Chapter 11: 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. We include only minimal usage information for the VMS-style editors.

11.1 The Available Editors

vi — 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 **vi** session, from which you'll at least want to exit, and 2) **vi** is the one editor you are guaranteed to find on all UNIX machines[1].

emacs — A powerful modern public-domain screen editor available for both VMS and UNIX. You will find **emacs** installed on most UNIX machines.

xemacs — An incarnation of the advanced, self-documenting, customizable, extensible real-time display editor **emacs**. It provides many powerful display and user-interface capabilities not found in **emacs**.

NEdit — A popular and intuitive X-based editor written at Fermilab for UNIX.

The remaining available editors are UNIX implementations of Digital's products for VMS:

nu/TPU — An emulation of TPU

fermitpu — A locally ported UNIX version of the newest EVE (from VMS V6.1)

EDT+ — A UNIX version of EDT[2]

The VMS-style editors are legacy products, and as such are not guaranteed to remain available beyond the VMS-to-UNIX migration period. It is also unlikely that you will find these third party editors on UNIX systems outside of Fermilab. They are provided solely to ease the transition to UNIX for users already familiar with these products on VMS. **Users not already familiar with these VMS editors are strongly discouraged from choosing one of them for use in UNIX!** *All* users are encouraged to begin learning one of the permanent UNIX editors.

---

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.
2. Due to budgetary constraints and soaring maintenance costs, support for **EDT** under UNIX is now frozen at release v6_3a. We expect that this release will continue to function properly. However, if future hardware or software changes cause **EDT** to break, it will not be replaced or upgraded. **EDT** will *not* be purchased for any new operating systems which may be supported in the future.

11.2 Comparison of Editors

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

Editor	Advantages	Disadvantages
vi	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.
emacs	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. **EDT**-style keypad emulation available. X and ASCII modes. Supports file versions.	Requires installation of **emacs**. Interface not very intuitive for users accustomed to VMS-style editors.
Xemacs	Same as for **emacs**, plus more. Intuitive, GUI interface. Language-specific syntax-highlighting. Multi-windowed, interactive, object-oriented class browser.	Requires installation of **Xemacs**.
NEdit	Very intuitive and well-documented. Customizable. Gaining popularity world-wide. X-based.	Requires installation of **NEdit**. Requires X-based terminal. Not available on all systems/platforms.
nu/TPU	Nearly identical to Digital's **TPU** for people familiar with **TPU** programming. Sufficient for a wide variety of simple editing tasks. Close enough to **EVE/EDT**-style keypad to be familiar to VMS users. Does not exhibit screen-repainting problem of **fermitpu** (see below).	Duration of availability is uncertain. Implementation includes many faults. Limited documentation available on-line. Uses old version of **EVE** (from VMS V5.0). Does not match the interface that VMS users are used to. Not X-based. Requires VT-100 or VT-200 compatible keyboard mappings.

Editor	Advantages	Disadvantages
fermitpu	Nearly identical to Digital's **EVE** of VMS V6.1 with the exception that the **EDT**-style keypad is used by default. Licensing is handled on the local node (i.e., no network license server problems). User-configurable and extendable. Does not require X.	Duration of availability is uncertain. Not available/installed on all systems/platforms. Not the same as **fermitpu** on the central facility VMS systems, although similar. Requires the FUE environment and the installation of **nu/TPU** and the **fermitpu** layered products. Not X-based. Requires VT-100 or VT-200 compatible keyboard mappings. On some platforms, the screen is repainted after every keystroke which is very slow.
EDT+	Very similar to an enhanced **EDT** for people familiar with **EDT** programming. Incorporates a powerful word processor, disaster recovery system, an extensive help facility, full feature programmable text processing and the GOLD-KEY style of editing. Does not require X, although X interface is available. Minimum learning curve for users familiar with Digital's **EDT**.	Duration of availability is uncertain; maintenance contract terminated. Requires the FUE environment and the installation of the **EDT** product. Not available/installed on all systems/platforms. Distributed licensing (implies that network problems and/or problems with the license server could prevent **EDT+** from being usable on occasion). Not X-based. Requires VT-100 or VT-200 compatible keyboard mappings.

Note: For all editors used with X display, the *DISPLAY* variable (see section 9.2) must be set properly.

11.3 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.

11.3.1 vi

As we mentioned in section 11.1 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

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

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

11.3.2 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 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). We also refer you to an on-line GNU EMACS manual accessible under **Editors** on the *UNIX Resource*s Web page. 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. All standard **emacs** keyboard commands can be used in **xemacs** instead of/in addition to the pull-down menu commands. An **xemacs** manual is accessible under **Editors** on the *UNIX Resource*s Web page.

Setup and Invoke emacs

To use **emacs**, the product needs to be installed. To set it up, 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, type:

```
% emacs [options] [filename]
```

To invoke it in ASCII mode without a new window using an X terminal, type:

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

Setup and Invoke xemacs

To use **xemacs**, the product needs to be installed. To set it up, 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*[1]. 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:

C-h	enter on-line help system
C-h i	enter the information browser which provides a menu of major topics (use tab and enter keys or 2nd mouse button to select a topic; navigation information is provided)
C-p, **C-n**, **C-f**, **C-b**	move up (to **p**revious), down (to **n**ext), **f**orward, or **b**ackward by one line or character, respectively
C-v, M-v	move forward, backward, by one screen
C-s	search forward for characters (system will prompt you for string). To continue search, type **C-s** again.
C-r	search backward

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.

C-d	delete a character
C-k	delete (kill) from cursor to end of line
C-y	restore what you've deleted
C-x u	undo last edit
C-g	get out of current command operation
C-x i	insert file at cursor position (system will prompt for filename)
M-q	fill paragraphs
C-@ or **C-**spacebar	set the mark for the start or end of a region to select
C-w	delete all between mark (see **C-@**) and cursor's current position (paste back with **C-y**)
M-w	copy all between mark (see **C-@**) and cursor's current position (paste back with **C-y**)
C-x C-x	exchange mark and cursor's current position (since the mark is invisible, this allows you to find it)
C-x C-s	save the file
C-x C-w	write to file (system will prompt for filename)
C-x C-b	display buffer list
C-x o	move cursor to other window (when more than one displayed)
C-x C-c	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 '(
```

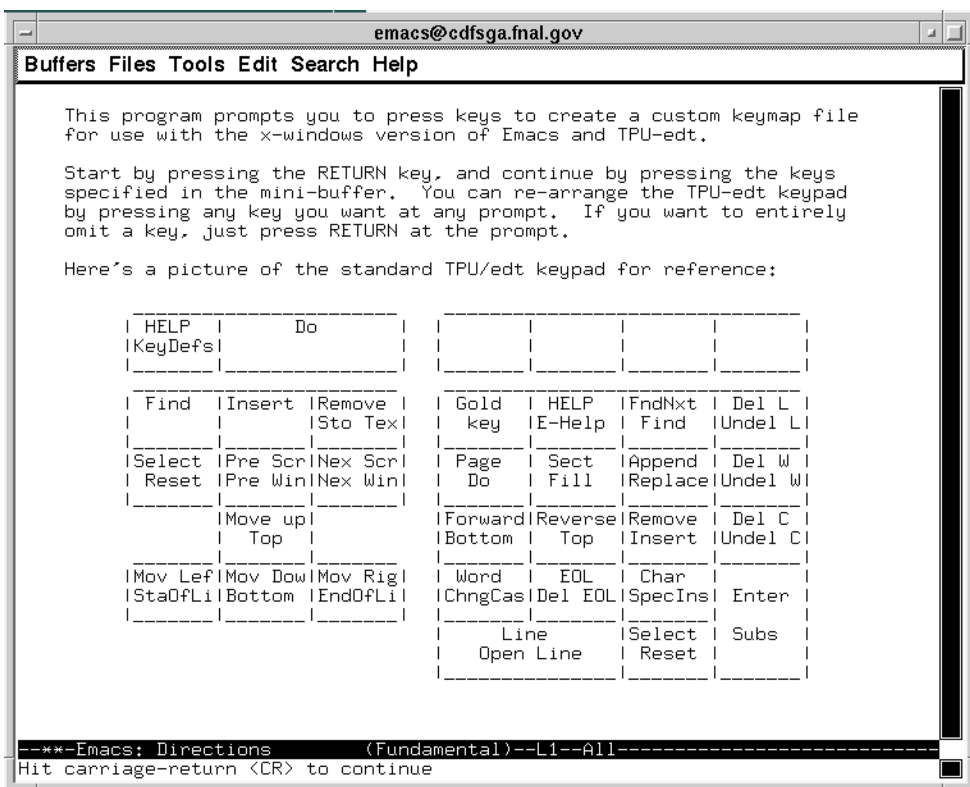
```
("\\.asm$" . asm-mode)      ("\\.s$"   . asm-mode)  ("\\.awk$" . awk-mode)
("\\.cc$"  . c++-mode)      ("\\.C$"   . cc-mode)   ("\\.hh$"  . c++-mode)
("\\.c$"   . c-mode)        ("\\.h$"   . c-mode)    ("\\.i$"   . c-mode)
("\\.m$"   . objc-mode)     ("\\.csh$" . c-mode)
("\\.cdf$" . fortran-mode)  ("\\.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. This is documented more thoroughly in our Web pages; 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 it will prompt you to setup a keyboard mapping (see screen below). 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.


```
emacs@cdfsga.fnal.gov
Buffers Files Tools Edit Search Help

    This program prompts you to press keys to create a custom keymap file
    for use with the x-windows version of Emacs and TPU-edt.

    Start by pressing the RETURN key, and continue by pressing the keys
    specified in the mini-buffer.  You can re-arrange the TPU-edt keypad
    by pressing any key you want at any prompt.  If you want to entirely
    omit a key, just press RETURN at the prompt.

    Here's a picture of the standard TPU/edt keypad for reference:

          _______________________    _________________________________
         | HELP  |      Do       |  |        |        |        |        |
         |KeyDefs|               |  |        |        |        |        |
         |_______|_______________|  |________|________|________|________|
          _______________________    _________________________________
         | Find  |Insert |Remove |  | Gold   | HELP   |FndNxt  | Del L  |
         |       |       |Sto Tex|  |  key   |E-Help  | Find   |Undel L |
         |_______|_______|_______|  |________|________|________|________|
         |Select |Pre Scr|Nex Scr|  | Page   | Sect   |Append  | Del W  |
         | Reset |Pre Win|Nex Win|  |  Do    | Fill   |Replace |Undel W |
         |_______|_______|_______|  |________|________|________|________|
                 |Move up|          |Forward |Reverse |Remove  | Del C  |
                 |  Top  |          |Bottom  |  Top   |Insert  |Undel C |
          _______|_______|_______   |________|________|________|________|
         |Mov Lef|Mov Dow|Mov Rig|  | Word   |  EOL   | Char   |        |
         |StaOfLi|Bottom |EndOfLi|  |ChngCas|Del EOL |SpecIns |  Enter |
         |_______|_______|_______|  |________|________|________|        |
                                    |      Line       |Select  |  Subs  |
                                    |    Open Line    | Reset  |        |
                                    |_________________|________|________|

--**-Emacs: Directions        (Fundamental)--L1--All---------------------------
Hit carriage-return <CR> to continue
```

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. We know the solutions for some combinations. We are planning to provide this information soon in a Web page under *UNIX Resources*, and to keep it updated as we learn more. You may also find some helpful information under the heading X-terminals from the CD home page; see *Misc. Information - Hints, Notes, User Experiences, Etc.*

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	Open 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	Open from the toolbar or File menu; type in the new file name
Include a file	Insert File from the File menu
Select text	Use the mouse
Highlight special syntax in color (for use with language modes)	Syntax Highlighting from Options menu.
Cut/Copy/Paste text	Cut/Copy/Paste from the toolbar or Edit menu
Search for text	Search from the Edit menu
Spell check	Spell from the toolbar or Edit menu.
Save the file	Save from the File menu
Close the file	Delete Buffer from the File menu (use the Buffer menu to select the buffer to close)
Exit **xemacs**	Exit EMacs from the File menu

All standard **emacs** commands can also be used. In particular, if you call up one of the bells and whistles that you didn't really want, use:

C-g	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 seven object-oriented languages (Eiffel, 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[1] 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.

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.

11.3.3 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**, include in your login script or enter:

```
% setup nedit
```

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

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	**`Open`** from the File menu; choose an existing file from popup window
Create a new file	**`New`** from the File menu
Include a file	**`Include`** from the File menu
Select text	Use the mouse, or the shift and arrow keys together
Cut/Paste text	**`Cut/Paste`** from the Edit menu
Search for text	**`Find`** from the Search menu
Fill paragraph	**`Fill Paragraph`** from the Edit menu
Spell check	**`spell`** from the Shell menu
Save the file	**`Save`** from the File menu
Close the file	**`Close`** from the File menu (you are prompted about saving)
Exit **NEdit**	**`Exit`** 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.

11.3.4 nu/TPU

nu/TPU is an excellent port of Digital's TPU (Text Processing Utility) programming language to UNIX. The major component in **nu/TPU** is software that can be used to build new text processors and batch-oriented text manipulation routines. **nu/TPU** is distributed with two editing interfaces:

- SI (Simple Interface), an **EDT**-style keypad map. This should be sufficient for users who do minimal editing and wish to use an **EDT**-style keypad.
- a clone of the old **EVE** interface (from VMS V5.0) complete with the EVE_functions (**but not the EVE$functions!**) from that version. This will probably not be satisfactory to users familiar with the later version of **EVE** (VMS V6.1), since it lacks many of the enhancements and fault corrections.

The product **tpu** needs to be installed prior to use. To set up **nu/TPU,** include in your login script or enter:

`% setup tpu`

To edit using the SI interface by default, type:

`% tpu [`*filename*`]`

To edit using the older **EVE** interface, enter:

`% tpu -section=tpusec.ini [`*filename*`]`

Once you are in the editor, on-line help is available via the PF2 key or the Help command.

For complete documentation on **nu/TPU**, including documentation on all command-line options and all **nu/TPU** programming statements, you can order the *nu/TPU Reference Manual.* Look under **Editors** on the *UNIX Resource*s Web page for documentation on the **TPU-edt** editor for GNU Emacs.

11.3.5 fermitpu

The **fermitpu** product contains a locally developed port of Digital's **EVE** editing interface from VMS V6.1 (the version of VMS which is current during the VMS to UNIX migration). Note that this is not the same as the VMS product **fermitpu**, which contains many enhancements beyond the VMS V6.1 **EVE** editor. Users who are familiar with programming in **EVE** and **TPU** will probably be most comfortable with this editing interface.

The product **fermitpu** needs to be installed prior to use. To invoke, include in your login script or enter:

`% setup fermitpu`

`% tpu [options] [`*filename*`]`

Once you are in **fermitpu**, the on-line help (accessible via PF2 or the Help command) contains documentation on all of the EVE callbacks and TPU built-ins.

Look under **Editors** on the *UNIX Resource*s Web page for documentation on the **TPU-edt** editor for GNU Emacs.

11.3.6 EDT+

This is a clone of Digital's **EDT** which includes the enhancement to remove the hard-coded 24-line display limit. The product **edt** needs to be installed. To setup **EDT+**, include in your login script or enter:

`% setup edt`

To invoke for editing within the same terminal window, type:

`% edt [`*options*`] [`*filename*`]`

To edit in a separate display window (requires X), type:

`% xedt [`*options*`] [`*filename*`]`

EDT+ allows both line mode and full screen editing. Full screen editing employs a keypad, which is located on the right-hand side of your keyboard. If you are in line mode when you first enter **EDT+**, simply type `c` to change to full screen mode. Once in full screen mode, you can start entering text, and your keypad is available for commands.

To return to line mode, enter <Ctrl-z>. At the prompt (`*`) you can type `exit` to exit and save, or `quit` which exits without saving the file updates.

Remember that when you're at the UNIX command prompt <Ctrl-z> is used as the suspend character.

Further documentation is available from the man pages, and from the vendor Boston Business Computing, Ltd.

Look under **Editors** on the *UNIX Resource*s Web page for documentation on the **EDT** emulation package for GNU Emacs.