Getting Started With XEmacs

July 1994
(General Public License upgraded, January 1991)

Richard Stallman

and

Rashmi Goyal

Preface

This guide is intended to help you get started on using the Emacs editor. It will show you some examples of simple customizations.

For detailed information on any topic, refer to the **XEmacs User's Manual**. This document will also refer the reader to the **XEmacs User's Manual** for more information on corresponding topics. You can also use the on-line, learn-by-doing tutorial, which you get by running Emacs and typing *C-h t* OR you can choose the **Emacs Tutorial** from the **Help** menu on the menu bar (which is located on the extreme right of the frame). With it, you learn Emacs by using Emacs on a specially designed file which describes commands, tells you when to try them, and then explains the results you see.

The first few chapters will introduce you to some basic Emacs commands. Later on, some examples of simple customizations will be shown.

To find the documentation on a particular command, look in the index. Keys (character commands) and command names have separate indexes. There is also a glossary, with a cross reference for each term.

This manual comes in two forms: the published form and the Info form. The Info form is for on-line perusal with the INFO program; it is distributed along with XEmacs. Both forms contain substantially the same text and are generated from a common source file, which is also distributed along with XEmacs.

Introduction

You are reading about XEmacs which is a self-documenting, customizable, extensible real-time display editor.

XEmacs is a *display* editor because normally the text being edited is visible on the screen and is updated automatically as you type. See section "Frame" in *XEmacs User's Manual*.

It is a *real-time* editor because the display is updated very frequently, usually after each character or pair of characters you type. This minimizes the amount of information you must keep in your head as you edit. See section "Basic Editing" in *XEmacs User's Manual*.

It is advanced because it provides facilities that go beyond simple insertion and deletion: filling of text; automatic indentation of programs; viewing two or more files at once; and dealing in terms of characters, words, lines, sentences, paragraphs, and pages, as well as expressions and comments in several different programming languages. It is much easier to type one command meaning "go to the end of the paragraph" than to find that spot with simple cursor keys.

Self-documenting means that at any time you can type a special character, `Control-h`, to find out what your options are. You can also use `C-h` to find out what a command does, or to find all the commands relevant to a topic. See section "Help" in *XEmacs User's Manual*.

Customizable means you can change the definitions of Emacs commands. For example, if you use a programming language in which comments start with '`<**`' and end with '`**>`', you can tell the Emacs comment manipulation commands to use those strings (see section "Comments" in *XEmacs User's Manual*). Another sort of customization is rearrangement of the command set. For example, you can set up the four basic cursor motion commands (up, down, left and right) on keys in a diamond pattern on the keyboard if you prefer. See section "Customization" in *XEmacs User's Manual*.

Extensible means you can go beyond simple customization and write entirely new commands, programs in the Lisp language to be run by Emacs's own Lisp interpreter. Emacs is an "on-line extensible" system: it is divided into many functions that call each other. You can redefine any function in the middle of an editing session and replace any part of Emacs without making a separate copy of all of Emacs. Most of the editing commands of Emacs are written in Lisp; the few exceptions could have been written in Lisp but are written in C for efficiency. Only a programmer can write an extension to Emacs, but anybody can use it afterward.

1 Entering and Exiting Emacs

While using Emacs you should be familiar with the following three terms:

Buffer
: A **buffer** is a region of memory holding characters. It is the basic editing unit; one buffer corresponds to one piece of text being edited. You can have multiple buffers but you can edit only one buffer at any one time. For more information, See section "Buffers" in *XEmacs User's Manual*.

File
: A **file** is a region of disk space holding characters. Emacs edits a file by reading it into a buffer, editing that buffer and writing out the buffer back to the file. To save your work permanently you have to write it to a file. So after you load and work with a file, you have to save it back.

Windows
: A **window** is a rectangular region in which a buffer is displayed. You can open multiple windows with multiple buffers and edit them by selecting the corresponding buffer. Initially, when you start emacs, it will automatically open up a window for you.

1.1 Entering Emacs

To enter Emacs type *`xemacs`* and press the Return key at the shell i.e. *`xemacs`* ⟨RET⟩. This will bring up an emacs window with '`*scratch*`' as the default buffer because Emacs must always have a buffer to work on. Then choose the **Open...** option from the **File** menu on the menubar at the top of the frame. It will prompt you to enter a filename. After you enter the filename, Emacs will read that file into the current buffer. You can also type :

`xemacs <filename>` ⟨RET⟩

directly which will bring up an Emacs frame with the *filename* as the buffer.

1.2 Emacs Frame

When you run XEmacs under X, a menu bar on top of the Emacs frame provides access to pull-down menus of file, edit, and help-related commands. The menus only provide convenient shortcuts, the options that they provide are available via key commands. You can invoke those commands from the keyboard also. For many of the options, their corresponding key commands are displayed right besides them. The five default menus on the menubar that you will see on the frame are **File**, **Edit**, **Options**, **Buffers** and **Help**. See section "XEmacs Pull-down Menus" in *XEmacs User's Manual*, for detailed information on the functions provided by the pull-down menus.

The Emacs frame has a rectangle shaped box at the extreme right and you can drag it up or down to scroll the window accordingly. Clicking on the arrows also serves the same purpose.

The last line in your window is '`the Mode line`' which will give you a description of what's going on in that particular window. See Section 1.4 [Mode Line], page 6, for more information. Below the mode line is the '`Echo area`'. Emacs uses this area to interact with the user. See Section 1.5 [Echo Area], page 7.

If you wish to open another file in a new window after you enter XEmacs, select **Open in New Frame...** from the **File** menu, which will prompt you for a filename and open a new window with that filename as the current buffer. If you want to open a new file in the same window, select **Open..** from the **File** menu. You need to enter XEmacs only once, you can edit multiple files by opening several other frames or by switching between buffers.

1.3 Exiting Emacs

There are two commands for exiting Emacs, one for *suspending* Emacs and the other for *killing* Emacs. *Suspending* means stopping Emacs temporarily and returning control to the shell, allowing you to resume editing later in the same Emacs job, with the same files, same kill ring, same undo history, and so on. This is the usual way to exit. *Killing* Emacs means destroying the Emacs job. You can run Emacs again later, but you will get a fresh Emacs; there is no way to resume the same editing session after it has been killed.

C-z Suspend Emacs (`suspend-emacs`). If used under the X window system, this command will shrink the X window containing the Emacs frame to an icon. Clicking on the icon will resume that Emacs process again. See section "Exiting Emacs" in *XEmacs User's Manual*.

C-x C-c Kill Emacs (`save-buffers-kill-emacs`). You can also select **Exit Emacs** option from the **File** menu to kill that Emacs process. If you haven't saved the file, Emacs will ask you if you wish to save the file before killing that process.

1.4 The Mode Line

When you enter XEmacs, each text window's last line is a *mode line* which describes what is going on in that window. Normally, the mode line looks like :

```
--ch-XEmacs: buf      (major minor)----pos------
```

This gives information about the buffer being displayed in the window: the buffer's name, what major and minor modes are in use, whether the buffer's text has been changed, and how far down the buffer you are currently looking.

The *ch* contains :

'`**`' if the text in the buffer has been edited

'`--`' if the text in the buffer has not been edited

'`%%`' if the buffer is a read-only-buffer i.e. it cannot be edited

buf is the name of the window's chosen *buffer*. If you are editing a file (which is the selected buffer), the file name appears in *buf*. See section "Buffers" in *XEmacs User's Manual*.

pos contains :

'`All`' if your entire file is visible on the screen.

'`Top`' if you are looking at the beginning of the file.

'`Bot`' if you are looking at the end of the file.

'*nn*`%`' *nn* will be a number corresponding to the percentage of the file above the top of the screen, for example '`52`', which means that 52% of the file is above the top of the screen.

major is the name of the *major mode* in effect in the buffer. At any time, each buffer is in one and only one major mode. The available major modes include Fundamental mode (the least specialized), Text mode, Lisp mode, and C mode. See section "Major Modes" in *XEmacs User's Manual*, for details on how the modes differ and how you select one.

minor is a list of some of the *minor modes* that are turned on in the window's chosen buffer. For example, '`Fill`' means that Auto Fill mode is on which means that lines are broken automatically when they become too wide. See section "Minor Modes" in *XEmacs User's Manual*, for more information on various minor modes and how to enable them.

You can also display time in the mode line. See section "The Mode Line" in *XEmacs User's Manual*, for more information regarding the mode line.

1.5 The Echo Area

The line at the bottom of the frame (below the mode line) is the *echo area*. Emacs uses this area to communicate with you:

- The *echo area* will print out the characters that you type. For example, if you choose the **Open...** option from the **File** menu you might get the following in the echo area:

  ```
  Find file: /usr/lib/x11/
  ```

  Now you need to give a file name to open, for example if the file name is '`myfile`', you will type '`myfile`' after '`/usr/lib/x11/`' and press the ⟨Return⟩ key. If you pause for more than a second while typing, you will see the characters that you type in the *echo area*.
- The *echo area* also prints error messages. For example, if you misspell '`usr`' and type '`/urs/lib/x11/myfile`' ⟨RETURN⟩ in the above example you might get an error message. Since Emacs will not be able to find the '`/urs`' directory, the *echo area* will say:

  ```
  error  Opening directory: no such file or directory, /urs/lib/x11/myfile
  ```

  This error message will be accompanied by a beep. Some XEmacs commands will print informative messages in the *echo area*. See section "The Echo Area" in *XEmacs User's Manual*, for more information on the *echo area*.

2 XEmacs Windows and Menus

The first section of this chapter will show you how you can manipulate XEmacs Windows and the other section will explain the Pull-down Menus of an XEmacs window.

2.1 XEmacs Windows

When you use XEmacs under X, you can open multiple windows and each window can display one buffer or multiple parts of one buffer. Each window will have its own *mode line* and *echo area*. At any one time there is only one *selected window* and the buffer it displays is the *selected buffer*. There are some commands for manipulating windows:

M-C-v
: This command will scroll the window which is not *selected* (`scroll-other-window`).

C-x 0
: This command will get rid of the selected window (`delete-window`). That is a zero. If there is more than one Emacs frame, deleting the sole remaining window on that frame deletes the frame as well. If the current frame is the only frame, it is not deleted.

C-x 1
: This command will get rid of all the windows except the selected one. (`delete-other-windows`). For example, if you use the **Describe variable** option from the **Help** menu, the window will split vertically and the bottom window will contain documentation for that variable. After you are done looking at that variable's documentation you might want to come back to your original single window. Just type *C-x 1* after your cursor is in the top window (the window which you want to keep) and hit ⟨RET⟩.

C-x 2
: This command will split the selected window into two windows, one above the other (`split-window-vertically`). Both the windows will start out by displaying the same buffer. The window in which you have your cursor will be your *selected window*.

C-x 3
: This will split the selected window into two windows positioned side by side (`split-window-horizontally`). A line of vertical bars will separate the window.

You can select a buffer in another window by using some other commands. These commands all have a prefix key *C-x 4*

C-x 4 b bufname ⟨RET⟩
: This command will select a buffer *bufname* in another window. This runs `switch-to-buffer-other-window`. It will prompt you for a buffername.

C-x 4 f filename ⟨RET⟩
: Visit file *filename* and select its buffer in another window. This runs `find-file-other-window`. See section "Visiting" in *XEmacs User's Manual*. It will prompt you for a filename.

C-x 4 d directory ⟨RET⟩
: Select a Dired buffer for directory *directory* in another window. This runs `dired-other-window`. See section "Dired" in *XEmacs User's Manual*.

C-x 4 m Start composing a mail message in another window. This runs `mail-other-window`, and its same-window version is *C-x m*. See section "Sending Mail" in *XEmacs User's Manual*, for information on how to **S**end **M**ail using XEmacs. See section "Reading Mail With Rmail" in *XEmacs User's Manual*, for information on reading mail using **Rmail**.

If you click the right button on the mouse on a mode line, you will get a menu with following options:

Delete Window
: Choosing this menu will remove the window above this modeline from the frame.

Delete Other Windows
: Delete all windows on the frame except for the one above this modeline.

Split Window
: Split the window above the mode line in half, creating another window.

Split Window Horizontally
: Split the window above the mode line in half horizontally, so that there will be two windows side-by-side.

Balance Windows
: Readjust the sizes of all windows on the frame until all windows have roughly the same number of lines.

2.2 XEmacs Pull-down Menus

When you run XEmacs under X, each Emacs frame has a menu-bar at the top which provides commands for editing, help and other options. All these options are also available via key commands, the menus just provide convenient short-cuts. The key commands are displayed right besides some of the options. The following is a brief description of the four default menus on the menu bar:

2.2.1 The File Menu

The **File** menu bar contains the following items. To choose a particular option, press the left mouse button and drag it to the item you wish to select. Then release the button.

Open... This option will prompt you for a file name. You will get a message in the echo area:

```
Find File:
```

After Find File, there might be a directory path also. After you type the file name and press ⟨RET⟩ the file will be loaded into a new buffer.

Open in New Frame...
: It prompts you for a file name and loads that file in a new buffer in a new frame. You can open many frames for the same Emacs session. You can delete the frame by selecting **Delete Frame**.

Insert File...
: Prompts you for a filename and inserts the contents of this filename in your current buffer. Position your cursor at the place you wish to insert the file and select this option. You will get the following message in the echo area:

```
Insert file:
```

Insert the file name and press ⟨RET⟩.

Save <Buffername>
: It saves the changes you have made to the buffer. If you have made changes which are not saved yet, the option will appear dark, otherwise it will be light and unselectable. If you do not wish to save the changes, select **Revert Buffer**.

Save As...
: Prompts you for a filename and saves the current buffer in that file. It loads the new file if the filename you specify is different from the one you were working with.

Print Buffer <buffername>
: Prints a hardcopy of the current or *selected* buffer.

New Frame
: Opens a new frame with ***scratch*** as the default buffer. It doesn't prompt you for a filename. To open a file you need to go to that frame and select **Open...**

Split Frame
: Splits the current window into two equal-sized windows with the same buffer. To get back a single frame, select **Un-Split (Keep This)**. See Section 2.1 [XEmacs Window], page 9, for more information about windows.

Un-Split (Keep This)
: If the frame contains multiple windows, it will remove all windows except the selected one.

Un-Split (Keep Others)
: If the frame contains multiple windows, it will remove the selected window and keep the other one.

Revert Buffer <buffername>
: If you do not wish to save the changes you made to the file since you opened it, select this option. It will restore the last saved version of the file to the current buffer.

Kill Buffer <buffername>
: It will kill the current buffer. If will prompt you if there are unsaved changes.

Exit Emacs
: It will kill the Emacs *process* as opposed to simply killing the *buffer*. Before it kills the process, it will prompt you as to which unsaved buffers you wish to save by going through the list of the buffers.

2.2.2 The Edit Menu

Most of the commands in this menu work on a block of text or a selected region. The text will be highlighted as you select it.

Undo Undoes the previous command. If you type something by mistake you can use this command. For example, if you select **Insert File...** from the **File** menu and insert a wrong file by mistake, you can select this item and it will remove the inserted file. It undoes a batch of text which is worth an emacs command.

Cut Removes the selected text block from the current buffer, makes it the X clipboard selection, and places it in the kill ring (see Section 9.4 [Moving Text], page 42). Before executing this command, you have to select a region using Emacs region selection commands or with the mouse. See Section 9.1 [Selecting Text], page 41.

Copy Makes a selected text block the X clipboard selection, and places it in the kill ring. You can select text using one of the Emacs region selection commands or by selecting a text region with the mouse. See Section 9.1 [Selecting Text], page 41, for more information.

Paste Inserts the current value of the X clipboard selection in the current buffer. Note that this is not necessarily the same as the Emacs `yank` command, because the Emacs kill ring and the X clipboard selection are not the same thing. You can paste in text you have placed in the clipboard using **Copy** or **Cut**. You can also use **Paste** to insert text that was pasted into the clipboard from other applications. See section "X Clipboard Selection" in *XEmacs User's Manual*, for information on using Clipboard Selection.

Clear Removes the selected text block from the current buffer but does not place it in the kill ring or the X clipboard selection. You will not be able to get this text back.

Start Macro Recording
After selecting this, Emacs will remember every keystroke you type until **End Macro Recording** is selected.

End Macro Recording
Selecting this tells emacs to stop remembering your keystrokes.

Execute Last Macro
Selecting this item will cause emacs to re-interpret all of the keystrokes which were saved between selections of the **Start Macro Recording** and **End Macro Recording** menu items. You can now execute the most recent keyboard macro. See section "Keyboard Macros" in *XEmacs User's Manual*, for further information.

2.2.3 The Options Menu

There are sub-menus for some of the menus which you will need to select. If sub-menus exist for an item, they will be displayed automatically when you drag the mouse on that item. The items in this menu provide some fancy editing operations.

Read Only
Selecting this item will cause the buffer to visit the file in a read-only mode. Changes to the file will not be allowed.

Case Sensitive Search
: Selecting this item will cause searches to be case-sensitive. If its not selected then searches will ignore case. This option is local to the buffer. For example, if this item is selected and you are searching for '`Smile`', then an occurrence of '`smile`' will not be recognized because of the smaller case of '`s`'.

Overstrike
: After selecting this item, when you type letters they will replace existing text on a one-to-one basis, rather than pushing it to the right. At the end of a line, such characters extend the line. Before a tab, such characters insert until the tab is filled in.

Auto Delete Selection
: Selecting this item will cause automatic deletion of the selected region. After you select a region and hit the ⟨RET⟩ key, the selected text will be deleted. The typed text will replace the selection if the selection is active (i.e. if its highlighted). If the option is not selected then the typed text is just inserted at the cursor.

Teach Extended Commands
: After you select this item, any time you execute a command with *`M-x`* which has a shorter keybinding, you will be shown the alternate binding before the command executes. For example if you type *`M-x find-file-other-window`* which performs the same function as the **Open in Other Window...** in **File** menu you will see the following message:

```
M-x find-file-other-window (bound to keys: C-x 4 f, C-x 4 C-f)
```

Syntax Highlighting
: You can customize your `.emacs` file to include the font-lock mode so that when you select this item, the comments will be displayed in one face, strings in another, reserved words in another, and so on. See section "Customization" in *XEmacs User's Manual*, for more information on customizing `.emacs` file. After selecting this item, you will find your code a lot easier to read. When **Fonts** is selected, different parts of the program will appear in different Fonts. When **Colors** is selected, then the program will be displayed in different colors. Selecting **None** causes the program to appear in just one Font and Color. Selecting **Less** resets the Fonts and Colors to a fast, minimal set of decorations. Selecting **More** resets the Fonts and Colors to a larger set of decorations. For example, if **Less** is selected (which is the default setting) then you might have all comments in green color. It does not matter what the comments contain. Whereas, if **More** is selected then a function name in the comments themselves might appear in a different Color or Font. Even though the comments themselves might appear in green color, a function name *within* the comments might appear in red color.

Paren Highlighting
: After selecting **Blink** from this item, if you place the cursor on a parenthesis, the matching parenthesis will blink. If you select **Highlight** and place the cursor on a parenthesis, the whole expression of the parenthesis under the cursor

will be highlighted. Selecting **None** will turn off the options (regarding **Paren Highlighting**) which you had selected earlier.

Font You can select any Font for your program by choosing from one of the available Fonts. The whole buffer will be converted to the Font you select.

Size You can select any size for the text in your buffer (ranging from **2** to **24**) by selecting the appropriate option.

Weight You can choose either **Bold** or **Medium** for the weight of the text of your buffer.

Buffers Menu Length...
Prompts you for the number of buffers to display. Then it will display that number of most recently selected buffers.

Buffers Sub-Menus
After selection of this item the Buffers menu will contain several commands, as submenus of each buffer line. If this item is unselected, then there are no submenus for each buffer line, the only command available will be selecting that buffer.

Save Options
Selecting this item will save the current settings of your Options menu to your `.emacs` file so that the next time you start XEmacs, you won't need to select the options again.

2.2.4 The Buffers Menu

The **Buffers** menu provides a selection of up to ten buffers and the item **List All Buffers**, which provides a Buffer List. If you select **Buffers Sub-menus** from the **Options** menu, you will get some sub-menus for each of the buffer listing.

2.2.5 The Help Menu

The Help Menu gives you access to Emacs Info and provides a menu equivalent for some of the choices you have when using `C-h`. See Chapter 5 [Help], page 23, for more information.

The **Describe variable** and **Describe function** will provide documentation for the corresponding variable or function. The Help menu also gives access to UNIX online manual pages via the **UNIX Manual...** option.

3 Basic Editing Commands

This chapter will introduce you to some basic editing commands. You can also learn the basic editing commands by typing *Control-h t* (help-with-tutorial OR by selecting **Emacs Tutorial** from the **Help** menu on the menu bar. Most of the Emacs commands will use the ⟨CONTROL⟩ key or the ⟨META⟩ key. The following abbreviations will be used for the ⟨CONTROL⟩ and ⟨META⟩ key in this manual:

C-<chr> This means that you should hold down the ⟨CONTROL⟩ key while typing *<chr>*. For example, if the command is *C-g*, you should hold the ⟨CONTROL⟩ key and type ⟨g⟩.

M-<chr> This means that you should hold down the *META* key while typing *<chr>*. If there is no *META* key on your keyboard, use the *ESC* key instead. For example, if the command is *M-x*, then type *ESC*, release it and type *x*.

The following abbreviations will be used for some other keys:

⟨SPC⟩ Space bar.

⟨RET⟩ Return key.

⟨LFD⟩ Linefeed key.

⟨TAB⟩ Tab.

⟨ESC⟩ Escape.

⟨SFT⟩ Shift.

3.1 Inserting Text

To insert printing characters into the text you are editing, just type them. Emacs will automatically insert the characters that you type into the buffer at the cursor. The cursor moves forward, but if you prefer to have text characters replace (overwrite) existing text characters, you can enable the **Overstrike** option from the **Options** menu in the menu bar.

To *delete* text you have just inserted, use ⟨DEL⟩. ⟨DEL⟩ deletes the character *before* the cursor (not the one that the cursor is on top of or under; that is the character *after* the cursor). The cursor and all characters after it move backwards. Therefore, if you type a printing character and then type ⟨DEL⟩, they cancel out.

To end a line and start typing a new one, type ⟨RET⟩. This inserts a newline character in the buffer. If point is in the middle of a line, ⟨RET⟩ splits the line. Typing ⟨DEL⟩ when the cursor is at the beginning of a line rubs out the newline before the line, thus joining the line with the preceding line.

Emacs automatically splits lines when they become too long, if you turn on a special mode called *Auto Fill* mode. See section "Filling" in *XEmacs User's Manual*, for information on using Auto Fill mode.

3.2 Moving Around

The following commands will allow you to move the cursor around the screen. The actual function names corresponding to the these commands are given in parenthesis. You can also invoke these commands by typing *M-x <function name>*. You can do this for any command in XEmacs.

C-b Move the cursor backward one character (`backward-char`).

C-f Move the cursor forward one character (`forward-char`).

C-p Move the cursor up one line vertically (`previous-line`).

C-n Move the cursor down one line vertically (`next-line`).

C-a Move the cursor to the beginning of the line (`beginning-of-line`).

C-e Move the cursor to the end of the line (`end-of-line`).

M-f Move the cursor forward one word (`forward-word`).

M-b Move the cursor backword one word (`backward-word`).

M-< Move the cursor to the top of the buffer (`beginning-of-buffer`).

M-> Move the cursor to the end of the buffer (`end-of-buffer`).

M-x goto-char RET <number> RET

To enable this command type *M-x goto-char*, and hit ⟨RETURN⟩ key. In the *echo area* you will see:

```
Goto char:
```

You should then type in a number right after the colon and hit the *RETURN* key again. After reading a number n this command will move the cursor to character number n. Position 1 is the beginning of the buffer. For example, if you type *M-x goto-char RET 200 RET*, then the cursor will move to the 200th character starting from the beginning of the buffer.

M-x goto-line RET <number> RET

To enable this command type *M-x goto-line*, and hit the ⟨RETURN⟩ key. After you see *Goto line:* in the *echo area*, type in a number n and hit ⟨RETURN⟩ key again. This command will position the cursor on the nth line starting from the beginning of the buffer.

M-x what-line RET

This command will display the current line number in the echo area.

3.3 Erasing Text

⟨DEL⟩ If you press ⟨DEL⟩ i.e. the *delete* key, it will delete the character before the cursor (`delete-backward-char`).

C-d This will delete the character after the cursor (`delete-char`).

C-k Kill to the end of the line (`kill-line`). If you kill the line by mistake you can *yank* or '`paste`' it back by typing *C-y*. See Section 9.4 [Moving Text], page 42, for more information on yanking.

M-d Kill forward to the end of the next word (`kill-word`).

M-⟨DEL⟩ Kill back to the beginning of the previous word (`backward-kill-word`).

M-k Kill to the end of current sentence (`kill-sentence`).

M-z char Kill up to next occurrence of *char* (`zap-to-char`). To use this command type *M-z*. You will see the following statement in the echo area :

```
Zap to char:
```

Type any char and press the ⟨RET⟩ key. For example, if you type '`p`' then the entire text starting from the position of the cursor until the first occurrence of '`p`' is killed.

3.4 Giving Numeric Arguments

Any Emacs command can be given a *numeric argument*. Some commands interpret the argument as a repetition count. For example, if you want to move forward ten characters, you could type *C-f* ten times. However, a more efficient way to do this would be to give an argument of ten to the key *C-f* (the command `forward-char`, move forward one character). Negative arguments are also allowed. Often they tell a command to move or act backwards. For example, if you want to move down ten lines, type the following:

```
C-u 10 C-n RET
```

After you press ⟨RET⟩ key, the cursor will move ten lines downward. You can also type:

```
M-10 C-n RET
```

Both *C-u* and *M-* allow you to give numeric arguments. If you want to move ten lines backward, you can also give negative arguments, like:

```
C-u -10 C-n RET
```

OR you could also type:

```
M--10 C-n RET
```

You can obviously use *C-b* to move backward rather than giving negative arguments to *C-n*. See section "Numeric Arguments" in *XEmacs User's Manual*, for more information on numeric arguments.

3.5 Undoing Changes

When you are editing a buffer, you might type something by mistake. Emacs allows you to undo all changes you make to a buffer (but not more than 8000 characters). Each buffer in Emacs keeps a record of the changes made to it individually, so the undo command applies to the current buffer. There are two undo commands:

C-x u Undo one batch of changes (usually, one command's worth). (`undo`).

C-_ The same as above, but this command might not be obvious to type on some keyboards so it might be better to use the above command.

See section “Undoing Changes” in *XEmacs User’s Manual*, for more information on undoing changes.

4 Customize key bindings and menus

When you start Emacs, it reads the file '~/.emacs' in your home directory. You can use this file to initialize and customize Emacs to your liking. This file should contain lisp-code. You can customize your '.emacs' file to create new menus, disable menus, change key bindings, enable a minor mode, etc. Any kind of customization affects only a particular Emacs job that you do them in. If you want to save your customizations 'permanently' i.e. for future use also, you have to put it in your '.emacs' file. After you make changes to your '.emacs' file and save it, the changes will be effective only after you start Emacs again i.e. for a new Emacs process. To try out some of the examples in this section, highlight that region and evaluate the region by giving the command *M-x eval-region*. You will be able to see the results of your customizations in that Emacs session only (see section "Lisp Eval" in *XEmacs User's Manual*).

4.1 Customize key bindings

Most of Emacs commands use key sequences. See section "Keystrokes" in *XEmacs User's Manual*, for more information about Keys and Commands. In Emacs, the keys themselves carry no meaning unless they are bound to a function. For example, *C-n* moves the cursor to the next line because its bound to the function **next-line**. Similarly, *C-p* moves to the previous line because its bound to the function **previous-line**. The functions themselves define a particular behavior. You can customize the key *C-n* to move to the previous line by binding it to **previous-line** and *C-p* to move to the next line by binding it to **next-line**. To bind keys to globally run commands you need to use the following syntax in your **.emacs** file:

```
(global-set-key keys cmd)
```

Here, `global-set-key` is a function which will bind the *keys* to the specified *cmd*. For example, if you type the following in your **.emacs** file:

```
(global-set-key "\C-p" 'next-line)
(global-set-key "\C-n" 'previous-line)
```

then *C-p* will move to the next line and *C-n* to the previous line.

You can also disable a key binding, by using 'nil' as the *cmd* in the syntax stated above. Here, 'nil' stands for 'false' which means disable a command or turn off a feature. If you want to enable a command or turn on a particular feature use 't' which stands for 'true'. For example, if you do not wish *C-x C-c* to 'Exit Emacs' you can type the following expression in your '.emacs' file:

```
(global-set-key "\C-x\C-c" nil)
```

You might want to have this statement in your '.emacs' file because its easy to hit this command by mistake and it could be annoying to exit Emacs unintentionally. There is a **Exit Emacs** option in the **File menu** which you might want to use instead. To make a particular key undefined you can also use:

```
(global-unset-key "\C-x\C-c")
```

Now if you use the command *C-x C-c*, you will get an error saying that the command is undefined.

Some other customizations you could try are:

- ```
  (global-set-key 'button3 'beginning-of-buffer)
  ```
  Now when you press the third button of your mouse, the cursor will be placed at the `beginning-of-buffer`.
- ```
  (global-set-key 'f1 'goto-line)
  ```
  If you press the ⟨F1⟩ key, you will be prompted for a line number. After you type the line number and hit ⟨RET⟩, the cursor will be placed on that line number.
- ```
  (global-set-key 'f2 'undo)
  ```
  Pressing ⟨F2⟩ will undo the last command. If you have a ⟨undo⟩ key on your keyboard, try binding that key to the undo command.

Another syntax for customizing key bindings is: `(define-key` *keymap keys def*`)` It defines *keys* to run *def* in the keymap *keymap*.

keymap is a keymap object which records the bindings of keys to the commands that they run.

keys is the sequence of keystrokes to bind.

def is anything that can be a key's definition:

Look at the following two examples:

```
(define-key global-map "\C-xl" 'make-symbolic-link)
(define-key c-mode-map "\C-xl" 'make-symbolic-link)
```

Both the examples bind the key *`C-xl`* to run the function `make-symbolic-link` (see section "Misc File Ops" in *XEmacs User's Manual*). However, the second example will bind the key only for C mode. See section "Major Modes" in *XEmacs User's Manual*, for more information on Major Modes in XEmacs.

4.2 Customizing Menus

You can customize any of the XEmacs Pull-down-Menus. You can create your own menu, delete an existing one, enable a menu or disable a menu. For more information on the default menus available to you, See Section 2.2 [Pull-down Menus], page 10.

Some of the functions which are available to you for customization are:

1. add-menu-item: *(menu-name item-name function enabled-p &optional before)*

   This function will add a menu item to a menu, creating the menu first if necessary. If the named item already exists, the menu will remain unchanged. For example, if you add the following example to your '`.emacs`' file or evaluate it (see Chapter 4 [Customization Basics], page 19),

   ```
   (add-menu-item '("Edit") "Replace String" replace-string t "Clear")
   ```

   a sub-menu **Replace String** will be created under **Edit** menu before the sub-menu **Clear**. The **Edit** menu will now look like:
   ```

```
Undo                    C-x u
Cut                     cut
Copy                    copy
Paste                   paste
Replace String
Clear
Start Macro Recording   C-x(
End Macro Recording     C-x)
Execute Last Macro      C-xe
```

Replace String will now execute the function `replace-string`. Select this menu item. Emacs will prompt you for a string name to be replaced. Type a string and hit ⟨RET⟩. Now type a new string to replace the old string and hit ⟨RET⟩. All occurrences of the old string will be replaced by the new string. In this example,

'`Edit`' is the *menu-name* which identifies the menu into which the new menu item should be inserted.

'`Replace String`' is the *item-name* which names the menu item to be added.

'`replace-string`' is the *function* i.e. the command to be invoked when the menu item "Replace String" is selected.

'`t`' is the *enabled-p* parameter which controls whether the menu item is selectable or not. This parameter can be either `t` (selectable), `nil` (not selectable), or a form to evaluate. This form is evaluated just before the menu is displayed, and the menu item will be selectable if the form returns non-`nil`.

'`Clear`' is the *&optional before* parameter which is the name of the menu before which the new menu or sub-menu should be added. The *&optional* string means that this parameter is optional. You do not need to specify this parameter. If you do not specify this parameter in the example above, the **Replace String** menu item will be added at the end of the list of sub-menus in the **Edit** menu i.e. after **Execute Last Macro**.

If you wish to add a new menu to the menubar, try:

```
(add-menu-item nil "Bot" 'end-of-buffer t)
```

This will create a new menu **Bot** on the menu bar. Selecting this menu will take you to the end of the buffer. Using `nil` for the parameter *menu-name* will create a new menu. Your menu-bar will now look like:

```
File Edit Options Buffers Bot                         Help
```

The following example will illustrate how you can add sub-menus to the submenus themselves:

```
(add-menu-item '("File" "Management") "Copy File" 'copy-file t)
(add-menu-item '("File" "Management") "Delete File" 'delete-file t)
(add-menu-item '("File" "Management") "Rename File" 'rename-file t)
```

This will create a sub-menu **Management** under the **File** menu. When you select the submenu **Management**, it will contain three submenus: **Copy File**, **Delete File** and **Rename File**.

2. delete-menu-item: *(menu-path)* This function will remove the menu item defined by *menu-name* from the menu hierarchy. Look at the following examples and the comments just above them which specify what the examples do.

```
;; deletes the "Replace String" menu item created earlier
(delete-menu-item '("Edit" "Replace String"))

;; deletes the "Bot" menu created earlier
(delete-menu-item '("Bot"))

;; deletes the sub-menu "Copy File" created earlier
(delete-menu-item '("File" "File Management" "Copy File"))

;; deletes the sub-menu "Delete File" created earlier
(delete-menu-item '("File" "Management" "Delete File"))

;; deletes the sub-menu "Rename File" created earlier
(delete-menu-item '("File" "Management" "Rename File"))
```

3. disable-menu-item: *(menu-name)* Disables the specified menu item. The following example

   ```
   (disable-menu-item '("File" "Management" "Copy File"))
   ```

   will make the **Copy File** item unselectable. This menu-item would still be there but it will appear faded which would mean that it cannot be selected.
4. enable-menu-item: *(menu-name)* Enables the specified previously disabled menu item.

   ```
   (enable-menu-item '("File" "Management" "Copy File"))
   ```

   This will enable the sub-menu **Copy File**, which was disabled by the earlier command.
5. relabel-menu-item: *(menu-name new-name)* Change the string of the menu item specified by *menu-name* to *new-name*.

   ```
   (relabel-menu-item '("File" "Open...") "Open File")
   ```

   This example will rename the **Open...** menu item from the **File** menu to **Open File**.

5 Help

XEmacs provides a comprehensive Help facility. On the extreme right of the menu-bar there is a **Help** menu. There are several help commands provided by this menu. You can also use `C-h` for invoking the Help facility. Type "?" for a list of keys you can type after typing `C-h`. If you want more information on what your options are and what kind of help you can get type "?" again. You will get a listing of all the keys you can type and what they will do. Initially if you want help, type `C-h` three times.

5.1 Help menu

When you click on the Help menu with any of the mouse buttons you will get the following menu items:

Info
: Selecting this item will take you to the Info page which is the online documentation browsing system. You can simply click on the highlighted items and "Info" will take you to the document providing information about that topic.

Describe Mode
: After you select this item, you will get a documentation on the major and minor modes which are enabled in the buffer you are working with. See Chapter 6 [Modes], page 27, for information on Modes.

Hyper Apropos...
: After you select this item, you will see the following message in the echo area:

```
List symbols matching regexp:
```

If you type "mode" and hit ⟨RET⟩, you will get a list of all the symbols (like functions and commands). You can now get documentation on any of the given symbols by "clicking" on any of the symbols (i.e. drag your mouse on the appropriate symbol and release the button). For example, if you "click" on the 'auto-fill-mode' you will get the following message in the window at the bottom:

```
auto-fill-mode

Function, Command:

  Toggle auto-fill mode.
  With arg, turn auto-fill mode on if and only if arg is positive.
  In auto-fill mode, inserting a space at a column beyond `fill-column'
  automatically breaks the line at a previous space.

Variable:

  value = nil

  variable not documented
```

Command Apropos...
: Selecting this item will prompt you for a string just like when you select **Hyper Apropos...**. After you give a string name, you will get a listing of all the func-

tions and commands containing that string name with a very short description about what that command does.

Full Apropos...
After you select this item, you will be prompted for a string name in the echo area:

```
Apropos (regexp):
```

Now you can give any string name, for example "mode" and hit ⟨RET⟩. You will get a listing of all the variables and commands containing that string i.e "mode" with a short description of its function.

List Keybindings
Select this item and you will get a listing of all the keys and the commands that they execute. Depending on which Major mode your buffer is in, you will get a listing of the special keybindings for that particular buffer also. For example, if you are in "Texinfo" mode, part of your list will contain:

```
C-c C-c n texinfo-insert-@node
C-c C-c o texinfo-insert-@noindent
C-c C-c s texinfo-insert-@samp
C-c C-c t texinfo-insert-@table
C-c C-c v texinfo-insert-@var
C-c C-c x texinfo-insert-@example
C-c C-c { texinfo-insert-braces
```

These keybindings apply only to "Texinfo" mode. See Chapter 6 [Modes], page 27, for more information on various modes.

Describe Key...
After you select this item, you will be see the following message in the echo area:

```
Describe Key:
```

After you type a command key sequence, full documentation of that command will be displayed. For example if you type *C-g*, you will see the following documentation for *C-g*:

```
keyboard-quit:
Signal a 'quit' condition.
```

This means that *C-g* will quit whatever command you gave earlier.

Describe Function...
This menu item provides documentation for a function. After you select this item, it will prompt you for a function name in the echo area:

```
Describe function (default <some function name>):
```

If you hit ⟨RET⟩ without giving a function name, you will get documentation for that default function name, otherwise if you type a function name and hit ⟨RET⟩, you will get documentation for the given function.

Describe Variable...
You can get documentation on any variable by selecting this menu item. It is similar to **Describe Function** and will prompt you for a variable name.

Unix Manual...
After you select this item you will be prompted for a Unix command for which you wish to see the man page. You will see the following message in the echo area:

```
Manual entry: (default <some name>)
```

Now you can type any command, for example type '`who`' and press ⟨RET⟩. You will get the man page for the Unix command '`who`' which lists who is on the system.

Emacs Tutorial
Select this item and you will get a tutorial on Emacs. It is good for new users.

Emacs News
Select this item and you will get a lot of historical and current news on Emacs !

For more information on the Help facility, See section "Help" in *XEmacs User's Manual*.

6 Major and Minor Modes

XEmacs is *language sensitive*. It has several *major* and *minor* modes. The major modes customize Emacs to edit text of a particular sort. There are major modes for C, Lisp, Emacs Lisp, LaTeX, English etc. Within each major mode, certain functions and keys are redefined to "suit" that particular sort of text. The minor modes provide certain features which can be turned off or on at any time. Emacs can only be in one major mode at any time, but it can turn on several minor modes at the same time. After you have selected any major or minor mode, you can select **Describe Mode** from the **Help** menu and you will get documentation about those modes.

6.1 Major Modes

Emacs has several major modes which customize Emacs to edit text of various sorts. You can have only one major mode at any time. Within each major mode, Emacs redefines certain functions (like cursor movement, indentation and text killing) to suit the needs of the text being edited. When you are editing a specific type of text you should switch to the appropriate mode. If you are working with C code, you should switch to C mode; if you are working with Lisp code, then switch to lisp mode and if you are working with English text switch to Text mode.

When you open a file to work on, Emacs usually selects the appropriate mode. For example, if you open a file called '`guide.c`' then Emacs will select the C mode because of the ".c" extension of the file. To explicitly select a mode type the following command:

```
;;; selects lisp mode
M-x lisp-mode

;;; selects C mode
M-x c-mode
```

To select any other mode, just add the major mode name before the '-mode'. The current mode in which you are in will be displayed in parenthesis in the mode-line at the bottom of the frame. All major modes have some special keybindings and you can get a listing of those keybindings by selecting **List Keybindings** from the **Help** menu on the menu bar.

Some of the available modes in XEmacs are :

fundamental-mode
: When you start XEmacs, usually you start with the default "Fundamental" mode. This mode has no special definitions or settings.

nroff-mode
: Use this mode when you have to format a text with nroff before it can be available in readable form. It redefines some indentation commands. See section "Nroff Mode" in *XEmacs User's Manual*, for information on this mode.

tex-mode
: Use this mode if you are using the LaTeX text-formatter. It provides commands for insertion of quotes, braces and other characters. It also allows you to format the buffer for printing. See section "TeX Mode" in *XEmacs User's Manual*, for information on this mode.

texinfo-mode
Texinfo is a documentation system that uses a single source file to produce both printed output and on-line documentation. When you use this mode, there will be some special keybindings for inserting some characters and executing some commands.

This manual itself is produced by 'Texinfo'

outline-mode
Use this mode for editing outlines. When you enable this mode, you can make part of the text temporarily invisible so that you can see the overall structure of the outline. See section "Outline Mode" in *XEmacs User's Manual*, for information on this mode.

c-mode Use this mode for C programs. It will redefine some indentation commands. See section "C Indent" in *XEmacs User's Manual*.

lisp-mode Use this mode for Lisp programs. Look at the XEmacs User's Manual for more information.

fortran-mode
Use this mode for Fortran programs. This mode provides special commands to move around and some other indentation commands. For more information on this mode, See section "Fortran" in *XEmacs User's Manual*.

edit-picture
This is the picture mode which you can use to create a picture out of text characters. See section "Picture" in *XEmacs User's Manual*, for more information.

asm-mode Use asm-mode for editing files of assembler code. Look at the file '`/usr/local/lib/xemacs-19.11/lisp/modes/asm.el`' for more information.

There are some other modes and commands for working with other kinds of text or programs. Emacs also provides commands for reading and sending Mail. For more information on these features look at the XEmacs Manual. Emacs also provides the functions of a desk calendar, with a diary of past or planned events. For more information on the calendar mode look at the manual for Calendar Mode and Diary.

6.2 Minor Modes

The minor modes in Emacs provide some optional features which you can turn on or off. Any number of minor modes can be active at the same time with any major mode. You can enable a minor mode in one buffer and disable it in other mode. To enable a minor mode, for example the font-lock mode type the following command:

```
M-x font-lock-mode
```

To enable the other minor modes, replace the "font-lock" with the name of the minor mode. To disable the mode type the command again. A positive argument will always turn the mode on. Whenever you type this command, it will turn the mode on if it was off, OR it will turn it off if it was on i.e. it toggles. Look at the mode-line at the bottom of the frame. If it says FLock in parentheses, then it means that this mode is on, otherwise it is off.

The following are some of the minor modes available in XEmacs. To enable any one of them type "M-x" in front of them.

font-lock-mode

You can also choose this mode by selecting the **Syntax Highlighting** menu item from the **Options** menu on the menu-bar at the top. If you wish to have this mode enabled permanently, choose **Save Options** from the **Options** menu. See Section 2.2.3 [Options Menu], page 12, for more information on the Options menu. You can also add statements in your '`.emacs`' file. For each major mode in which you wish to enable this minor mode, you need a statement in your '`.emacs`' file. The following example shows how to enable the font-lock mode when the major mode is c-mode.

```
(add-hook 'c-mode-hook 'turn-on-font-lock)
```

See Chapter 8 [Other Customizations], page 35.

When you enable this mode, the text will be displayed in different colors and fonts depending on the type of the text. This makes the text very easy to read and understand. For example, comments might be displayed in red, variables in black, functions in blue and other keywords in different colors and fonts. When you select **More** from the **Syntax Highlighting** option, you get very detailed display of colors and fonts; function names within comments themselves might appear in a different font and color.

auto-fill-mode

Enabling this mode will provide automatic word-wrapping. The ⟨SPC⟩ key will break lines i.e. insert newlines as you type to prevent lines from becoming too long.

overwrite-mode

When you enable this mode, the text that you type will replace the existing text rather than moving it to the right (the default case). You can enable this mode by selecting **Overstrike** menu-item from the **Options** menu from the menu-bar.

abbrev-mode

After you enable this mode, you can define words which will expand into some different text i.e. you can define abbreviations. For example, you might define "expand" to "expand will eventually expand to this text". After this definition you will be able to get "expand will eventually expand to this text" simply by typing

```
expand ⟨SPC⟩
```

See section "Abbrevs" in *XEmacs User's Manual*, for more information on this mode and on defining abbreviations.

auto-save-mode

After you enable this mode in a buffer, the contents of that buffer will be saved periodically. This will reduce the amount you might lose in case of a system crash.

line-number-mode

After you enable this mode, the line number at which your cursor is present will be displayed continously in the mode line.

blink-paren

To enable this command, just type

```
M-x blink-paren
```

Do not add the "-mode" to it. You can also select the **Paren Highlighting** option from the **Options** menu. After you enable this command, put your cursor on one of the left parenthesis. The other matching parenthesis will start blinking. See Section 2.2.3 [Options Menu], page 12, for more information on the **Paren Highlighting** option.

For information on some other modes, look at the XEmacs User's Manual and the associated files.

7 Files

The basic unit of stored data in Unix is the *file*. To edit a file, you must tell Emacs to read the file into a buffer. This is called *visiting* the file. You can now edit the buffer and to save the changes you must write the buffer back to the file.

In addition to visiting and saving files, Emacs can delete, copy, rename, and append to files, and operate on file directories.

7.1 File Names

Most of the Emacs commands that operate on a file require you to specify a file name. For example, you might specify the file name initially when you enter Emacs :

```
xemacs myfile RET
```

After you hit ⟨RET⟩, you will enter XEmacs with "myfile" read into the current buffer. If you do not specify the filename when entering Emacs, you can use the **Open...** option from the **File** menu. You will be prompted for a filename in the echo area:

```
Find file: /usr/workspace/
```

Type in a file name which you want to open after the "/" and hit ⟨RET⟩. The specified file will be read into the current buffer. The "/usr/workspace" might be the *default directory*. When Emacs prompts you for a file, it uses the default-directory unless you specify a directory. You can see what the default directory of the current buffer is by using the **Describe Variable** option from the **Help** menu. When Emacs prompts you for the variable name to describe, type *default-directory*. If you wish to open a file in some other directory, use ⟨DEL⟩ or the ⟨BackSpace⟩ key to go back and type the path name of the new directory.

You can create a new directory by typing `M-x make-directory`. This command will prompt you for a directory name:

```
Create directory: /usr/workspace/
```

After you type a directory name and press ⟨RET⟩, a new directory with the specified name will be created. If you do not wish to create a new directory, then simply press `C-g` to quit the command. Similarly, you can also remove a directory by using the command `remove-directory`. The command `M-x pwd` will print the current buffer's default directory. For more information on file names, See section "File Names" in *XEmacs User's Manual*.

7.2 Visiting Files

To edit a file in Emacs you need to *visit* it. *Visiting* a file means copying its contents (or reading them) into the current buffer. Emacs will create a new buffer for each file that you visit. The buffer will be named after the file that you open. If you open a file '`/usr/workspace/myfile.texinfo`', the buffer will be called "myfile.texinfo". If a buffer with this name already exists, a unique name will be constructed by appending '`<2>`', '`<3>`', etc. If this is the second buffer with the same name, a "<2>" will be appended, "<3>" for a third buffer and so on. The name of the buffer which is being displayed in the window will be shown both at the top and bottom of the frame. Once you are in XEmacs, you can use the following commands:

C-x C-f This command will visit a file (`find-file`). It will prompt you for a file name to visit. The **Open...** option from the **File** menu does the same thing:

```
Find file: /usr/workspace/
```

Type in a filename and press ⟨RET⟩. You will see a new buffer on the screen with its name in the mode-line. If the filename you specify already exists in Emacs, the buffer containing that file will be selected. You will get an error message if the filename does not exist. If you still press ⟨RET⟩, a new buffer with the given filename will be displayed on the screen.

C-x C-v This command (`find-alternate-file`), will visit a different file instead of the one visited last. It is similar to *C-c C-f* except that it kills the current buffer (after offering to save it).

C-x 5 C-f This command will visit a file in another frame (`find-file-other-frame`) without changing the current window or frame. The **Open in New Frame...** from the **File** menu will do the same thing. It will prompt you for a file name in the echo area. After you type the file name and press ⟨RET⟩, the specified file will be read into a new buffer and displayed on a new frame.

7.3 Saving Files

The changes that you make after visiting a file will not be saved unless you save the buffer. When you save the buffer, Emacs writes the current contents of the buffer into the visited file. Some commands to save buffers are:

C-x C-s This command will permanently save the current buffer in its visited file (`save-buffer`). You will see the following message in the echo area if you save a file called "myfile.texinfo" :

```
Wrote /usr/workspace/myfile.texinfo
```

Try using this command twice. You will get the above message the first time you use this command, the second time you will get the following message:

```
(No changes need to be saved)
```

This message indicates that you haven't made any changes since the last time you saved the file.

C-x s This command will save all the buffers in their visited files (`save-some-buffers`). It will prompt you for typing yes or no:

```
Save file /usr/workspace/myfile.texinfo? (y or n)
```

You will get the above message for all the buffers. Type "y" if you want to save the buffer.

C-x C-w This command will prompt you for a file name and save the current buffer in that file. (`write-file`). You will see the following message in the echo area:

```
Write file: /usr/workspace/
```

After you type in a file name, press ⟨RET⟩. The buffer will be saved in a new file. You can make copies of a particular file using this command.

You can also undo all the changes made since the file was visited or saved by reading the text from the file again (called *reverting*). For more information on this option, See section "Reverting" in *XEmacs User's Manual*.

When you save a file in Emacs, it destroys its old contents. However, if you set the variable *make-backup-files* to non-*nil* i.e. '`t`', Emacs will create a *backup* file. Select the **Describe variable** option from the **Help** menu and look at the documentation for this variable. Its default value should be '`t`'. However, if its not then use *`M-x set-variable`* to set it to '`t`' (see Section 8.1 [Setting Variables], page 35). The backup file will contain the contents from the last time you visited the file. Emacs also provides options for creating numbered backups. For more information on backups, See section "Backup" in *XEmacs User's Manual*.

Emacs also saves all the files from time to time so that in case of a system crash you don't lose lot of your work. You will see the message '`Auto-saving...`' displayed in the echo area when the buffer is being saved automatically. The auto saved files are named by putting the character '`#`' in front and back. For example a file called "myfile.texinfo" would be named as '`#myfile.texinfo#`'. For information on controlling auto-saving and recovering data from auto-saving, See section "Auto Save Files" in *XEmacs User's Manual*.

Emacs provides protection from simultaneous editing which occurs if two users are visiting the same file and trying to save their changes. It will put a lock on a file which is being visited and modified. If any other user tries to modify that file, it will inform the user about the lock and provide some options. For more information on protection against simultaneous editing, See section "Interlocking" in *XEmacs User's Manual*.

8 Other Customizations

You can modify the behavior of Emacs in minor ways permanently by putting your changes in your '.emacs' file. This file contains Lisp function call expressions. Each of these expressions will consist of a function name followed by arguments, all surrounded by parentheses. For example, to turn on the auto-fill-mode (i.e. break lines automatically when they become too long) , put the following line in your '.emacs' file:

```
(add-hook 'text-mode-hook
        '(lambda() (auto-fill-mode 1)))
```

Emacs has a function named "turn-on-auto-fill" which is defined as "(lambda() (auto-fill-mode 1))". Therefore you can also write the above as:

```
(add-hook 'text-mode-hook 'turn-on-auto-fill)
```

Emacs provides a number of hooks for the sake of customization. The hook variables contain list of functions to be called with no arguments. To turn on the auto-fill-mode, add the appropriate hook as shown in the example above.

Similarly, to enable the "font-lock mode" which displays your program in different fonts and colors(see Chapter 6 [Modes], page 27), put the following in your '.emacs' file. The comments above the statement explain what the statements do.

```
;;; enables the font-lock-mode in Lisp Mode
(add-hook 'lisp-mode-hook    'turn-on-font-lock)

;;; enables the font-lock-mode in Texinfo Mode
(add-hook 'texinfo-mode-hook    'turn-on-font-lock)

;;; enables the font-lock mode in C Mode
(add-hook 'c-mode-hook 'turn-on-font-lock)
```

To turn on the font-lock mode in other Major Modes like emacs-lisp, just put the name of the mode with "-hook" appended to it as the middle parameter in the above examples. You can also select the color that the functions, comments or other keywords should be displayed in :

```
;;; the function names will now be displayed in blue color
(set-face-foreground 'font-lock-function-name-face "blue")

;;; the comments will be displayed in forest green
 (set-face-foreground 'font-lock-comment-face "forest green")
```

For other customizations regarding the font-lock face, look at the file '/usr/local/lib/xemacs-19.11/etc/sample.emacs'.

8.1 Other Customizations

In XEmacs, *variables* are used for internal record-keeping and customizations. There are some variables called "options" which you can use for customizations. To examine a variable use:

```
;;; print the value and documentation of the variable, use either of the
;;; following commands
C-h v
```

```
M-x describe variable
```

After you type any of the above commands, you will be prompted for a variable name in the *echo area*. Type in the name of the variable, for example, type *case-fold-search* ⟨RET⟩ Your window will split into two and you will see the following message in that window:

```
case-fold-search's value is t
This value is specific to the current buffer.

Documentation:
*Non-nil if searches should ignore case.
Automatically becomes buffer-local when set in any fashion.
```

Since this variable's value is 't' searches will ignore case. If you want case-sensitive-search (i.e. if you are searching for "Foo" and you do not want "foo" to be included in the search, you need to set this variable to "nil". In order to do that, use:

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

Emacs will prompt you for the variable which you wish to set. Type in "case-fold-search" and hit ⟨RET⟩. You will see the following message:

```
Set case-fold-search to value:
```

Type "nil" and hit ⟨RET⟩. Now if you again use `M-x describe variable` , you will see that the new value of case-fold-search will be "nil" and your searches will be case-sensitive. This will be effective only for that Emacs session. If you want to change the value of a variable permanently put the following statement in your '`.emacs`' file :

```
(setq case-fold-search nil)
```

This statement will make searches case-sensitive only in the current buffer which is the '`.emacs`' file. This will not be very useful. To make searches case-sensitive globally in all buffers, use:

```
(setq-default case-fold-search nil)
```

If you want to change the value of any other variable, use :

```
(setq <variable-name> <new value>)
```

"setq" will assign the "new value" to the "variable-name" .

If you want a list of the "options" i.e. the variables available for customization type:

```
;;; displays a buffer listing names, values and documentation of options
M-x list-options

;;; displays options and allows you to edit those list of options
M-x edit-options
```

Try these options. If you are using edit-options to edit a variable, just point at the variable you wish to edit and use one of the following commands:

1 Set the value of the variable to t (non-nil).

0 Set the value of the variable to nil.

n Move to the next variable.

p Move to the previous variable.

There are some other options available to make the value of a variable local to a buffer and then to switch to its global value. You can also have a *local variables list* in a file which specifies the values to use for certain Emacs variables when you edit that file. See section "Variables" in *XEmacs User's Manual*, for information on these options.

8.2 Init File Examples

For customizing Emacs, you need to put Lisp expressions in your '`.emacs`' file. The following are some useful Lisp expressions. If you find any of them useful, just type them in your '`.emacs`' file:

- The following expression will make ⟨TAB⟩ in C mode insert a real tab character if the cursor or point is in the middle of the line. Now hitting the ⟨TAB⟩ key will indent a line only if the cursor is at the left margin or in the line's indentation:

  ```
  (setq c-tab-always-indent nil)
  ```

  The value of the variable *c-tab-always-indent* is usually '`t`' for '`true`'. When this variable is true, then hitting the ⟨TAB⟩ key always indents the current line.
- This expression will turn on the *auto-fill-mode* when you are in text mode:

  ```
  (setq text-mode-hook 'turn-on-auto-fill)
  ```

  This mode will automatically break lines when you type a space so that the lines don't become too long. The length of the lines is controlled by the variable *fill-column*. You can set this variable to a value you wish. Look at the documentation for this variable to see its default value. To change the value to 75 for example, use:

  ```
  (setq-default fill-column 75)
  ```

  This will change the value of this variable globally.
- The following expression will enable the use of *eval-expression* without confirmation:

  ```
  (put 'eval-expression 'disabled nil)
  ```

  Now when you use *eval-expression*, it will print the value of the expression you specify in the *echo area* without confirming with you.
- This expression will remove the binding of *`C-x C-c`*, because its easy to hit this key by mistake and you will exit Emacs unintentionally. You can use the **Exit Emacs** option from the **File** menu to exit Emacs.

  ```
  (global-set-key "\C-x\C-c" nil)
  ```

  Now if you type *`C-x C-c`*, you won't exit Emacs.
- The following expression will make the ⟨BACKSPACE⟩ and the ⟨DEL⟩ key work in the same manner:

  ```
  (global-set-key 'backspace [delete])
  ```

- This expression will make searches case sensitive:

  ```
  (setq-default case-fold-search nil)
  ```

  If we use "setq" instead of "setq-default" then searches will be case-sensitive only in the current buffer's local value. In this case the buffer would be the '`.emacs`' file. Since this would not be too helpful and we want to have case-sensitive searches in all buffers, we have to use "setq-default".

- This expression will enable the font-lock mode when you are using texinfo mode:

```
(add-hook 'texinfo-mode-hook 'turn-on-font-lock)
```

  See Section 6.2 [Minor Modes], page 28, for information on font-lock mode.

- Rebinds the key *C-x l* to run the function `make-symbolic-link`:

```
(global-set-key "\C-xl" 'make-symbolic-link)
```

  We use the single quote before "make-symbolic-link" because its a function name. You can also use the following expression which does the same thing:

```
(define-key global-map "C-xl" 'make-symbolic-link)
```

- The following expression will bind *C-x l* to run the function `make-symbolic-link` in C mode only:

```
(define-key c-mode-map "C-xl" 'make-symbolic-link)
```

  Instead of binding *C-xl* to run `make-symbolic-link`, you can bind the ⟨F1⟩ key to run this function:

```
(define-key c-mode-map 'f1 'make-symbolic-link)
```

  Here, you have to use lower case for naming function keys like ⟨F1⟩.

- You can bind the function `undo` i.e. *C-x u* to any key, for example to ⟨F2⟩:

```
(global-set-key 'f2 'undo)
```

- The following statement will display the current time in the modeline of the buffer:

```
(display-time)
```

- This displays the current line number on which the cursor is present in the modeline:

```
(setq line-number-mode t)
```

- If you don't want the text to be highlighted when you use commands for marking regions so as to use the *kill* and *yank* commands later, you can use the following expression in your '`.emacs`' file:

```
(setq zmacs-regions nil)
```

  Now if you use a command like *C-x C-p* (`mark-page`), the text will not be highlighted.

- To control the number of buffers listed when you select the **Buffers** menu, you need to set the variable *buffers-menu-max-size* to whatever value you wish. For example, if you want 20 buffers to be listed when you select **Buffers** use:

```
(setq buffers-menu-max-size 20)
```

- If you want the window title area to display the full directory/name of the current buffer's file, and not just the name, use:

```
(setq frame-title-format "%S: %f")
```

- To get rid of the menu, use :

```
(set-menubar nil)
```

- If you want an extensive menu-bar use the following expression in your '`.emacs`' file.

```
(load "big-menubar")
```

  If you want to write your own menus, you can look at some of the examples in '`/usr/local/lib/xemacs-20.0/lisp/packages/big-menubar.el`' file.

For more information on initializing your '`.emacs`' file, See section "Init File" in *XEmacs User's Manual*. You should also look at '`/usr/local/lib/xemacs-20.0/etc/sample.emacs`', which is a sample '`.emacs`' file. It contains some of the commonly desired customizations in Emacs.

9 Selecting and Moving Text

Many Emacs commands operate on an arbitrary contiguous part of the current buffer. You can select some part of the buffer and edit only that part of the buffer. This selected buffer is called a *region*. You can select text in two ways:

- You use special keys to select text by defining a region between the cursor and *the mark* (which you set).
- If you are running XEmacs under X, you can also select text with the mouse.

9.1 Setting the Mark

To define a region you need to set *the mark* at one end of it and move the cursor to the other end. Once you set the mark, it remains there until you set it again to some other place. Each buffer has its own *mark ring* (a place where Emacs remembers 16 previous locations of the mark). To set *the mark*, you can use the following commands:

`C-SPC` This command will set *the mark* at the position of your cursor (`set-mark-command`). You can move your cursor around and *the mark* will stay there.

`C-x C-x` Interchange mark and point (`exchange-point-and-mark`). Since Emacs will have only one cursor, after you move the cursor it will be unable to show you where you set the *the mark*. In order to see *the mark* you can type the command `C-x C-x` which will put your cursor on the position of your mark and your mark on the position of your cursor. Use the command again to reset the positions of your cursor and mark.

`C-<` This command will push the mark at the beginning of the buffer without changing the position of your cursor.

`C->` This command will push the mark at the end of the buffer without changing the position of your cursor.

You can also give arguments to `C-<` or `C->`. See section "The Mark and the Region" in *XEmacs User's Manual*, for more information.

9.2 Selecting Text with Mouse

If you are using XEmacs under X, you can use the mouse to select text. The selected text will always be highlighted, so just by looking at the text you know what you have selected so far. To select a word just double-click with the left-mouse-button on the word. To select a whole line triple-click anywhere on the line with the left-mouse-button. You can also use the **Copy** item from the **Edit** menu on the menu-bar to select text. This kind of selection is called **Clipboard** selection, See section "X Clipboard Selection" in *XEmacs User's Manual*, for more information. To select an arbitrary region, follow these steps:

1. Move the mouse cursor over the character at the beginning of the region of text you want to select.
2. Press and hold the left mouse button.

3. While holding the left mouse button down, drag the cursor to the character at the end of the region of text you want to select.
4. Release the left mouse button.

The selected region of text is highlighted.

See section "Selecting Text with the Mouse" in *XEmacs User's Manual*, for more information regarding the Mouse and additional mouse operations.

9.3 Operating on the Region

Once you have selected a region you can do a lot of things to the text in the region:

- Kill the text with `C-w`. For example if you want to kill a paragraph, position the cursor to the beginning of the paragraph and type `C-SPC`. Then go to the end of the paragraph and type `C-w`. The entire paragraph will be deleted. You can also select the text with a mouse and type `C-w` to kill the entire region. See section "Killing" in *XEmacs User's Manual*, for more information.
- Save the text in a buffer or a file (see section "Accumulating Text" in *XEmacs User's Manual*).
- You can convert the case of the text with `C-x C-l` or `C-x C-u` If you type `C-x C-u` the selected text will become all upper-case. If you type `C-x C-l` the selected text will become all lower-case.
- Print hardcopy with `M-x print-region`. See section "Hardcopy" in *XEmacs User's Manual*, for more information. This command will print a hardcopy of only the selected text.
- Indent it with `C-x` ⟨TAB⟩ or `C-M-\` See section "Indentation" in *XEmacs User's Manual*, for more information.

9.4 Moving Text

The most common way to move or copy text in Emacs is through *killing* or '`cutting`' it and then *yanking* or '`pasting`' it. You can also use the **Cut** or **Copy** option from the **Edit** menu for killing and copying respectively. See Section 2.2.2 [Edit menu], page 11, for reviewing the commands for killing text. All the killed text in Emacs is recorded in the *kill ring*. Since there is only one kill ring in Emacs, you can kill text in one buffer and yank it in another buffer. To '`paste`' or '`yank`' the killed text you can use the following commands:

`C-y`
: This command will yank or paste the last killed text (`yank`).

`M-w`
: Save region as last killed text without actually killing it (`copy-region-as-kill`). You can use this command to copy a selected region and then yank (or paste) it without actually removing it from the buffer.

`C-M-w`
: Append next kill to last batch of killed text (`append-next-kill`). This command will append whatever you killed last to what you kill now. Then later you will be able to yank the entire appended text from the *kill ring*.

9.5 Accumulating Text

The following commands can be used for accumulating text from different buffers into one place or for copying one region of text into many buffers:

M-x append-to-buffer
: Append region to contents of specified buffer (`append-to-buffer`). After you type in this command and press ⟨RET⟩, Emacs will prompt you for a buffer name. You will see a message in the echo area:

```
Append to buffer: (default <buffer name>)
```

After you type in a buffer name, a copy of the region will be inserted at the location of the cursor into that buffer. If there is no buffer with the name given by you, Emacs will create a new buffer with that name. By default the cursor's position in the `<`buffer name`>` is at the end.

M-x prepend-to-buffer
: Prepend region to contents of specified buffer. This command is similar to the above command except that the cursor in the buffer (by default) is at the beginning rather than at the end.

M-x copy-to-buffer
: Copy region into specified buffer, deleting that buffer's old contents. This command will also prompt you for a buffer name.

M-x insert-buffer
: Insert contents of specified buffer into current buffer at point. This command will prompt you for a buffername which you want to be copied into the current buffer at the location of the cursor.

M-x append-to-file
: This command will prompt you for a filename and append the region to the end of the contents of the specified file.

See section "Accumulating Text" in *XEmacs User's Manual*, for more information regarding this topic.

You can also use *rectangle commands* for operating on rectangular areas of text. See section "Rectangles" in *XEmacs User's Manual*, for more information regarding rectangle commands.

Emacs also provides *registers* which serve as temporary storage for text or positions. Each register has a one character name and they can store *regions*, a *rectangle*, or a *mark* i.e. a cursor position. Whatever you store in register stays there until you store something else in that register. To find out about commands which manipulate registers See section "Registers" in *XEmacs User's Manual*.

10 Searching and Replacing

Emacs provides commands for searching for occurrences of a particular string. The search is incremental i.e. it begins even before you complete typing the whole string. All searches in Emacs ignore the case of the text they are searching, i.e. if you are searching for "String", then "string" will also be one of the selections. If you want a case sensitive search select the **Case Sensitive Search** from the **Option** menu. You can also set the variable *case-fold-search* to *nil* for making searches case-sensitive. For information on setting variables, See Section 8.1 [Setting Variables], page 35. The two commands for searching for strings in XEmacs are:

`C-s` This command will prompt you for a string to search :

```
I-search:
```

If you type "myname" as the string to be searched, then Emacs will start searching for "m", "my", "myn", etc as you go on typing the whole string in the forward direction. The cursor will be on the matching string which has been found so far. If you find the correct match just hit ⟨RET⟩ or type `C-f` or `C-b` to set the cursor's position. If you find a matching string "myname" but you were looking for a different occurrence of it, use `C-s` again. If the search is unable to find the string, it will give you an error message.

`C-r` This command will perform an incremental search in the backward direction. It will prompt you for a string name:

```
I-search backward:
```

After you start typing the string name, it will search for the string in the same fashion as it does for `C-s` except that it will search in the backward direction. If it cannot find the string name, it will give you an error message.

If you make a mistake while typing the string names when you use the above commands, you can use the ⟨DEL⟩ key to erase characters. Each ⟨DEL⟩ will erase the last character. At any time if you want to quit the search, just type `C-g`.

To do a non-incremental search i.e. to start the search only after you have typed the whole string you can use the following commands:

`C-s` *RET string RET*
This command will search for the specified string in the forward direction and will give an error message if the string is not found.

`C-r` *RET string RET*
This command will search for the specified string in the backward direction.

For information on how Emacs searches for words and regular expressions, See section "Search" in *XEmacs User's Manual*.

To replace all occurrences of a string in Emacs, you can use the following command:

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

After you type `M-x replace-string`, you will be prompted for a string name to replace:

```
Replace string:
```

After you type in a string name, for example "FOO" and press ⟨RET⟩, you will see another prompt:

```
Replace string FOO with:
```

Now type the string which you want to replace "FOO" with and press ⟨RET⟩. After all the occurrences are replaced you will see the message "Done" in the echo area. If you want only some occurrences of the string to be replaced, use *M-x query-replace RET <string> RET <newstring> RET*. For more information, See section "Query Replace" in *XEmacs User's Manual*.

XEmacs also provides a utility for checking spellings. Use *M-x ispell-buffer* to check for spellings in the whole buffer. You can also check the spelling of a word or a region. You can use menus to check for spellings:

Evaluate the expression `(load "big-menubar")`. To evaluate this expression you need to hit the ⟨META⟩ or the ⟨ESC⟩ key twice and type in the expression in the echo area before hitting ⟨RET⟩. You will get an extensive menubar. Select the **Spell Check** menu item from the **Utilities** menu for checking spellings.

Key (Character) Index

C

D

M

R

Command and Function Index

T

W

Y

Z

Variable Index

Concept Index

M

N

O

P

R

S

T

U

V

W

Y

Short Contents

Table of Contents