

# Chapter 10: Editors

Several text editors are available at Fermilab. In this chapter we present our view of the advantages and disadvantages of the available editors, and we provide some basic information on the setup and use of each one. You will learn how to invoke each editor, and how to create, edit, and save a file in each one using a small subset of commands and features.

## 10.1 The Available Editors

---

<b>vi</b>	A native UNIX screen editor. It is not generally considered to be one of the better available editors, however knowing the basics is useful for two reasons: 1) you may occasionally encounter an application that throws you into a <b>vi</b> session, from which you'll at least want to exit, and 2) <b>vi</b> is the one editor you are guaranteed to find on all UNIX machines <sup>1</sup> .
<b>emacs</b>	A powerful modern public-domain screen editor. You will find <b>emacs</b> installed on most UNIX machines.
<b>xemacs</b>	An incarnation of the advanced, self-documenting, customizable, extensible real-time display editor <b>emacs</b> . It provides many powerful display and user-interface capabilities not found in <b>emacs</b> .
<b>NEdit</b>	A popular and intuitive X-based editor written at Fermilab for UNIX.

---

1. In addition, UNIX provides **sed** for non-interactive editing and **awk** for more sophisticated and complex non-interactive editing. These two products are not covered in this manual.

## 10.2 Comparison of Editors

---

The following table discusses the advantages and disadvantages of each of the above-mentioned editors.

Editor	Advantages	Disadvantages
<b>vi</b>	Available on all supported platforms and systems as a native UNIX tool. Well-documented. It is set as the default editor for many applications on many systems.	Considered to be very primitive in its screen capabilities. Non-intuitive interface. Lowest common denominator of screen editors.
<b>emacs</b>	Widely available public domain software package. Well documented. Many macros and enhancements available via the internet. Language-specific text editing modes (e.g., English, Lisp, C, FORTRAN). Lots of expertise and consulting available via newsgroups, etc. Very configurable. <b>EDT</b> -style keypad emulation available. X and ASCII modes. Supports file versions.	Requires installation of <b>emacs</b> . Interface not very intuitive.
<b>Xemacs</b>	Same as for <b>emacs</b> , plus more. Intuitive, GUI interface. Language-specific syntax-highlighting. Multi-windowed, interactive, object-oriented class browser.	Requires installation of <b>Xemacs</b> .
<b>NEdit</b>	Very intuitive and well-documented. Customizable. Gaining popularity world-wide. X-based.	Requires installation of <b>NEdit</b> . Requires X-based terminal. Not available on all systems/platforms.

Note: For all editors used with X display, the *DISPLAY* variable (see section 9.6 *Some Important Variables*) must be set properly.

## 10.3 Getting Started with the Editors

---

In this section we present a few important commands for each editor. The minimal information necessary for you to edit and save a simple file is provided. All these editors have many commands and sophisticated features that we do not cover here.

## 10.3.1 vi

As we mentioned in section 10.1 *The Available Editors* above, you may occasionally find yourself thrown into the **vi** editor unexpectedly, and you will certainly want to know how to exit, if nothing else. For that reason alone you should become familiar with a few **vi** commands. As is typical in UNIX, the **vi** commands are case sensitive.

**vi** requires no setup. Invoke it using the command:

```
% vi [<filename>]
```

Once the file is opened, you are in *command mode*. From this mode you can issue commands, move the cursor, and invoke insert mode.

To enter *insert mode*, type **i**. Text you type will be inserted *before* the cursor. From insert mode you can enter new text in the file. Press the **ESCAPE** key to exit insert mode and return to command mode. On many keyboards the **ESCAPE** key is labelled; if not, the sequence **<CTRL-[]** is mapped to the escape function.

### Some useful vi commands available in command mode

<b>h, j, k, l</b>	move cursor left, down, up, right, respectively
<b>H</b>	move to top line of screen
<b>L</b>	move to bottom line of screen
<b>&lt;CTRL-F&gt;, &lt;CTRL-B&gt;</b>	scroll forward, backward one screen
<b>/&lt;pattern&gt;</b>	search for <b>&lt;pattern&gt;</b>
<b>/</b>	repeat search in forward direction
<b>x</b>	delete current cursor position
<b>X</b>	delete back one character
<b>dw</b>	delete current word
<b>dd</b>	delete current line
<b>&lt;n&gt;dd</b>	delete <b>&lt;n&gt;</b> lines starting with current
<b>p</b>	insert ( <b>paste</b> ) last deleted text after cursor
<b>:r &lt;filename&gt;</b>	read in contents of <b>&lt;filename&gt;</b> after cursor
<b>:x</b>	quit <b>vi</b> , writing file only if changes were made
<b>:w</b>	write file, do not quit
<b>:w &lt;file&gt;</b>	save copy to <b>&lt;file&gt;</b> , do not quit
<b>:q!</b>	quit file, discarding edits

Most UNIX guides contain a complete description of **vi**.

## 10.3.2 emacs and xemacs

We'll cover these two editors in the same section because although they are separate products, they are closely related.

**emacs** is a popular editor available on the net. It can be invoked in windows mode or ASCII mode. Many UNIX books cover **emacs**, and a good reference for **GNU emacs** is *Learning GNU Emacs* (O'Reilly & Associates). Further documentation can be found from the man pages.

**xemacs** is a graphical, X window implementation of **emacs**, with a few extra bells and whistles. (It is somewhat fancier than **emacs** in windows mode, and not to be confused with it.) **xemacs** is very similar in appearance and operation to many PC text processing applications for Windows. It is menu/mouse driven, with keyboard shortcuts available. **xemacs** has commands for passing single command lines to shell processes; it can also run a shell interactively.

### Setup and Invoke emacs

To use **emacs**, the product needs to be installed. To set it up (under **UPS**), include in your login script or enter:

```
% setup emacs
```

The mode (X or ASCII) in which **emacs** attempts to start-up is determined according to your *DISPLAY* variable. To invoke emacs, type:

```
% emacs [<options>] [<filename>] [&]
```

To invoke it in ASCII mode without a new window, type:

```
% emacs -nw [<options>] [<filename>]
```

### Setup and Invoke xemacs

To use **xemacs**, the product needs to be installed. To set it up (under **UPS**), include in your login script or enter:

```
% setup xemacs
```

To invoke **xemacs**, type:

```
% xemacs [<filename>] [&]
```

### Help Facilities

**emacs/xemacs** has an extensive interactive help facility, but the facility assumes that you know how to manipulate **emacs** windows and buffers. **<CTRL-H>** enters the Help facility. **<CTRL-H>-T** enters the help tutorial, which can teach beginners the fundamentals of **emacs** in a few minutes. **<CTRL-H>-A** enters Help Apropos, to help you find commands by function. For **emacs** in windows mode, there is also a **HELP** menu. **<CTRL-H>-I** enters

the Info facility which brings up the on-line documentation browsing system. The initial page (the Directory node) gives a menu of major topics. The information is presented in a hierarchical tree format. **xemacs** provides this via an **INFO** button.

## Keyboard Commands

**emacs/xemacs** commands use the *Control key* and the *Meta key*<sup>1</sup>. In the following list **C-** indicates that the control key is pressed at the same time as the character that follows. Similarly, **M-** indicates the use of the Meta key, although it's not necessary to keep the Meta key pressed down while typing the next character. Note that some command sequences use multiple keystrokes, with and without the Control and Meta keys. A sequence like **C-x u** means hold down control while you press **x**, then just press **u**. Following is a list of the **emacs** commands used most often:

<b>C-h</b>	enter on-line help system
<b>C-h i</b>	enter the information browser which provides a menu of major topics (use <b>TAB</b> and <b>ENTER</b> keys or 2nd mouse button to select a topic; navigation information is provided)
<b>C-p, C-n, C-f, C-b</b>	move up (to <b>p</b> revious), down (to <b>n</b> ext), <b>f</b> orward, or <b>b</b> ackward by one line or character, respectively
<b>C-v, M-v</b>	move forward, backward, by one screen
<b>C-s</b>	search forward for characters (system will prompt you for string). To continue search, type <b>C-s</b> again.
<b>C-r</b>	search backward
<b>C-d</b>	delete a character
<b>C-k</b>	delete (kill) from cursor to end of line
<b>C-y</b>	restore what you've deleted
<b>C-x u</b>	undo last edit
<b>C-g</b>	get out of current command operation
<b>C-x i</b>	insert file at cursor position (system will prompt for filename)
<b>M-q</b>	fill paragraphs
<b>C-@</b> or <b>C-SPACEBAR</b>	set the mark for the start or end of a region to select

---

1. If you have a key labelled **META**, use it; if you don't, try the **ALT** key or the **ESCAPE** key (if you're running a native X window). As a last resort, the sequence **<CTRL-[>** should always work as a Meta key.

<b>C-w</b>	delete all between mark (see <b>C-@</b> ) and cursor's current position (paste back with <b>C-y</b> )
<b>M-w</b>	copy all between mark (see <b>C-@</b> ) and cursor's current position (paste back with <b>C-y</b> )
<b>C-x C-x</b>	exchange mark and cursor's current position (since the mark is invisible, this allows you to find it)
<b>C-x C-s</b>	save the file
<b>C-x C-w</b>	write to file (system will prompt for filename)
<b>C-x C-b</b>	display buffer list
<b>C-x o</b>	move cursor to other window (when more than one displayed)
<b>C-x C-c</b>	exit emacs

Note, if the serial port or terminal device you are typing on is configured for **<CTRL-S>/<CTRL-Q>** flow control, you may find that **<CTRL-S>** (written above as **C-s**) within **emacs** causes the terminal to stop sending characters, the same as when used at the shell prompt. If you want to use the usual **emacs** key bindings and to have **C-s** work properly within **emacs**, you'll need to reconfigure your line to not do flow control. Where and how you do this depends on how you're connected. If you're connected via a modem, you may need to reconfigure your modem, as well as the pseudo-terminal on your UNIX host. The latter can be done via the command:

```
% stty stop undef start undef
```

(which sets the stop and start characters to "undefined"). You could include this statement in your `.login` or `.profile`. The intermediate step, between the on-site modem and the Cisco router/terminal server, has flow control turned off by default.

## Language-Specific Text Editing Environments

**emacs/xemacs** supports several text editing environments (called modes), each geared to a particular language (e.g., English, Lisp, C, FORTRAN). When the editor is started, it normally loads the file `$HOME/.emacs`, if present, which contains Lisp commands for initialization. In particular, by setting up a mapping in this file between file extensions and languages, you can configure **emacs/xemacs** to come up in the appropriate mode according to the extension of the file you specify on the invoking command line. The text you need to include in `.emacs` has the syntax:

```
(setq auto-mode-alist (append `
  ("\\.extension1$" . language1-mode)
  ("\\.extension2$" . language2-mode)
  ("\\.extension3$" . language3-mode)
  ("\\.extension4$" . language4-mode)
```

```
...  
  ) auto-mode-alist))
```

For example:

```
(setq auto-mode-alist (append `(  
  ("\\.asm$" . asm-mode)          ("\\.s$" . asm-mode)  
  ("\\.awk$" . awk-mode)  
  ("\\.cc$" . c++-mode)          ("\\.C$" . cc-mode)  
  ("\\.hh$" . c++-mode)  
  ("\\.c$" . c-mode)            ("\\.h$" . c-mode)  
  ("\\.i$" . c-mode)  
  ("\\.m$" . objc-mode)        ("\\.csh$" . c-mode)  
  ("\\.cdf$" . fortran-mode)   ("\\.cin$" . fortran-mode)  
  ("\\.for$" . fortran-mode)  ("\\.f$" . fortran-mode)  
  ("\\.F$" . fortran-mode)  
  ("\\.inc$" . fortran-mode)   ("\\.car$" . fortran-mode)  
  ("\\.cra$" . fortran-mode)   ("\\.crb$" . fortran-mode)  
  ("\\.tex$" . TeX-mode)       ("\\.txi$" . Texinfo-mode)  
  ("\\.el$" . emacs-lisp-mode) ("\\.icc$" . c++-mode)  
  ) auto-mode-alist))
```

The on-line GNU EMACS manual provides more information on creating and modifying this file.

## Use with EDT-Style Keypad



**emacs/xemacs** can be set to have an **EDT**-style keypad. See the emacs and xemacs product documentation for details; here we provide some start-up information.

**emacs** (windows) and **xemacs** The key bindings are in the control of the editor emulation. When you first start up the **tpu-edt** emulator (instructions follow) it will prompt you to setup a keyboard mapping. You need a separate mapping for each different keyboard type you use.

**emacs** in non-windows mode Here you are at the mercy of your window emulator. There are things you can do to remap keys on most vt100 window emulators, but it is different for each OS/emulator. There are too many permutations to document.

To invoke the emulation, put the following text at the top of your `$HOME/.emacs` file:

```
(tpu-edt)                ;; Basic Emulation  
(tpu-set-scroll-margins "10%" "15%") ;; Set scroll margins  
10% (top) and 15% (bottom).  
(load "vt-control" t)    ;; VT terminal controls (No complaint  
if not available)  
;;
```

```
;; TPU-edt treats words like EDT; here's how to add word
separators.
;; Note that backslash (\) and double quote (") are quoted
with `\'`.
(tpu-add-word-separators "]\\[-_\\.\"'+()*/*#;!&,$")
```

To try this out without changing your `.emacs` file, first invoke the editor, then press **ALT-x** (hold down **ALT** while pressing **x**), and type **tpu-edt** followed by a carriage return.



On some platforms you may have trouble with the Gold key. The problem and its solution are dependent on the type of terminal and keyboard you are using.

## The xemacs GUI Interface

Due to the user-friendly nature of the product, we present only a few basic commands to give you a flavor of this type of editor if you are not familiar with it:

Open a file	<b>OPEN</b> from the toolbar or File menu (you can also open in another window or in a new frame); choose an existing file from popup window
Create a new file	<b>OPEN</b> from the toolbar or File menu; type in the new file name
Include a file	<b>INSERT FILE</b> from the File menu
Select text	Use the mouse
Highlight special syntax in color (for use with language modes)	<b>SYNTAX HIGHLIGHTING</b> from Options menu.
Cut/Copy/Paste text	<b>CUT/COPY/PASTE</b> from the toolbar or Edit menu
Search for text	<b>SEARCH</b> from the Edit menu
Spell check	<b>SPELL</b> from the toolbar or Edit menu.
Save the file	<b>SAVE</b> from the File menu
Close the file	<b>DELETE BUFFER</b> from the File menu (use the Buffer menu to select the buffer to close)
Exit xemacs	<b>EXIT EMACS</b> from the File menu
<b>C-g</b>	get out of current command operation

## The xemacs oo-Browser

The OO-Browser is a multi-windowed, interactive, object-oriented class browser. It currently supports the following object-oriented languages (Eiffel, C, C++, Objective-C, CLOS (Lisp), Java, Python and Smalltalk), one



non-object-oriented language (C), and one documentation language, (GNU Info). On the Web you can find documentation for the oo-Browser with the **xemacs** product.

Before using the browser, you must create a database of information for each set of source code files you plan to use.

Here is a brief set of instructions for creating a database:

1) Run **setup xemacs**

2) Change to the directory in which you want the browser database output to go (we'll call it `outputdir`), and invoke **xemacs**.

3) Select OO-Browser from the **TOOLS** pull down menu. In the prompting window at the bottom of the window it will say: `Load/Create OO-Browser Environment: {outputdir}/`

Enter a filename. This file is used to store the answers to the following questions. The application will then read this file to build your browsing environment. The file can be reused in future sessions. Here we'll use the filename `OOBR`, to create the file `{outputdir}/OOBR`.

4) Next it prompts for a language: `Choose: 1) C++/C; 2) Eiffel; 3) Info; 4) Java; 5) Lisp; 6) Obj-C; 7) Python; 8) Smalltalk`

Enter the number corresponding to your choice.

5) Some error messages may rapidly scroll by. Ignore them<sup>1</sup> and wait for the following prompt: `Please specify the "OOBR" Environment (Hit RET to begin).`

Enter **RETURN** as requested.

6) Next it prompts for a list of "system" directories and then "library" directories. You can specify each directory using an absolute or a relative path name (relative to your current working directory). Specify your own source code files as "system", and any library source code files you need as "library". Terminate the list by entering a carriage return on a fresh line.

7) The next prompt is: `Build Environment from spec in file, "{outputdir}/OOBR"? (y or n)`

Enter **y**

8) The final prompt is: `Build Environment in a background process? (y or n)`

Enter **n** (in order to monitor what happens)

The oo-Browser starts scanning all of the files in the directory tree underneath the directories you specified. When it finishes, your database is made and you are ready to start browsing.

---

1. A knowledgeable source suspects that these messages are the result of a bug and will go away in a future release. She has heretofore ignored them with no ill effects.

A couple of useful keys are **F** and **V**. **F** displays an expanded (full) summary of the member functions belonging to the selected class. If you enter **V** with the cursor on a class name, the source file that defines the class (usually a header file) is displayed for viewing. Enter **E** to display it for editing. If you enter **V** with the cursor on a member function name, it displays the source code for that member function (usually an implementation file). If there are many functions in the same file, the browser places you at the correct line number for the selected function.

### 10.3.3 NEdit

**NEdit** is very similar in appearance and operation to many PC text processing applications for Windows. It is menu/mouse driven, with keyboard shortcuts available. Some UNIX shell commands are available from within the editor. Make sure the **NEdit** product is installed on your system. To set up **NEdit** under UPS, include in your login script or enter:

```
% setup nedit
```

To invoke **NEdit**, type:

```
% nedit [<filename>]
```

Due to the user-friendly nature of the product, we present only a few basic commands to give you a flavor of this type of editor if you are not familiar with it:

Open a file	<b>OPEN</b> from the File menu; choose an existing file from popup window
Create a new file	<b>NEW</b> from the File menu
Include a file	<b>INCLUDE</b> from the File menu
Select text	Use the mouse, or the shift and arrow keys together
Cut/Paste text	<b>CUT/PASTE</b> from the Edit menu
Search for text	<b>FIND</b> from the Search menu
Fill paragraph	<b>FILL PARAGRAPH</b> from the Edit menu
Spell check	<b>SPELL</b> from the Shell menu
Save the file	<b>SAVE</b> from the File menu
Close the file	<b>CLOSE</b> from the File menu (you are prompted about saving)
Exit <b>NEdit</b>	<b>EXIT</b> from the File menu

Further information is available from the man pages and a plain text document in the distribution kit, but the on-line help in the program is complete and more convenient. The documentation is also available on the Web in the product documentation area.

