for deletion; the first and last are left alone, all other backups will be deleted.
c	copy; copies the file into a minibuffer; you must supply a file name to copy to
f	find; visits the file you have selected; if the line is not a file, but a directory, dired will be enacted upon that directory
o	similar to find file, but uses another buffer to display the file
r	rename; allows you to rename a file by entering a new name
v	view; allows you to conveniently view a file, you may scroll through but may not alter the file

Command	Description
1	buffer selected is to be shown in full-screen window
2	select two windows; the previous buffer in one, and the selected buffer in the other window

Dired for Buffers

You can manage your several emacs buffers by using a special dired for buffers called the “buffer-menu.” If you have been working in several different buffers while in an emacs window, you can manage them by calling up a menu listing using the command: **M-x buffer-menu**. **C-x C-b** also invokes the buffer menu. This will call up a list of all the buffers in that window, and you can use the commands listed in Table 3.3.

Command	Description
d	delete; delete buffer, shows up as a “D” by the line
k	kill; same as delete
C-d	similar to “d,” but then moves up a line instead of down a line
s	save; allows you to save the buffer
~	mark buffer “unmodified”
x	execute the deletions, and save buffers
u	remove any request made for the current line, and move to next line
DEL	move to previous line and delete any request made for that line
SPC	move to next line and delete any request made for that line

TABLE 3.2. Buffer Menu Commands, to be used in the “*Buffer List*” buffer

Basic Customization of Emacs

Now that you know all the basics, what if you would like to map your keyboard for emacs, or write your own keyboard macros? This section will show you the basics and provide some examples of often-used commands you might want to map.

Customization

Simple Keyboard Macros

Defining keyboard macros is accomplished in emacs using the commands in Table 4.1.

Command	Description
C-x (	start defining keyboard macro
C-x)	end defining keyboard macro
C-x e	execute most recent keyboard macro
M-x name-last-kbd-macro	give a command name (for the duration of the emacs session) to the most recently defined keyboard macro
M-x insert-kbd-macro	insert in the buffer a keyboard macro's definition as Lisp code

TABLE 4.1. Commands to Define Keyboard Macros

If you wish to temporarily define a macro, type

C-x (

macro key sequence

C-x)

where *macro key sequence* is the series of keystrokes which make up the macro, and your macro is defined. You can then type **C-x e** to execute the last defined macro, and your command is thus shortened to a single keystroke.

You can also define a keyboard macro (for the duration of the emacs session) as a command to be executed in conjunction with the Meta key. First, name the last keyboard macro by typing **M-x name-last-kbd-macro**, then enter a name for the macro. To execute this macro, simply type **M-x** *name-of-macro* and it will be executed. You may use **C-u** as explained in the subsection *Repeating Commands* to repeat your macro multiple times.

By naming a macro, you ensure that it won't get overwritten when you define another macro (unless you give the new macro the same name). To save the macro for future emacs sessions, save the Lisp code into a file (normally your *.emacs* file) by opening the file in which you want to save the macro, then typing:

M-x insert-kbd-macro RETURN *macroname* **RETURN**

This will insert Lisp code for the command into the file.

If you inserted the code into your *.emacs* file, the macros will be defined each time you run emacs. If you have saved it in a separate file, you must use **M-x load-file** to load the Lisp library.

A *.emacs* file (pronounced “dot-emacs”) is a file that is loaded each time you start emacs. It contains Lisp code for all of the keys mapped for use in your emacs window, as well as code for user-defined macros.

Why add macros to your .emacs file if you can save them as a separate file? Adding keyboard macros to your .emacs file defines them for all of your emacs sessions. When you save the macros in a separate file and then use **M-x load-file** to load the Lisp code, the macros are only defined for that emacs session.

Key Binding

Mapping Key Sequences

Global keymap and local keymap are the two types of key mapping. When you change a global keymap, the change becomes effective in all major modes of emacs (unless there are overriding local definitions). When you change a local keymap, you change only the buffers executing that particular major mode.

M-x global-set-key RETURN *key command* **RETURN**

will define *key* globally to execute *command.*

M-x local-set-key RETURN *key command* **RETURN**

will define *key* locally (in the current major mode in effect) to execute *command*.

For example: **M-x global-set-key RETURN C-f doctor RETURN** will re-define **C-f** to execute **M-x doctor** each time the key sequence **C-f** is executed.

Local keymap works in a similar way.

Miscellaneous Features

RMAIL

If you'd like to use an emacs-based mail reader, you might want to try RMAIL. RMAIL, unlike pine or elm, only supports one folder for all mail messages. If you save a lot of mail and want to have it sorted in different folders, RMAIL may not be for you. To enter RMAIL mode, type **ESC-x rmail**. This creates a file in your directory called **RMAIL**, containing consecutive mail messages ordered by date and formatted to be read by RMAIL. Once you have created this file you should change permissions on it using the UNIX command **chmod** so that others cannot read your mail file. (To learn more about using **chmod**, enter **man chmod** at the shell prompt or read the document UNIX1, *Introduction to the UNIX Operating System on IT Systems*.)

From there, you can manage your mail by using the commands in Table 5.1.

Command	Description
m **C-c C-c**	mail: causes the mail window to split into two buffers, the second being your composition mail buffer (while in the composition mail buffer) causes your mail to be sent
r	reply: causes a mail buffer to appear, addressed in reply to the message which you are currently reading
d	delete: deletes a message from your RMAIL file
e	expunge: saves the mail file, throwing out deleted messages
x	same as expunge
u	undo: this will undo any RMAIL command you have specified
q	quit: saves the mail file, expunges deleted messages, and closes RMAIL, leaving the previous buffer visible
h	opens a directory of your mail from which you can delete and select messages
i	runs RMAIL on a particular RMAIL file when in the mailer
w **C-c C-c**	allows you to edit messages in RMAIL will save your changes
t	toggle: shows all mail headers
o	causes message to be saved as a separate file
s	saves your RMAIL file
g	rescans your RMAIL file for new mail
j	moves to the first message in your RMAIL file
a	add label; this allows you to distinguish messages
k	remove label; see **a**

Command	Description
-	also removes label
c	creates a mail buffer with the last letter you mailed as its content

RMAIL is easy to use and convenient once you become used to using emacs. For help with RMAIL, or when using any emacs modes, type **C-h m** in that window, and a help buffer will appear.

GNUS

Although the tin newsreader is widely used on the Rice campus, GNUS is an alternative newsreader which is available in your emacs window. To access this newsreader, type **M-x gnus**, and your editor will enter GNUS mode.

M-x gnus runs GNUS using the default news server, while **C-u M-x gnus** allows you to specify a different news server.

How you operate in GNUS depends upon which buffer you are in at the time. GNUS begins in the newsgroup buffer with a menu listing all the newsgroups to which you are subscribed.

Some of the most useful commands available in the newsgroup buffer are listed in Table 5.2.

Command	Description
SPC	enters the newsgroup on which the cursor is positioned
n	moves the cursor forward one line
p	moves the cursor backward one line
j	prompts for newsgroup name then jumps to it
u	toggles between being subscribed and unsubscribed to a newsgroup
L	lists all newsgroups including those that are unsubscribed
q	exits GNUS

TABLE 4.2. GNUS Commands While in the Newsgroup Buffer

Once you are in a particular newsgroup, a “subject” buffer will appear at the top of your window and the “article” buffer will appear below it. Table 5.3 lists the most common commands you will want to use within a newsgroup.

Command	Description
C-c C-s C-s	sorts subject display buffer by subject alphabetically, ignoring 'Re:'s; creates a "threaded" subject list
SPC	scrolls text forward one window; selects next article if page is complete
DEL	scrolls to previous page of current article
RETURN	scrolls down one line of current article
<	go to beginning of article body
>	go to end of article body
t	toggles presence of all headers of current article
h	calls up subject buffer
s	calls up article buffer
d	mark article as read, then move to next entry
D	mark article as read, then move to previous entry
u	mark article as unread, then move to next entry
U	mark article as unread, then move to previous entry
M-u	remove current article's mark, move to next entry
M-U	remove current article's mark, move to previous entry
k	mark all articles with same subject as current article as read, then select the next unread article
C-k	mark all articles with same subject as current article as read; does not select next article
n	read next article, skips articles already read
N	read next article
M-C-n	read next article with same subject
p	read previous article, skips articles already read
P	read previous article
M-C-p	read previous article with same subject
.	read first unread article
l	read the last selected article
J (number) RETURN	read article (number)
c	"catch up"-mark all articles in newsgroup as read; exit current newsgroup and return to newsgroup buffer
Q	quit the current newsgroup without updating read article info; returns to newsgroup buffer
G	exit newsgroup saving article info then reselect the newsgroup

TABLE 4.3. GNUS Subject Buffer Commands

When you want to save a particular article, GNUS allows you to save the article in the format of your choice. You can specify the default format in which you want your article to be saved using the command:

M-x gnus-Subject-save-in-

mail saves the current article in UNIX mailbox format

rmail saves the current article in RMAIL format

folder saves the current article in MH format

file saves the article in article format

To respond to or post an article while in the subject buffer, use the commands in Table 5.4 below.

Command	Description
f	followup to the current article
F	followup to the current article, including the full text of the original article
a	compose a new article; post
C	cancel the current article you posted
r	reply to the author of the current article
R	reply to the author of the current article, including the full text of the original article
m **C-c C-y**	compose a mail message in another window allows you to include a copy of the article (may be used only while in the mail buffer)

TABLE 4.4. GNUS Mailing Commands

You can customize the GNUS environment however you like by adding emacs macros to your *.emacs* file. See the previous section, *Basic Customization of Emacs*, for more information.

Problems or Question?

You can get assistance at the Consulting Center located in 103 Mudd Lab. They can also be reached at 713-527-4983 or via e-mail at problem@rice.edu. You can also submit a problem to http://problem.rice.edu.

Useful Emacs Command Summary

File Commands	
C-x C-f	find a file to edit
C-x C-s	save a file
C-x C-c	quit
C-x C-v	replace this file with a different file
C-x d	invoke Dired, the directory editor
C-x C-w	rename a file
Movement Commands	
M-v **C-v**	moves file back one screen advances file one screen
C-p **C-n**	moves cursor back one line advances cursor one line
C-a **C-e**	moves cursor to the beginning of a line moves cursor to the end of a line
M-a **M-e**	moves cursor backward to the beginning of a sentence moves cursor forward to the end of the next sentence
C-b **C-f**	moves cursor backward one character moves cursor forward one character
M-b **M-f**	moves cursor back to the beginning of the previous word moves cursor forward to the beginning of the next word
M-< **M->**	moves cursor to the beginning of the file moves cursor to the end of the file
M-[**M-]**	moves cursor back one paragraph moves cursor forward one paragraph
M-C-b **M-C-f**	moves cursor back one sexp (for programming languages) moves cursor forward one sexp
Editing Commands	
M-x query-replace or M-%	usage: **M-x query-replace RETURN *old* RETURN *new* RETURN** prompts at each occurrence of *old*, asking whether it should be replaced with *new.*
M-x replace-string	usage: **M-x replace-string RETURN *old* RETURN *new*** will replace every occurrence of *old* with *new* without prompting.
C-s	search forward
C-r	search backward
C-k	kill line
C-d	delete next character
M-d	delete word forward