

TEN/PLUS* User Interface
Guide

INTERACTIVE

product family

INTERACTIVE
.....
A Kodak Company

First printing (March 1990)

No part of this manual may be reproduced in any form or by any means without written permission of:

INTERACTIVE Systems Corporation
2401 Colorado Avenue
Santa Monica, California 90404

© Copyright INTERACTIVE Systems Corporation 1985-1990

RESTRICTED RIGHTS:

For non-U.S. Government use:

These programs are supplied under a license. They may be used, disclosed, and/or copied only as permitted under such license agreement. Any copy must contain the above copyright notice and this restricted rights notice. Use, copying, and/or disclosure of the programs is strictly prohibited unless otherwise provided in the license agreement.

For U.S. Government use:

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR Section 52.227-14 (Alternate III) or subparagraph (c)(1)(ii) of the clause at DFAR 252.227-7013, Rights in Technical Data and Computer Software.

All rights reserved. Printed in the U.S.A.

The *termcap* and *curses* code and documentation were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley Campus of the University of California under the auspices of the Regents of the University of California.

The following trademarks shown as registered are registered in the United States and other countries:

INed and TEN/PLUS are registered trademarks of INTERACTIVE Systems Corporation.

UNIX is a registered trademark of AT&T.

AT&T Personal Computer 6300 is a trademark of AT&T.

DEC, VT100, VT220, and VT240 are trademarks of Digital Equipment Corporation.

Esprit 6310 is a registered trademark of Esprit Systems, Inc.

HP LaserJet II is a registered trademark of Hewlett-Packard Company.

AT and IBM are registered trademarks of International Business Machines Corporation.

CoEdit and LPI are trademarks of Language Processors, Inc.

Epson is a registered trademark of Seiko Epson Corporation.

WYSE 60 is a registered trademark of Wyse Technology.

Programs described in this manual are copyrighted and their copyright notices may be found in heralds, by using the UNIX System *what* program, and by reading files whose names start with "coprisc".

TEN/PLUS User Interface Guide

CONTENTS

Introduction to the TEN/PLUS User Interface

TEN/PLUS User Interface Release Notes

TEN/PLUS Primer

TEN/PLUS Tutorial

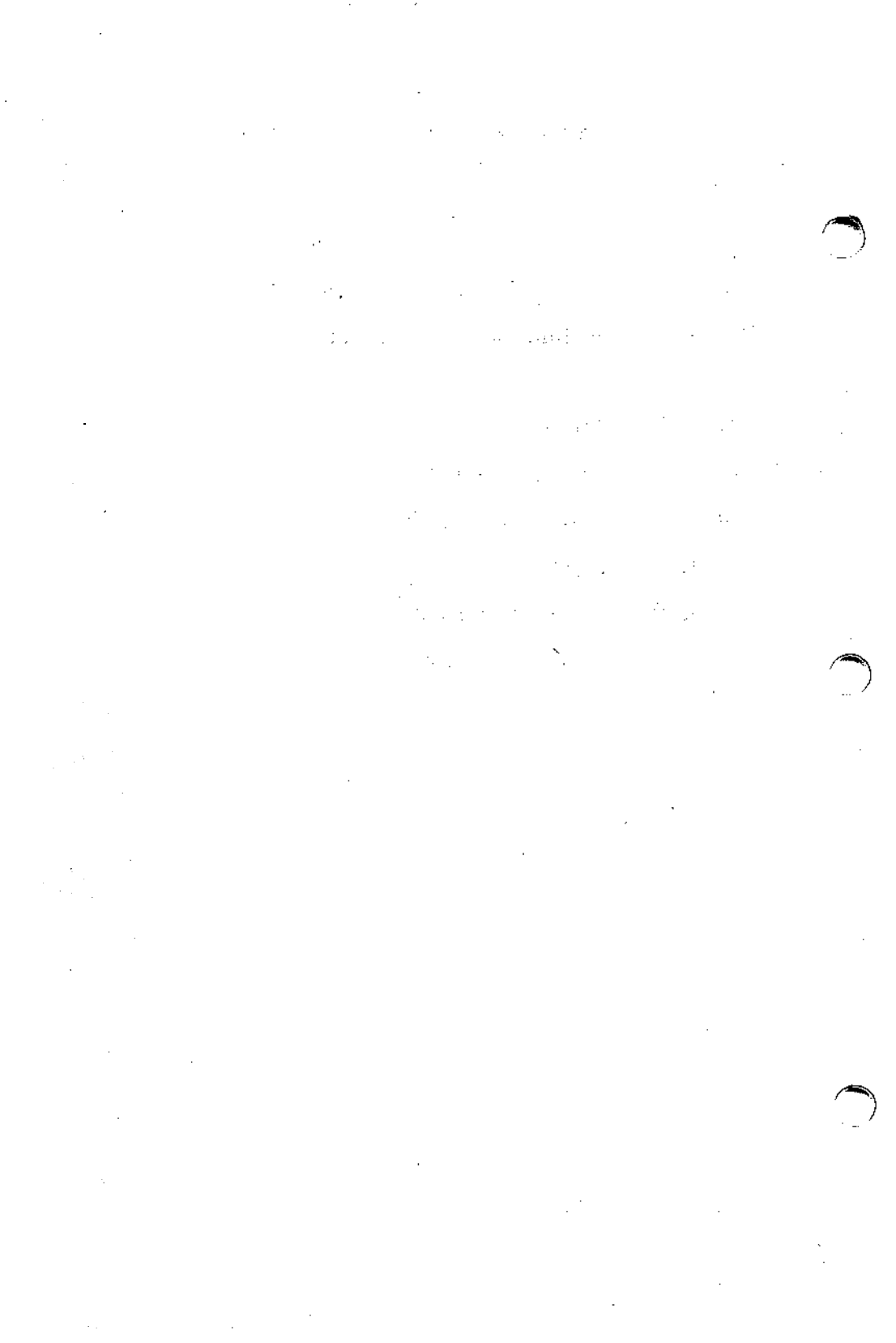
TEN/PLUS Reference Manual

TEN/PLUS Keyboard Information

TEN/PLUS Profiles

TEN/PLUS User Interface Manual Entries

TEN/PLUS User Interface Installation Instructions



Introduction to the TEN/PLUS* User Interface

Welcome to the *TEN/PLUS User Interface Guide*. This guide contains the documentation you need to install, use, and customize the TEN/PLUS User Interface. Before you begin to use the TEN/PLUS system, be sure to read the next few pages of this document. They will tell you what documents are contained in this guide and how to use the guide to your best advantage.

WHAT'S INCLUDED

The *TEN/PLUS User Interface Guide* includes:

- **TEN/PLUS User Interface Release Notes**
Provides a description of the current release of the TEN/PLUS User Interface.
- **TEN/PLUS Primer**
Provides a tutorial introduction on how to create and edit files and directories and how to use the basic TEN/PLUS functions.
- **TEN/PLUS Tutorial**
Provides a detailed training guide for beginners who wish to learn how to use the TEN/PLUS system. The tutorial describes how to use three main features of the TEN/PLUS User Interface: the INed* editor, the File Manager, and the History Display.
- **TEN/PLUS Reference Manual**
Provides a detailed description of the components and functions of the TEN/PLUS User Interface.
- **TEN/PLUS Keyboard Information**
Describes how the TEN/PLUS functions are mapped to your keyboard.
- **TEN/PLUS Profiles**
Provides a training guide for users who wish to learn how to use TEN/PLUS profiles to customize the TEN/PLUS system.

- **TEN/PLUS User Interface Manual Entries**
INTERACTIVE's proprietary manual entries, which supplement the *INTERACTIVE UNIX System V/386 Release 3.2 User's/System Administrator's Reference Manual*.
- **TEN/PLUS User Interface Installation Instructions**
Provides step-by-step instructions on how to install and configure the TEN/PLUS User Interface.
- **Reader's Comment Form**
Provides you with a way to tell us what you like or dislike about this guide and to send us your ideas for making it even better.

WHERE TO BEGIN

The *TEN/PLUS User Interface Guide* includes a variety of documents for users at varying levels of experience. Depending on your level of experience, you may want to use this guide in a number of different ways. The outline below provides some suggested ways to use this guide:

- **If you are a beginner . . .**
Read through the "TEN/PLUS Primer" or the "TEN/PLUS Tutorial." If you want a quick introduction to the basic TEN/PLUS functions, try the "TEN/PLUS Primer." If you would like a more in-depth introduction to the TEN/PLUS system, work through the "TEN/PLUS Tutorial."
- **If you are an experienced TEN/PLUS user . . .**
Read the "TEN/PLUS Reference Manual," which offers concise descriptions of the TEN/PLUS functions and how to use them; or you may wish to refer to "TEN/PLUS Profiles" to learn how to customize your TEN/PLUS environment.
- **If you are installing the system . . .**
Read and follow the steps outlined in the "TEN/PLUS User Interface Installation Instructions." Once you have completed the basic system installation, refer to "TEN/PLUS Profiles" to learn how to customize your TEN/PLUS environment.
- **If you want the latest system information . . .**
Refer to the "TEN/PLUS User Interface Release Notes" for a listing of the the newest TEN/PLUS features.

CONVENTIONS USED

Numbers preceded by the symbol § refer to section numbers within that document.

OVERVIEW OF THE TEN/PLUS USER INTERFACE

The TEN/PLUS User Interface is a fully integrated user environment that makes INTERACTIVE UNIX* System V/386 Release 3.2 easy to learn and easy to use. The TEN/PLUS User Interface provides ten basic functions, plus a number of more advanced functions. The basic functions allow you to perform most tasks simply and productively, while the advanced functions permit you to gradually learn more complex tasks.

The TEN/PLUS User Interface runs on a wide range of computers including personal computers, multi-user micros, minis, and mainframes. You obtain the maximum power from your computer system in a way that is easy to use. The TEN/PLUS functions are consistent, which means it is not necessary to learn a new set of commands each time you use a new application or try a new task. The same command will have the same result, regardless of whether you are reading a mail message, writing a computer program, or updating your calendar. On some systems, TEN/PLUS functions are invoked solely through keys on the keyboard. On other systems, some TEN/PLUS functions and operations can be invoked by using a mouse. If your system is equipped with a mouse, consult the appropriate operating manual for specific instructions.

TEN/PLUS USER INTERFACE FEATURES

The TEN/PLUS User Interface consists of six components: the INed editor, the File Manager, the Profile Helper, the Print Helper, the History Display, and the C Helper. These are used to create, edit, and manage text files. Together, they provide a bridge to the INTERACTIVE UNIX System, allowing users to perform basic functions without having to learn a lengthy set of UNIX System commands.

All functions that can be performed from the UNIX shell can be performed using one of the TEN/PLUS User Interface components. Since all six components use a system of menus and functions that allow the user to select a desired operation, it is rarely necessary to

remember a specific UNIX System command. The six components of the TEN/PLUS User Interface are described below.

INed Editor

INed is a screen-oriented text editor that allows users to display and edit text files. Users enter and edit text by typing on the display as they would on a typewriter. The display contains a window that can be divided into smaller windows for editing and examination of files. This feature can be used in conjunction with other functions to “pick up” text from one file and “put” it into another file or another portion of the same file. The INed editor provides a variety of functions to insert, delete, and move text on the screen. Other INed features include text processing, paragraph fill, right margin justification, and global replacement.

File Manager

The TEN/PLUS File Manager allows users to create, access, move, copy, and delete files. The File Manager uses the same functions as the INed editor. Files are picked up and moved, or copied and moved, using the same INed functions that perform these operations on lines of text. The File Manager is easy to use because there are no UNIX commands to remember.

Profile Helper

The TEN/PLUS Profile Helper allows users to customize the editing environment to suit individual needs. It can be used to help the editor locate forms, helpers, messages, and forms language scripts, as well as to add, change, or delete the options on certain menus. The Profile Helper uses forms to build custom menus that simplify operations in the TEN/PLUS environment. Custom menus allow the user to perform routine tasks by selecting options from menus.

Print Helper

The TEN/PLUS Print Helper allows users to print a file by selecting an option from the Print Menu. The options on the Print Menu can be customized by editing the print profile. Like the Profile Helper, the Print Helper uses forms to build the custom Print Menu. The custom Print Menu simplifies printing operations in the TEN/PLUS

environment by giving the user a choice of menu options for printing a file.

History Display

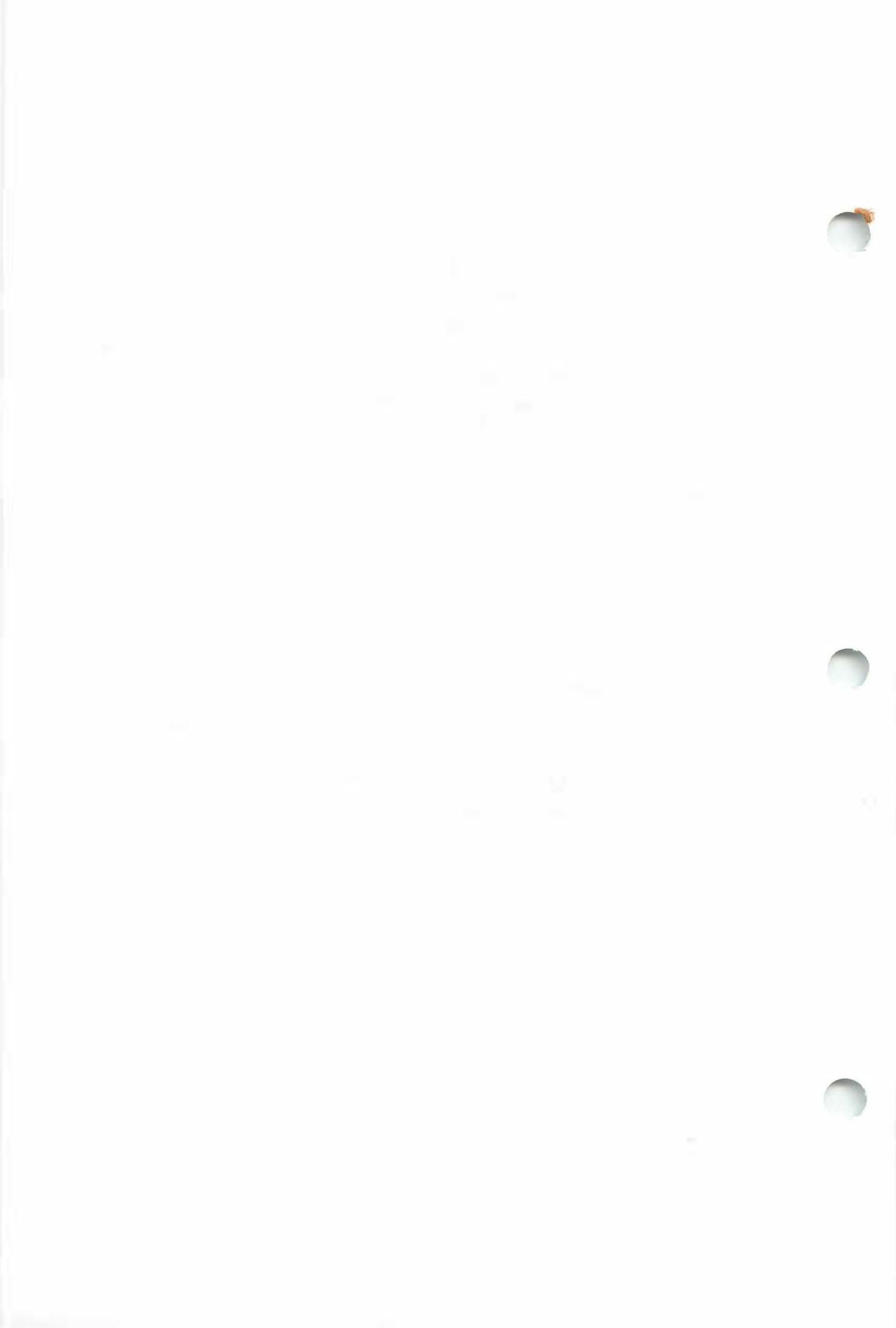
The TEN/PLUS History Display provides a way to keep track of changes made to structured files. It displays a form detailing information about previous changes to a file and allows the user to view and/or copy any previous version. The History Display uses menus and forms to implement its functions. History Display functions are accessed by positioning the cursor at the desired option on the menu or form and using a function. Like the File Manager, it does not require knowledge of any specific commands.

C Helper

The TEN/PLUS C Helper allows a programmer to develop, compile, and test C programs. It uses the make utility and will only function properly if the INTERACTIVE Software Development System is installed.

FOR MORE INFORMATION

The TEN/PLUS User Interface is supported by a complete set of documentation. For a complete listing of all documentation that relates to the INTERACTIVE UNIX Operating System, refer to the “Documentation Roadmap” included in the *INTERACTIVE UNIX System V/386 Release 3.2 Operating System Guide*.




TEN/PLUS User Interface
Release 2.2.5
Release Notes

CONTENTS

1. INTRODUCTION	1
2. SUPPORT FOR JOB CONTROL	1
3. NEW OPTION AND XPCTERM SUPPORT	2
4. NEW TEN/PLUS FORMS FEATURES	3
4.1 TEN/PLUS Forms	3
4.2 Forms Compiler	3
4.3 Support for Large Screens	3
5. NEW ENVIRONMENT FOR C PROGRAMMERS	4
5.1 C Programming Helper	4
5.2 New Filters	4
6. NEW UTILITIES FOR STRUCTURED FILES	5
7. NEW TERMINAL DESCRIPTIONS	6
8. TRANSLATABLE HELP MESSAGE FILES	6





TEN/PLUS* User Interface


Release 2.2.5

Release Notes

March 1990


1. INTRODUCTION

We are pleased to provide Release 2.2.5 of the TEN/PLUS User Interface. This is a new release for the INTERACTIVE UNIX* Operating System and it runs on Intel * 386*- and 486*-based computers. It is also the first release of the TEN/PLUS editor to be bundled with the TEN/PLUS Mail System. These release notes reflect the changes for the TEN/PLUS User Interface only. For details on the TEN/PLUS Mail System, refer to the *TEN/PLUS Mail System Guide*. New features of the TEN/PLUS User Interface are documented here and include:

- 
- Support for job control
 - New `xpcterm` support
 - Inclusion of a TEN/PLUS forms compiler
 - Support for larger screens
 - Inclusion of a C programming helper
 - Support for new filters
 - New utilities for structured files
 - New terminal descriptions
 - Translatable help message files

These release notes describe the purpose and use of each new feature.

2. SUPPORT FOR JOB CONTROL



A new feature of the INTERACTIVE UNIX Operating System is job control, which makes it possible to put a foreground process in the background and return it to the foreground at a later time.

This facility provides a convenient way to switch between applications on the same terminal or screen without having to exit one

application and enter another one. The TEN/PLUS editor now properly supports the job control feature.

To use the job control feature, make sure that you are running the C shell and that a `suspendcharacter` is set. By convention, the `suspendcharacter` is often set to be `CTRL Z`. To set the `suspendcharacter` to be `CTRL Z` in the TEN/PLUS User Interface, type:

```
stty '^z'
```

To move a foreground process to the background, simply type `CTRL Z`.

For example, to temporarily interrupt a TEN/PLUS editor session, type `CTRL Z`. The TEN/PLUS editor will clear the screen and the C shell prompt will appear. To return to the TEN/PLUS editing session, use the C shell `fg` command. The TEN/PLUS editor will reappear exactly as you left it.

3. NEW OPTION AND XPCTERM SUPPORT

A PC terminal emulator program called `xpcterm` is a part of the INTERACTIVE X11 product that allows users to run text-based applications, such as the TEN/PLUS environment, in an X window. When TEN/PLUS is used in an `xpcterm` window, the editor recognizes the size of the window (in columns and lines). This means that the editor session window will have the exact same size and shape as that of the window it is running in.

When a window size is changed, the window size of the TEN/PLUS editing session can be readjusted by using the new `ENTER REFRESH` command. A new command line option, `-ar`, has been introduced to make this readjustment happen automatically. If this option is used when the TEN/PLUS editor is invoked, the editor session window will automatically adapt itself to the new window size each time it is changed.

The same command and option can be used when TEN/PLUS is invoked in an INTERACTIVE MultiView window. For consistency, the command line option is now named `-ar` instead of `-autoresize`.

4. NEW TEN/PLUS FORMS FEATURES

4.1 TEN/PLUS Forms

TEN/PLUS makes extensive use of forms that define the layout of the screen. A TEN/PLUS form consists of two parts: an area for invariant text and an area for data entry. The part of the form that contains invariant text usually contains items such as descriptive text and titles. When a file is created using a TEN/PLUS form, the invariant text cannot be modified. The other part of the form is to enter data. Modifications can be made to this part of a file created with a TEN/PLUS form.

4.2 Forms Compiler

The forms compiler is a program that compiles a TEN/PLUS source form into a more compact binary file that is used by the TEN/PLUS editor. A TEN/PLUS source form can be edited and translated so that all of the invariant text appears in the user's own language. Source forms continue to be supplied with the TEN/PLUS User Interface, and the forms compiler is now also part of the User Interface. This allows the user to make compiled versions of translated forms. Refer to *fc(1)* for information about how to use the forms compiler.

4.3 Support for Large Screens

When forms are used on screens larger than they were intended for, problems may occur. For example, if a TEN/PLUS menu is accessed near the corner of the window and **CANCEL** is used, parts of that menu may remain visible because the editor only cleans up the screen as far as the form goes. To correct the problem, recompile the form for the new screen size. This is only necessary for forms that contain a great deal of invariant text (such as the one for the editor profile file).

The dimensions of the screen can be specified in the forms profile file, `frmprf`, which is located in the directory `/usr/lib/INed/profiles`. For example, if you always use `xpcterm` with 80 by 45 windows, specify 45 lines in your personal `frmprf` and recompile the problem forms. (Do not recompile the standard form `std`.) Store the forms in a personal directory and specify that directory in your `editorprf` file.

5. NEW ENVIRONMENT FOR C PROGRAMMERS

The TEN/PLUS User Interface includes new features that will increase the productivity of C programmers who use the TEN/PLUS editor. These features include a C Programming Helper and a set of utilities.

5.1 C Programming Helper

The C Programming Helper is automatically invoked when a C source file (typically a file that has a name ending in .c or .h) is edited. It allows a programmer to edit, compile, and test programs without leaving the TEN/PLUS editor. It can only be used with the AT&T compiler that comes with the INTERACTIVE Software Development System because it relies on the format of the error messages produced by that compiler. Developers who use the LPI* ANSI C compiler should use CoEdit* instead. The C Helper is described in more detail in the “TEN/PLUS Tutorial” in this guide.

5.2 New Filters

The TEN/PLUS User Interface Release 2.2.5 includes some new filters that are particularly useful to C programmers. The new filters are box, unbox, space, unspace, indent, undent, tab, and untab.

The `box` and `unbox` filters can be used to put C-style comments around text and remove it again. For example, suppose you have the following text in your file:

```
for (i = 0 ; i < total ; i++) {
    printf ("%7d",values[i]);
    if (((i + 1) % 10) == 0)
        printf("\n");
}
```

Position the cursor on the line that contains the word `for`. Press the **ENTER** key, type the word `box`, then use **DO**. The text will now look something like this:

```
/******  
/*          for (i = 0 ; i < total ; i++) {          */  
/*              printf ("%7d",values[i]);          */  
/*              if (((i + 1) % 10) == 0)          */  
/*                  printf("\n");          */  
/*          }          */  
/******
```

Refer to *filters(1)* for more information about these filters.

6. NEW UTILITIES FOR STRUCTURED FILES

The TEN/PLUS environment, particularly the TEN/PLUS Mail System, makes use of TEN/PLUS structured files. These are index sequential files rather than straight text files.

Prior to this release, printing the information stored in a TEN/PLUS structured file could be problematic. This was because it was only possible to print one record at a time and in the same format as the information was displayed on the screen. With release 2.2.5, three new utilities (`catsf`, `sortsf`, and `prtsf`) are provided to remedy this problem. These utilities are described briefly below. Refer to the appropriate manual entries for more details about these utilities.

The `catsf` program creates a new TEN/PLUS file that contains records from one or more existing TEN/PLUS files, as selected by the user. For example, a user can create a new mailbox file containing only those messages that were sent to him by user `angela`.

The `sortsf` program sorts the records in a TEN/PLUS file according to the values of the field specified by the user. Its behavior is quite similar to that of the UNIX* System `sort` program. Users can sort their mailboxes by items such as subject or date.

The `prtsf` program is a simple report generator. It can be used to prepare documents (or printouts when piped into the `lp` command). Users can specify which text should go into the document and where the text should be inserted. Documents can be generated using text from all records in the file or only those records selected by the user.

Here is an example of a format description that can be used to generate a document that lists all mail messages received on March 7:

```
#h Mail messages received on March 7 1990
#h Document prepared on {=DATE}
Message send by {FromTo}
Subject: {Subject}
Total message follows below:
```

```
[Message/*
```

```
]
```

With the TEN/PLUS editor, it is possible to draw boxes (as for diagrams) in the middle of text. Because these drawing characters are stored as control characters, most printers need an IBM* 8-bit code to print them. However, the TEN/PLUS print helper has an

option that converts these control characters into printable characters.

To use the IBM codes, a sample filter is supplied called `prtfilter`. This shell script is stored in the file `/usr/lib/INed/bin`. As supplied, it outputs line drawing characters from the IBM extended graphics character set. It can be easily modified for compatibility with Epson* or other printers. It has been tested with several dot-matrix printers in the IBM Graphics Printer compatibility mode and with the HP LaserJet II* in its default mode. The `printhelp` file contains examples of how to use this filter.

7. NEW TERMINAL DESCRIPTIONS

Three new terminal descriptions are provided, in addition to the already existing description for the Wyse 60* terminal. Wyse 60 terminals are available with different keyboard types. If you set your TERM environment variable to `wyse-en`, `wyse-at`, or `wyse-as`, you can use TEN/PLUS with any of the three most popular keyboards. These are the Enhanced AT*, original AT, and ASCII keyboards. The `wyse-en` variable differs from the original variable in that it uses `[ESCAPE]` instead of `[CTRL] [z]`.

Keyboard layouts are listed in the `keys.map` file in an easy to understand format. To access that file, simply use `[HELP]` and choose `keyboard layouts`.

An `ansi` terminal description is provided for ANSI terminal emulators. This is specifically designed for users of MultiView Desktop. It should work with any terminal that has a fully functional ANSI-compatible mode and a PC-style keyboard.

8. TRANSLATABLE HELP MESSAGE FILES

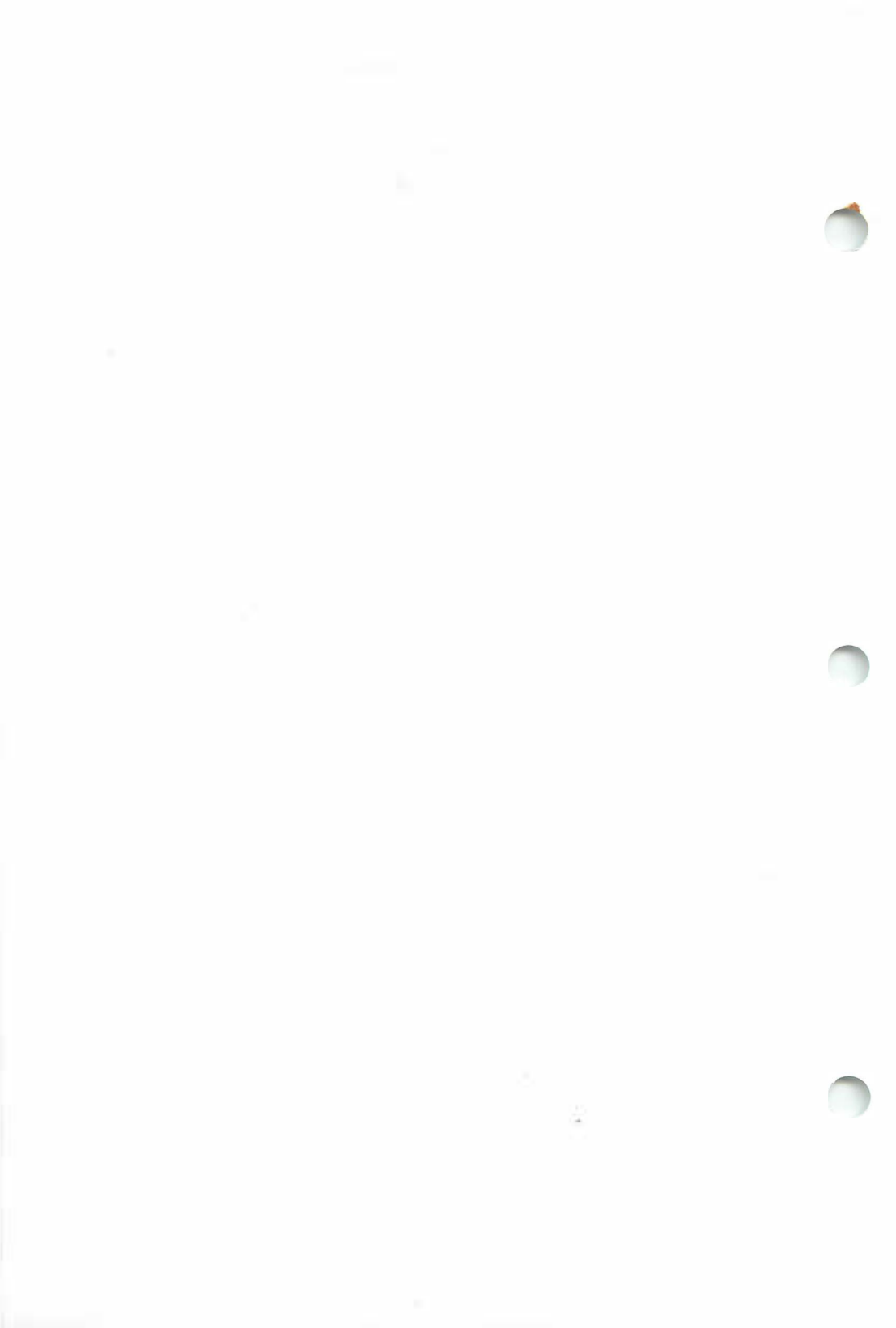
The TEN/PLUS product can be translated into another language without the source code. To do this two things need to be translated:

- The TEN/PLUS forms
- The TEN/PLUS help message files

Forms that have sources in `/usr/lib/INed/srcforms` can be copied, edited, translated, and compiled with the forms compiler, `fc`. The resulting files (all having a name ending in `.ofm`) can then be installed in `/usr/lib/INed/$LANG/forms`.

LANG is a shell variable that is recognized by the TEN/PLUS editor. It should be set to the language desired.

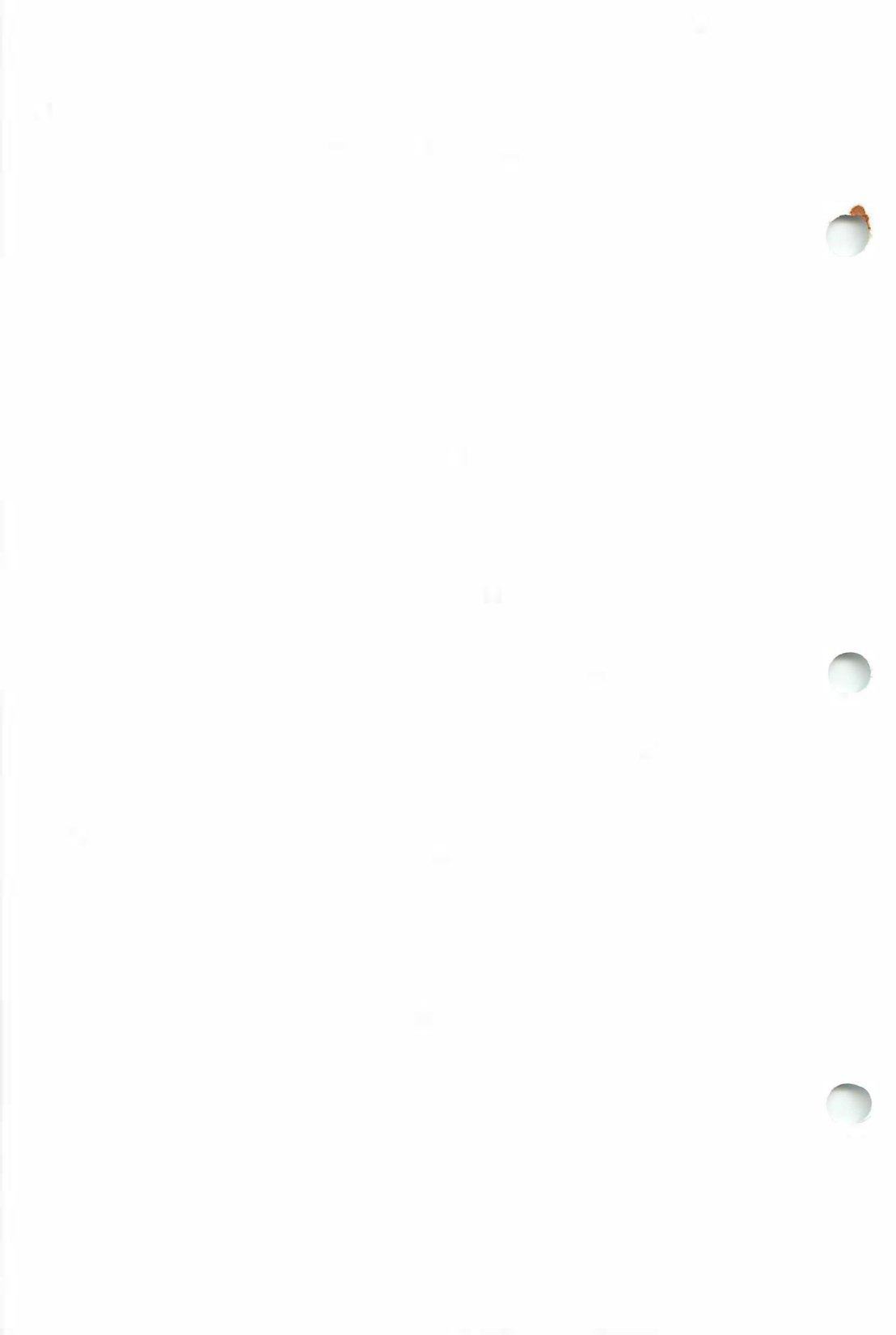
In a similar way, the TEN/PLUS help messages can be translated. All help message files are found in `/usr/lib/INed/hmgs`. Release 2.2.5 of the TEN/PLUS User Interface now contains the necessary forms to do this, as well as a help message helper. The use of this helper is self-explanatory.



TEN/PLUS Primer

CONTENTS

INTRODUCTION	1
What Will I Learn From This Primer?	1
GETTING STARTED	2
Your Home Directory	2
Using ZOOM-IN and ZOOM-OUT	4
Using the Cursor-Positioning Functions	4
CREATING DOCUMENTS	5
USING HELP AND CANCEL	8
CREATING DIRECTORIES	9
EDITING FILES AND DIRECTORIES	11
Using PICK-UP and PUT-DOWN	11
Using PICK-COPY and PUT-COPY	13
Using FORMAT and INSERT	15
USING MENUS	18
Using MENU	18
Using LOCAL-MENU	19
MORE ABOUT EDITING	22
Changing Margins and Tabs	22
Alternating Between Insert and Overwrite Modes	23
Using +SEARCH, -SEARCH, and BREAK	23
Using USE	24
Printing Documents	24
FOR MORE INFORMATION	25
SUMMARY OF TEN/PLUS FUNCTIONS	26
The Ten Basic Functions of the TEN/PLUS System	26
Some Additional TEN/PLUS Functions	27
GLOSSARY	31





TEN/PLUS* Primer



INTRODUCTION

What Will I Learn From This Primer?

This primer will introduce you to the ten basic *functions* of the TEN/PLUS environment, and will show you how to:

- Create and store documents, such as memoranda, letters, reports, and tables.
 - Retrieve documents.
 - Revise documents.
 - Set up new filing systems.
 - Perform a number of complex tasks, such as moving information between documents.
 - Print documents.
- 
- 

GETTING STARTED

To use the TEN/PLUS system, you must turn on your computer or terminal and *log in*. At the `login:` prompt, you will type in a user identification that has been assigned to you, followed by **ENTER**. After your user identification has been received by the system, you may be prompted for a password. If you have been assigned a password, type it at the `Password:` prompt, followed by **ENTER**. (The system will not display the password on the screen.) If you have not been assigned a password, press **ENTER** if you see a `Password:` prompt.

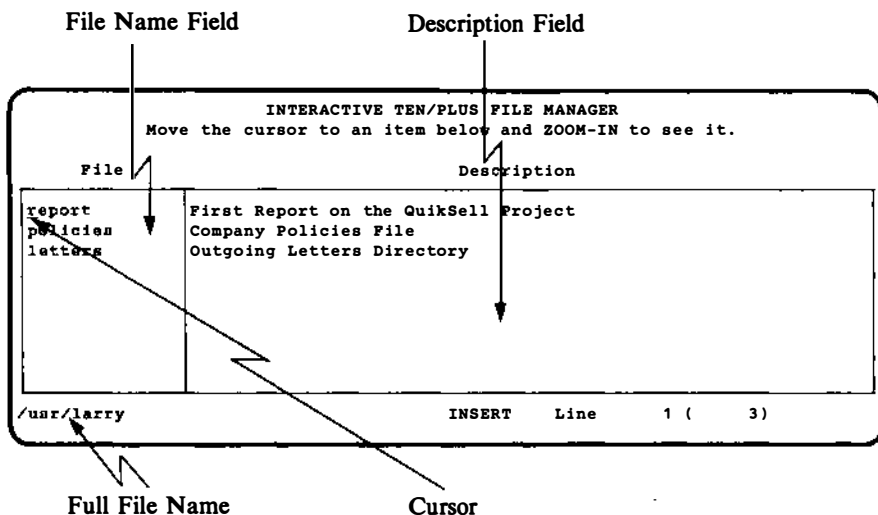
Your system may be installed so that an TEN/PLUS screen will automatically appear when you log in. If the system displays a prompt such as `$` or `%` when you log in, access the TEN/PLUS environment by typing `e $HOME`, followed by **ENTER**. If you are still unable to enter the TEN/PLUS environment, ask the person responsible for installing the system to help you.

Place your *keyboard template* in a convenient location so you can refer to it as you read this primer. If you do not have a keyboard template, see the person responsible for installing the TEN/PLUS system.

Your Home Directory

The first screen you see when you log in to the system is usually your *home directory*. You can think of your home directory as a master list or index of the information you are storing on the computer. A *directory* is similar in function to a file cabinet: a file cabinet has several drawers, each of which can contain many folders, each holding one or more documents. This is very useful for organizing your files.

Your home directory can contain subdirectories (folders) that hold related files (documents) and other subdirectories. For example, you can have a subdirectory called `letters`, which contains all the outgoing letters you've written. The `letters` subdirectory can also contain a number of other subdirectories, one for each of your major projects. For example, there could be a `quik.sell` subdirectory under `letters`, which contains all the letters related to the QuikSell project. Here is a typical home directory listing:



If you are a new user, your home directory may not show any *files* because you have not yet created any. Take a moment to study the display shown above, paying particular attention to the labels identifying specific elements on the screen. (Some screens may differ in minor ways.)

The *cursor* is a pointer to where the next character will appear. As you type, each character appears at the cursor position, and the cursor moves one space to the right. Entering text is similar to typing with a typewriter. However, in most cases in the TEN/PLUS system, when you reach the right-hand margin, the cursor and the word being typed move automatically to the next line. This feature is called *word wrap*. You can, of course, end a line before you reach the right-hand margin by pressing **ENTER**. **ENTER** moves the cursor to the beginning of the next line.

The *File field* contains the names of the files and directories stored in the displayed directory. A file usually contains a single document, such as a report or a memorandum. A directory is also a file, but it is a special type of file that can contain other files.

The *Description field* allows you to supply a short description of the file or directory.

You should limit file and directory names to 10 characters. File names can contain any characters other than], [, *, ?, /,

and space. File names must not start with a hyphen (-) or a plus (+).

The *full file name* completely describes the location of the file in the system. Slashes are used to separate directory names and filenames. The full file name is displayed at the bottom of the screen and when you attempt to create a new file.

Using ZOOM-IN and ZOOM-OUT

You can use the TEN/PLUS functions **ZOOM-IN** and **ZOOM-OUT** to move around within a directory structure. **ZOOM-IN** moves you to a lower level in the directory structure and **ZOOM-OUT** moves you to a higher level. To move from a directory to a file or directory in the next lower level, position the cursor on the line on which the file or directory is listed and **ZOOM-IN**. To move from a file or directory to the directory in the next higher level, **ZOOM-OUT**. It does not matter where the cursor is positioned in the file or directory when you **ZOOM-OUT**.

Using the Cursor-Positioning Functions

The *cursor-positioning functions* move the cursor on the screen. In addition to **ENTER** (discussed in a previous subsection), the most frequently used cursor-positioning functions are **↓**, **↑**, **→**, and **←**; they move the cursor down, up, right, and left, respectively. You can also use **BACKSPACE** to move the cursor to the left, but, unlike **←**, **BACKSPACE** erases characters as it moves. **BACKSPACE** is used to correct typing errors.

TAB moves the cursor to the next tab stop on the right, while **-TAB** moves the cursor to the previous tab stop on the left. On most terminals, the cursor-positioning functions repeat automatically. Holding down the key(s) for any of these functions will result in continuous cursor motion until the keys are released. You can use the cursor-positioning functions to place the cursor anywhere on the screen, including on any previously typed character, as well as on the borders of any field. (If you place the cursor on such a border and attempt to type there, you'll get an error indication, usually a "beep.")

CREATING DOCUMENTS

You can create documents (also called files) in any of your directories, including your home directory. Suppose you want to create a file called `status`, with the description `Current Sales Status`, and that you want this file to be in your home directory. To create the file, move the cursor to the first blank line on your home directory screen. Type the file name, `status`, in the File field. Use `TAB` or one of the other cursor-positioning functions to move the cursor to the Description field, then type `Current Sales Status`:

```

      INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

      File                                     Description
-----
status                               Current Sales Status_

/usr/larry                            INSERT   Line   1 ( 1 )
```

Now refer to your keyboard template to locate the key or keys used to `ZOOM-IN`, and then `ZOOM-IN`. The TEN/PLUS system determines that a file named `status` does not exist in your home directory, and shows you a menu with four options:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
status	Current Sales Status
<p>You are attempting to create file "/usr/larry/status"</p> <p>Move the cursor to the type of file you want and touch EXECUTE. Touch CANCEL to do nothing, HELP for help.</p> <p>Create an ASCII file (without history) Create a structured file (with history) Create a directory Re-enter the file name</p>	
/usr/larry	INSERT Line 1 (1)

The options shown allow you to create an *ASCII file*, a *structured file*, or a directory, or to reenter the file name in case you mistyped it. Usually, you will create ASCII files, which are ordinary text files. The examples in this primer use only directories and ASCII files. Structured files allow users to keep a record of different versions of a document, and to recreate any prior version. Structured files are particularly useful when you need to look at past versions of a document, such as a legal contract or a business plan. Many TEN/PLUS applications, such as the TEN/PLUS Mail System, use structured files. (Refer to section 2 of the "TEN/PLUS Reference Manual" in this guide for more information about file types.)

With the cursor on the option Create an ASCII file (without history), use **EXECUTE**. After a brief pause, a blank window appears:

```

1      t      t      t      t      t      t      t      t      t      r
-
/usr/larry/status                INSERT   Line   1 (   0)

```

You have just created a file with the short name `status`. The line at the top of the screen is a “ruler” showing the positions of the left- and right-hand margins (`l` and `r`) and of the tab stops (`t`). (Instructions for changing margins and tab stops appear later on in this primer.)

Type some text into your new file. When you complete the last line on the screen, the text will *scroll* forward (up), bringing in additional blank lines at the bottom of the screen. You can also scroll forward by using either `+PAGE` or `+LINE` to move beyond the area displayed on the screen. `+PAGE` scrolls text forward by one screen, and `+LINE` scrolls text forward by approximately one-third of a screen. Similarly, `-PAGE` and `-LINE` scroll text backward (down).

Use `ZOOM-OUT` to move from the `status` file back to your home directory. It does not matter where the cursor is positioned in the file when you `ZOOM-OUT`.

USING HELP AND CANCEL

HELP provides details about menus and *popup boxes*. Use **HELP** whenever you are uncertain about what to do. Either a popup box with additional instructions or a menu of options will appear on your screen.

CANCEL allows you to remove menus and popup boxes from your screen. For example, if you decide that you do not want to create a new file, use **CANCEL** when the file creation menu appears. This causes the menu to be removed from the screen.

To see how **HELP** and **CANCEL** work, **ZOOM-IN** to the status file, and then **ZOOM-IN** again. The message **Cannot zoom in any further** appears in a popup box. Use **HELP**. The error message is explained in more detail in another popup box. Use **CANCEL** to remove the popup boxes.

CREATING DIRECTORIES

You can create directories using the same procedure that you used to create new files. Create a directory called `practice` in your home directory. Type `practice` in the File field of your home directory, then **TAB** to the Description field and type `Practice Files Directory`:

```

      INTERACTIVE TEN/PLUS FILE MANAGER
      Move the cursor to an item below and ZOOM-IN to see it.

      File                               Description
-----
status      Current Sales Status
practice    Practice Files Directory_

/usr/larry                                INSERT   Line    2 (    2)

```

Now **ZOOM-IN**. The file creation menu appears. Select the option to create a directory. After a brief pause, a blank directory screen appears:

```

      INTERACTIVE TEN/PLUS FILE MANAGER
      Move the cursor to an item below and ZOOM-IN to see it.

      File                               Description
-----
-

```

`/usr/larry/practice` INSERT Line 1 (0)

Now you can type in the names and descriptions of your practice files, and use **ZOOM-IN** to create them, just as you created the `status` file in your home directory.

EDITING FILES AND DIRECTORIES

The TEN/PLUS environment offers a range of editing tools based on the use of six TEN/PLUS functions:

PICK-UP	PUT-DOWN
PICK-COPY	PUT-COPY
INSERT	FORMAT

Using PICK-UP and PUT-DOWN

PICK-UP is used with **PUT-DOWN** to move text. To see how this is done, create the file `priorities` in the `practice` directory. Enter this text into the file:

```
l   t   t   t   t   t   t   t   t   t   r
Priority list of tasks to be accomplished before Account Group meeting.

Obtain cost figures for three inventory profiles
Review copy for ad campaign
Memo to George about comments for ad campaign
Write outline for presentation
**Reminder -- follow up on getting projection charts from
  Art Department_

/usr/larry/practice/priorities      INSERT   Line   11 (   11)
```

To move the line `Write outline for presentation` to the top of the list, position the cursor on that line and use **PICK-UP**. The entire line is picked up and disappears from the screen:

```

1   t   t   t   t   t   t   t   t   t   t   r

Priority list of tasks to be accomplished before Account Group meeting.

      Obtain cost figures for three inventory profiles
      Review copy for ad campaign
      Memo to George about comments for ad campaign
      *Reminder -- follow up on getting projection charts from
        Art Department

/usr/larry/practice/priorities      INSERT      Line      9 (      10)

```

Note that **PICK-UP** picks up the entire line, regardless of where the cursor is positioned, and moves the subsequent lines up. To move the line to the top of the list, position the cursor on the line Obtain cost figures for three inventory profiles, then use **PUT-DOWN**:

```

1   t   t   t   t   t   t   t   t   t   t   r

Priority list of tasks to be accomplished before Account Group meeting.

      Write outline for presentation
      Obtain cost figures for three inventory profiles
      Review copy for ad campaign
      Memo to George about comments for ad campaign
      **Reminder -- follow up on getting projection charts from
        Art Department

/usr/larry/practice/priorities      INSERT      Line      6 (      11)

```

You can also transfer information from one file to another by using **PICK-UP** and **PUT-DOWN** in conjunction with **ZOOM-IN** and **ZOOM-OUT**. First, **ZOOM-IN** to a file and **PICK-UP** the line you want. Then, using **ZOOM-OUT** and **ZOOM-IN**, as appropriate, access the directory containing the file into which the line is to be placed. **ZOOM-IN** to this file and use **PUT-DOWN** to insert the line where you want it.

You have just seen how to **PICK-UP** and **PUT-DOWN** one line of text at a time. You can use **PICK-UP** several times in succession, followed by **PUT-DOWN** the same number of times, to move several lines of text at one time. (Refer to section 4 of the “TEN/PLUS Tutorial” in this guide to learn how to pick up several lines of text in a single operation.) **PICK-UP** can also be used to delete text—simply **PICK-UP** that text and do not put it down.

Using PICK-COPY and PUT-COPY

PICK-COPY and **PUT-COPY** are similar in function to **PICK-UP** and **PUT-DOWN**. However, **PICK-COPY** leaves the original line where it was, and only picks up a copy. **PUT-COPY** allows you to put down several copies of the last line picked up by using either **PICK-UP** or **PICK-COPY**.

Create a file named `copiers` in your `practice` directory, then enter this text:

```

1      t      t      t      t      t      t      t      t      t      r

Evaluation chart for buying new copier

BRAND          COST          SPEED          QUALITY          SERVICE
Hybrid-II      $2,300          100 per min.   B+              A-

                Comments:  Has enlargement/reduction features, LED displays.
                Paper refill easy, toner refill somewhat clumsy._

/usr/larry/practice/copiers          INSERT      Line      10 (      10)

```

In this example, you typed in information about one brand of copier, Hybrid-II. To use the same headings (BRAND, COST, SPEED, QUALITY, SERVICE) to begin a new section for another brand, move the cursor to the heading line and use **PICK-COPY**. Then move the cursor to where you wish to start typing your evaluation of the second brand, and **PUT-DOWN** the copy:

```

1      t      t      t      t      t      t      t      t      t      r

Evaluation chart for buying new copier

BRAND          COST          SPEED          QUALITY          SERVICE
Hybrid-II      $2,300          100 per min.   B+              A-

                Comments:  Has enlargement/reduction features, LED displays.
                Paper refill easy, toner refill somewhat clumsy.

BRAND          COST          SPEED          QUALITY          SERVICE

/usr/larry/practice/copiers          INSERT   Line   14 (   14)

```

In this example, you could have used **PUT-COPY** instead of **PUT-DOWN** to continue putting copies of the heading at the beginning of each new section, without having to use **PICK-COPY** each time. **PUT-COPY** inserts a copy of the same line until you pick up another line by using either **PICK-UP** or **PICK-COPY**.

PICK-UP, **PUT-DOWN**, **PICK-COPY**, and **PUT-COPY** can also be used to move, copy, and delete entire files and directories. For example, you can delete a file or directory from a directory by placing the cursor in either field on the line where the file is listed and using **PICK-UP**.

☛ Deleting a directory deletes all files and directories in that directory!

You can transfer a file from one directory to another, perhaps from your home directory to another directory containing several related files. This procedure is similar to that for transferring data between files. Use **ZOOM-IN** and/or **ZOOM-OUT** to access the directory containing the file you wish to transfer. Position the cursor on the line where the file is listed, and **PICK-UP** the file; move to the destination directory and **PUT-DOWN** the file. You can also use this procedure to move directories.

PICK-COPY and **PUT-COPY** can be used to copy—rather than move—files and directories. For example, you might want to maintain a master version of a form letter that can be copied and completed as needed. Position the cursor on the line where the file is

listed and use **PICK-COPY**. Next, move to the destination directory and use **PUT-DOWN**. If you copy a file into its original directory, a popup box appears, asking you to give the copy a new name. This is because two files in the same directory cannot have the same name. Type a new file name (for example, form.1tr2) in the popup box and use **EXECUTE**.

Using FORMAT and INSERT

Two other basic TEN/PLUS functions are **FORMAT** and **INSERT**. They are used to format text between the current margins, and to insert space for new text.

Create the file memo in your practice directory, then enter the text as shown:

```
l   t   t   t   t   t   t   t   t   t   r
|-----|
| Ellen:                                     |
|                                           |
| I wanted to solicit your thoughts about  |
| the Sales Conference next week. I know  |
| that she's working on getting the       |
| display materials ready for the TransCorp|
| presentation next month, but if you    |
| could spare her, I think it would be a  |
| valuable experience for her internship  |
| and an asset to us if we hire her upon  |
| graduation. One of the problems with a  |
| lot of MBAs when they get out of school|
| is that they don't have a good feel for|
| the personal side of business. I think |
| it would be good for Lisa to get some  |
| of this exposure. What do you think?   |
| Will the TransCorp schedule permit?_   |
|-----|
|usr/larry/practice/memo                   INSERT   Line   14 (   13)
```

The body of this example can be split into two paragraphs to make it more readable. Move the cursor to the O in the word One on the fifth line of the paragraph, then use the space bar until the phrase One of the has been moved to the next line:

```

l   t   t   t   t   t   t   t   t   t   r
Ellen:

I wanted to solicit your thoughts about the possibility of Lisa attending
the Sales Conference next week. I know that she's working on getting the
display materials ready for the TransCorp presentation next month, but if
you could spare her, I think it would be a valuable experience for her
internship and an asset to us if we hire her upon graduation.
One of the
problems with a lot of MBAs when they get out of school is that they don't
have a good feel for the personal side of business. I think it would be
good for Lisa to get some of this exposure. What do you think? Will the
TransCorp schedule permit?

/usr/larry/practice/memo          INSERT   Line   10 (   14)

```

☛ If touching the space bar does not move the text to the right, read the section “Alternating Between Insert and Overwrite Modes” later on in this primer.

Use a blank line to separate paragraphs. With the cursor on the line One of the, use **INSERT**. A blank line is inserted:

```

l   t   t   t   t   t   t   t   t   t   r
Ellen:

I wanted to solicit your thoughts about the possibility of Lisa attending
the Sales Conference next week. I know that she's working on getting the
display materials ready for the TransCorp presentation next month, but if
you could spare her, I think it would be a valuable experience for her
internship and an asset to us if we hire her upon graduation.
-
One of the
problems with a lot of MBAs when they get out of school is that they don't
have a good feel for the personal side of business. I think it would be
good for Lisa to get some of this exposure. What do you think? Will the
TransCorp schedule permit?

/usr/larry/practice/memo          INSERT   Line   10 (   15)

```

You can use **INSERT** to insert as many blank lines as you wish. For example, you can use **INSERT** to insert space for new paragraphs, or to reserve space for diagrams on a printed copy of a file.

In the above example, the second paragraph needs to be reformatted because it begins with a short line. Position the cursor on the short line and use **FORMAT**. **FORMAT** reformats text to fit within the current margins:

```

1      t      t      t      t      t      t      t      t      t      r
Ellen:

I wanted to solicit your thoughts about the possibility of Lisa attending
the Sales Conference next week. I know that she's working on getting the
display materials ready for the TransCorp presentation next month, but if
you could spare her, I think it would be a valuable experience for her
internship and an asset to us if we hire her upon graduation.

One of the problems with a lot of MBAs when they get out of school is that
they don't have a good feel for the personal side of business. I think it
would be good for Lisa to get some of this exposure. What do you think?
Will the TransCorp schedule permit?

/usr/larry/practice/memo          INSERT   Line   11 (   14)

```

FORMAT reformats text from the current cursor position to the next blank line. Be sure to leave one or more blank lines between paragraphs as you type. Otherwise, **FORMAT** will run all of your paragraphs together!

USING MENUS

The TEN/PLUS environment includes two functions, **MENU** and **LOCAL-MENU**, that simplify the way you perform more complex tasks.

Using MENU

MENU offers a menu, called New Task Menu, that provides a number of general-purpose options. Your New Task Menu displays the same options regardless of which file or directory you are looking at. Here is a typical example of a New Task Menu:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
priorities	Priorities for the Account Group Meeting
copier	Copier evaluations
memo	Letter to Ellen about Lisa
<p style="text-align: center;">New Task Menu</p> <p>Move the cursor to an item and touch EXECUTE. Touch CANCEL to do nothing, HELP for help.</p> <p><u>S</u>how home directory Dⁱsplay the current date and time R^ead or send mail S^how your profiles directory E^dit your editor profile H^ousekeep Dⁱsplay history of current file</p>	
/usr/larry/practice	
INSERT Line 3 (3)	

The options on your New Task Menu may differ somewhat from those displayed above. You can easily add or remove options from your New Task Menu. (Refer to section 2 of "TEN/PLUS Profiles" in this guide for additional information about changing your New Task Menu.) The New Task Menu displayed above contains these options:

Show home directory

Returns you to your home directory, regardless of where you are in the directory structure.

Display the current date and time

Displays the current date and time in a popup box. (Use **CANCEL** to remove the box.)

Read or send mail

Displays your mailbox if the TEN/PLUS Mail System is available on your computer.

Show your profiles directory

Displays your profiles directory so you can view or edit your TEN/PLUS profiles.

Edit your editor profile

Displays your editor profile, which is used to customize your editing environment. (Refer to “TEN/PLUS Profiles” in this guide for information about changing your editor profile.)

Housekeep

Removes all versions of files except the current version.

■ You must use Housekeep periodically (say, once a day) to prevent various files from growing too large and wasting storage space.

Display history of current file

Displays a list of all versions of the current file.

To select an option from this or any other menu, position the cursor on the desired option and use **EXECUTE**, or use any of the functions **(1)** through **(8)**, depending on the number that corresponds to the line on which your choice is listed. For example, to select **Show your profiles directory** from the default **New Task Menu**, use **(4)**, since this is the fourth option on the menu. Use **CANCEL** to remove a menu from the display.

Using LOCAL-MENU

LOCAL-MENU is similar to **MENU**, except that it displays a menu of options that apply specifically to the type of information or to the application you are using at the moment. (Refer to the “TEN/PLUS Tutorial” in this guide or to the appropriate

application guide for more information about specific local menus.) Here is a sample local menu for an optional TEN/PLUS application, the TEN/PLUS Mail System:

```

Subject: Forwarded: First thoughts on the QuikSell account
To:      larry
Cc:      brian

Bcc:
Date: 18 Apr 1984 0957-PDT
From:    Janet Brown

I have extracted the juiciest parts here ... let me know what you think.

      Electronic Mail

Move the cursor to an item and touch EXECUTE.
Touch CANCEL to do nothing, HELP for help.

  (1) Mail this message
  (2) Reply to this message
  (3) Forward this message
  (4) Delete this message
  (5) Restore deleted message
  (6) File this message in another mailbox
  (7) Show in-box and add new mail

      they want to accomplish with
      soon with a couple of
      t down here my best
      notes of my own on things we
      ls. Also, I have included
      orrespondence from the
      a better idea of what has
      u have looked at all of this,
      questions you may have about
      roach that will have a very
      if our proposals are very

Size: 150 Lines   Sent by: janet at RALEIGH   Status:
/usr/larry/mhs/1   INSERT   Line   6 ( 150)

```

If you use **LOCAL-MENU** when looking at a directory, this menu will appear:

```
                      INTERACTIVE TEN/PLUS FILE MANAGER
                Move the cursor to an item below and ZOOM-IN to see it.

          File                        Description
-----
priorities        Priorities for the Account Group Meeting
copiers           Copier evaluations
memo              Letter to Ellen about Lisa
*
Move the cursor to desired action and touch EXECUTE.
To do nothing, touch CANCEL.
For help, touch HELP.

- (1) Display "visible" files
- (2) Display all files
--- Return to normal directory display
(4) Show details about files
(5) Show more details about this file
--- Show more details about this file

/usr/larry/practice                INSERT      Line      3 (    3)
```

This local menu is seldom used in simple applications. It is explained in detail in section 5 of the “TEN/PLUS Tutorial” in this guide.

MORE ABOUT EDITING

In addition to the basic TEN/PLUS functions, several other editing functions are also quite useful. They are used to change margins and tab stops, to provide alternate methods for modifying or editing existing text, to find a specific word or phrase in a file, and to move text between files. The sections below explain how to use these functions, as well as how to print your documents.

Changing Margins and Tabs

Up to now, you have used the *default* margins and tabs that are set automatically. You can change these margins and tabs at any time.

To change the left margin, position the cursor where you wish the new left margin to be and use **MARGIN**. The l on the ruler on the top line moves to the new left margin. To type an indented paragraph, for example, move the left margin in, type the paragraph, and then move the left margin back to its original position.

To change the right margin, position the cursor where you wish the right margin to be and use **ENTER**, then **MARGIN**. The r on the ruler on the top line moves to the new right margin.

You may have occasion to type a line or lines of text that extend beyond the right-hand border of the screen. To do this, you will need to change the right-hand margin to a column greater than 77 (the default column position for the right margin). You can use **RIGHT** to bring into view a portion of the file (approximately one-third of the width of the screen) that extends beyond the right-hand border. Using **RIGHT** again will bring another such portion into view. Bring as much of the file into view as necessary to reset the right-hand margin. The maximum setting of the right-hand margin is column 200.

Similarly, you can use **LEFT** to bring into view a portion of the file (if any) that extends beyond the left-hand border.

You can use **BEGIN-LINE** to move the cursor to the leftmost character of the current line, and **END-LINE** to move the cursor one position to the right of the rightmost character of the current line.

To set a new tab stop, position the cursor at the desired column and use **SET-TAB**. To remove a tab stop, use **TAB** to position the cursor on that stop, then use **ENTER** followed by **SET-TAB**.

Alternating Between Insert and Overwrite Modes

Until now, you typed text while in insert mode. When insert mode is in effect, new text is inserted at the cursor position, and existing text is moved to the right or word-wrapped to the next line. The word **INSERT** on the bottom line of the screen indicates that insert mode is in effect.

On the other hand, new text replaces existing text as you type when overwrite mode is in effect. The word **OVERWRITE** replaces the word **INSERT** on the bottom line of the screen.

You can switch between insert and overwrite modes with **INSERT-MODE**. If you are in insert mode, **INSERT-MODE** places you in overwrite mode; conversely, if you are in overwrite mode, **INSERT-MODE** places you in insert mode.

■ **INSERT-MODE** is independent of **INSERT**. **INSERT-MODE** is used to switch between insert and overwrite modes, while **INSERT** is used to insert blank lines in a file or directory.

Using +SEARCH, -SEARCH, and BREAK

On occasion, you may want to quickly locate a word or phrase in a file. To search for a word or phrase, starting at the current cursor position and continuing to the end of the file, use **ENTER**, type the word or phrase you wish to find, then use **+SEARCH**. The system searches for an exact character match. For example, a search for the word **sales** will find **sales**, but not **Sales**, because the small **s** and the capital **S** are different characters. If there is no matching word or phrase, a message will appear in a popup box. If the search is successful, the cursor will be positioned on the matching word or phrase. You can then search for the next occurrence of the same word or phrase by simply using **+SEARCH** again.

You can also search backward through the file, starting at the current cursor position and continuing to the beginning of the file. To search backward use **-SEARCH** instead of **+SEARCH**.

You can interrupt a search operation by using **BREAK**. For example, if you discover after using either **+SEARCH** or **-SEARCH** that you made an error while typing the word or phrase to be found, you can use **BREAK** to stop the search operation.

Using USE

As you have seen, you can move through the directory structure to a specific file by using **ZOOM-OUT** and **ZOOM-IN**, as appropriate. Because you are able to see the directory and file names at the various directory levels, you do not need to remember the full name of a file in order to access it.

You can, however, use another method to access a specific file directly, if you know its exact name. Use **ENTER**, type the name of the desired file in the popup box, and use **USE**. **USE** brings the desired file onto the screen. At this point, you have established what is known as an alternate file. Now every time you use **USE**, the system will alternately display the original file and the alternate file.

To directly access another file, or to establish a new alternate file, simply use **ENTER**, type the name of the new alternate file, and use **USE**. Once you have established an alternate file, you may move text between the original file and the alternate file by using **PICK-UP** or **PICK-COPY** in one file, then **USE**, and then **PUT-DOWN** or **PUT-COPY** in the other.

Printing Documents

To print a document, **ZOOM-IN** until its contents are visible, then use **PRINT**. A menu appears, listing the print options available on your system.

To print a document on your default printer, position the cursor at the option **Print on default printer**, and use **EXECUTE**. (Refer to section 3 of “TEN/PLUS Profiles” in this guide for additional information about the **Print Menu**.)

FOR MORE INFORMATION

This concludes your introduction to the TEN/PLUS environment. The TEN/PLUS environment consists of the TEN/PLUS User Interface, described briefly in this primer, as well as optional development tools and applications, such as the TEN/PLUS Mail System. After you have learned the basic TEN/PLUS functions, you may want to know more about advanced TEN/PLUS functions and capabilities, such as the ability to display and work with several files (“windows”) at the same time. The “TEN/PLUS Tutorial” in this guide reviews the ten basic functions and describes in detail various advanced functions available in the TEN/PLUS User Interface. Refer to the “TEN/PLUS Reference Manual” in this guide for more information about specific TEN/PLUS User Interface capabilities.

SUMMARY OF TEN/PLUS FUNCTIONS

This section summarizes the ten basic functions, as well as a number of additional functions, that are available with the TEN/PLUS system. Many of these functions can be modified or enhanced when used with other functions. For detailed descriptions of these functions, and of other functions available with the TEN/PLUS system, refer to the “TEN/PLUS Tutorial” and to the “TEN/PLUS Reference Manual” in this guide.

The Ten Basic Functions of the TEN/PLUS System

ZOOM-IN	Displays a more detailed level of information.
ZOOM-OUT	Displays a less detailed level of information.
PICK-UP	Removes the current line and saves it until it is PUT-DOWN .
PUT-DOWN	Inserts, at the current cursor position, the last line picked up.
PICK-COPY	Picks up a copy of the current line and moves the cursor down one line.
PUT-COPY	Inserts, at the current cursor position, a copy of the last line picked up.
INSERT	Inserts a blank line at the current cursor position.
FORMAT	Formats the current paragraph within the current margins. The previous, unformatted version of the paragraph is available with RESTORE .
MENU	Displays the New Task Menu.
LOCAL-MENU	Displays a menu of options specific to the current application or to the type of information you are using.

Some Additional TEN/PLUS Functions

↑ ↓ → ←

Move the cursor up, down, right, or left.

BACKSPACE

Corrects typing errors by erasing characters.

BEGIN-LINE

Moves the cursor to the leftmost character of the current line.

BOX-MARK

Marks lines or rectangular blocks of text for copying, inserting, or deleting. Can be used with cursor-positioning and other TEN/PLUS functions, such as **PICK-UP**, **PICK-COPY**, **INSERT**, and **DELETE**.

BREAK

Stops **+SEARCH** and **-SEARCH**.

CANCEL

Removes an error message or popup box from the display.

CENTER

Centers the current line between the current margins.

DELETE

Deletes the current line. The deleted line can be restored with **RESTORE**.

DELETE-CHARACTER

Deletes the character at the current cursor position.

END-LINE

Moves the cursor one space to the right of the rightmost character of the current line.

ENTER

Enhances or modifies many TEN/PLUS functions, such as **PICK-UP**, **PICK-COPY**, **INSERT**, and **DELETE**. For example, **ENTER** 5 **PICK-UP** picks up five successive lines of text. The “TEN/PLUS Tutorial” in this guide describes all the capabilities of **ENTER**.

EXECUTE

Invokes options that appear in menus or “help” popup boxes. Cursor-positioning functions are used to place the cursor at the option to be **EXECUTE**d.

EXIT

Exits from the TEN/PLUS environment, saving all changes made to files during this editing session, or since the last **SAVE**.

FUNCTIONS

Displays a menu showing which of the ten basic TEN/PLUS functions are active and allows you to select one.

GO-TO

Displays the beginning of the file and positions the cursor on the first line. If the cursor is already on the first line of the file, **GO-TO** displays the end of the file and positions the cursor on the last line of the file. **ENTER GO-TO** always displays the end of the file and positions the cursor on the last line of the file. **ENTER** line-number **GO-TO** moves the cursor to that line of the file. For example, **ENTER** 47 **GO-TO** displays the portion of the file that contains line 47 and positions the cursor on that line.

HELP

Displays information relevant to the current situation.

HOME

Moves the cursor to the upper left-hand corner of the display.

INSERT-MODE

Alternates between insert and overwrite modes.

LEFT

Brings into view a portion (approximately one-third of the width of the screen) of the file that extends beyond the left-hand border.

+LINE

Scrolls the text forward (up) approximately one-third of the screen at a time. **ENTER** n **+LINE** scrolls the text forward n lines. For example, **ENTER** 3 **+LINE** scrolls the text forward three lines.

-LINE

Scrolls the text backward (down) approximately one-third of the screen at a time. **ENTER** n **-LINE** scrolls the text backward n lines. For example, **ENTER** 4 **-LINE** scrolls the text backward four lines.

MARGIN

Sets the left-hand margin at the current cursor position. **ENTER** **MARGIN** sets the right-hand margin.

+PAGE

Scrolls the text forward (up) a full screen at a time. **ENTER** n **+PAGE** scrolls the text forward n full screens. For example, **ENTER** 2 **+PAGE** scrolls the text forward two full screens.

-PAGE

Scrolls the text backward (down) a full screen at a time. **ENTER** n **-PAGE** scrolls the text backward n full screens. For example, **ENTER** 3 **-PAGE** scrolls the text backward three full screens.

PRINT

Displays a menu of print options.

REFRESH

Redraws the screen display.

RESTORE

Restores, at the current cursor position, the most recently **DELETE**d text.

ENTER

Moves the cursor to the beginning of the next line.

RIGHT

Brings into view a portion (approximately one-third of the width of the screen) of the

file that extends beyond the right-hand border.

SAVE

Saves all changes made to files during this editing session, or since the last **SAVE**.

+SEARCH

Searches forward through the file for a specified word or phrase. For example, **ENTER** Sam **+SEARCH** searches for the next occurrence of the word Sam; subsequent occurrences of Sam can be found by using **+SEARCH** alone.

-SEARCH

Searches backward through the file for a specified word or phrase. Like **+SEARCH**, **-SEARCH** can be used alone to find subsequent occurrences of the word or phrase.

SET-TAB

Sets a tab stop at the current cursor position. **ENTER** **SET-TAB** removes the tab stop at the current cursor position.

TAB

Moves the cursor to the next tab position on the right.

-TAB

Moves the cursor to the previous tab position on the left.

TEXT-MARK

Marks text for copying, inserting, or deleting. Can be used with cursor-positioning and other TEN/PLUS functions, such as **PICK-UP**, **PICK-COPY**, **INSERT**, and **DELETE**, to manipulate multiline sentences and nonrectangular text regions.

USE

Used with **ENTER** to directly access another file. Also used to switch back and forth between two files.

GLOSSARY

ASCII file

An ordinary text file. ASCII stands for “American Standard Code for Information Interchange,” and refers to the way characters are represented in a computer.

cursor

A cursor is a pointer on your screen to where the next action will take place. When typing, the cursor indicates the position where the next character will appear. When using a TEN/PLUS function, the cursor indicates the line or paragraph on which that function will operate.

cursor-positioning functions

The functions that permit you to move the cursor from one position to another. Examples are **TAB**, **-TAB**, **↑**, **↓**, **←**, **→**, and **ENTER**.

default

The option that will be chosen for you by the system if you do not make a choice. For example, margins and tabs are set automatically by the system, so that you do not have to set them every time you access a file. You can, however, change many of these settings at will. Also, if your computer system has several printers, and you print a document, it will be printed on the default printer unless you select another one.

directory

An TEN/PLUS directory contains documents or files, and/or other directories. A typical directory contains related documents, such as memoranda or monthly sales reports.

field

A location in a file or form that is reserved for a single piece of information. For example, a personnel file might include separate fields for an employee’s name, address, Social Security number, and so on. Each TEN/PLUS directory screen has two fields, **File** and **Description**.

file

A document or a collection of information stored on the computer. For example, a file could be a phone list, a memorandum, or a report.

full file name

The full file name of the file completely describes the location of the file in the system. It is also called a “full path name.” It is made up of directory names and file names, separated by slashes. The full file name is displayed at the bottom of the screen and when you attempt to create a new file.

function

A way of performing an often complex operation with one or more keystrokes, which usually saves you from having to perform a series of steps.

home directory

The directory that contains the “master list” or index of the information you are storing on the computer.

keyboard template

An illustration of a keyboard, that indicates the locations of the keys used to invoke functions.

log in

Activate your computer or terminal to gain access to your information.

popup box

A small box on your screen that displays instructions or error messages.

scroll

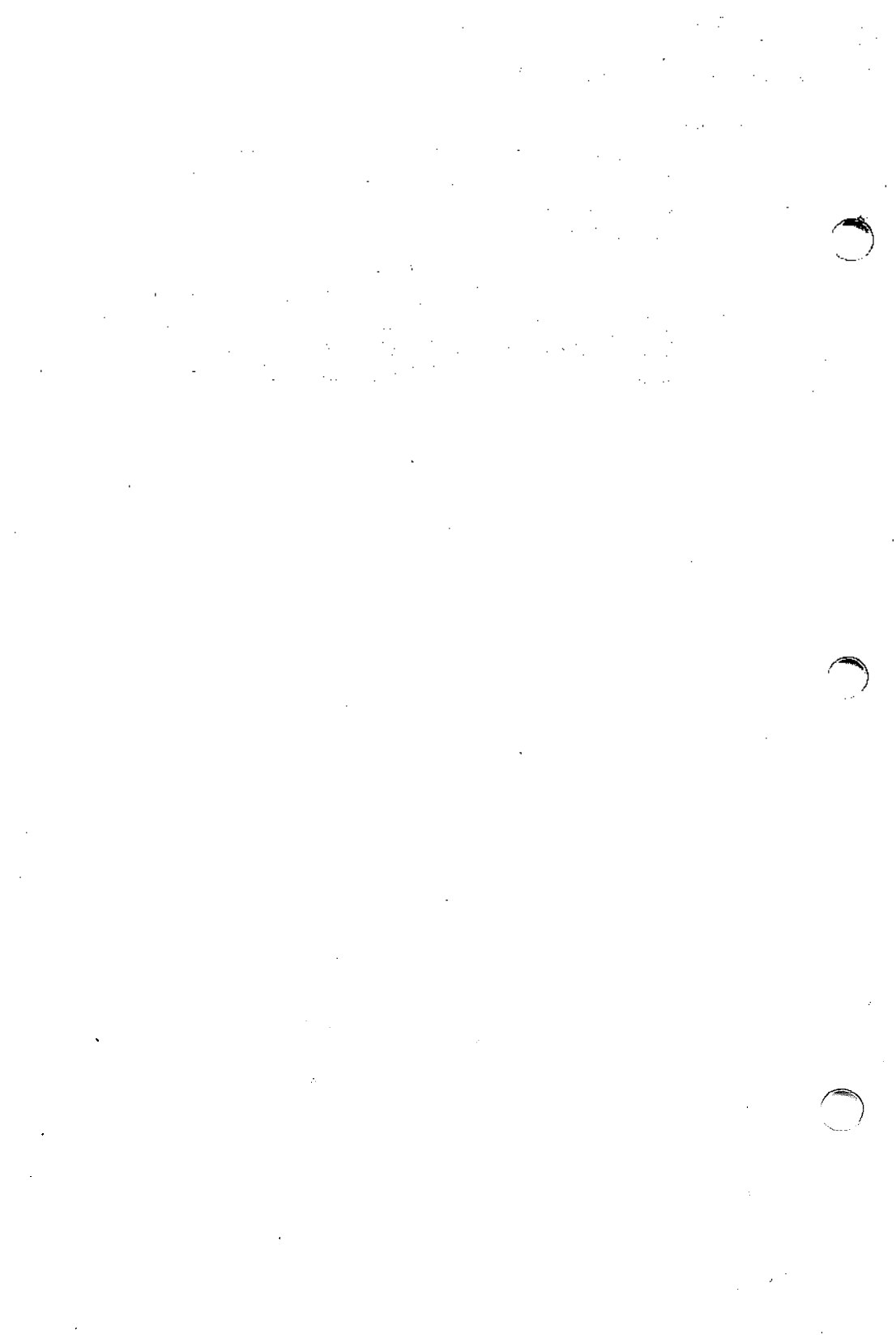
Move text off of the display so that you can see the preceding or following text. When you reach the bottom of the screen while typing, TEN/PLUS automatically scrolls the text forward, so that you always have a place for the next line. When you are reading a document, you can scroll text forward (up) one screen at a time by using **+PAGE**, or one-third of a screen at a time by using **+LINE**. Similarly, you can use **-PAGE** and **-LINE** to scroll text backward (down).

structured file

A special type of file available in the TEN/PLUS environment. TEN/PLUS keeps information about structured files that allows you to look at previous versions of those files.

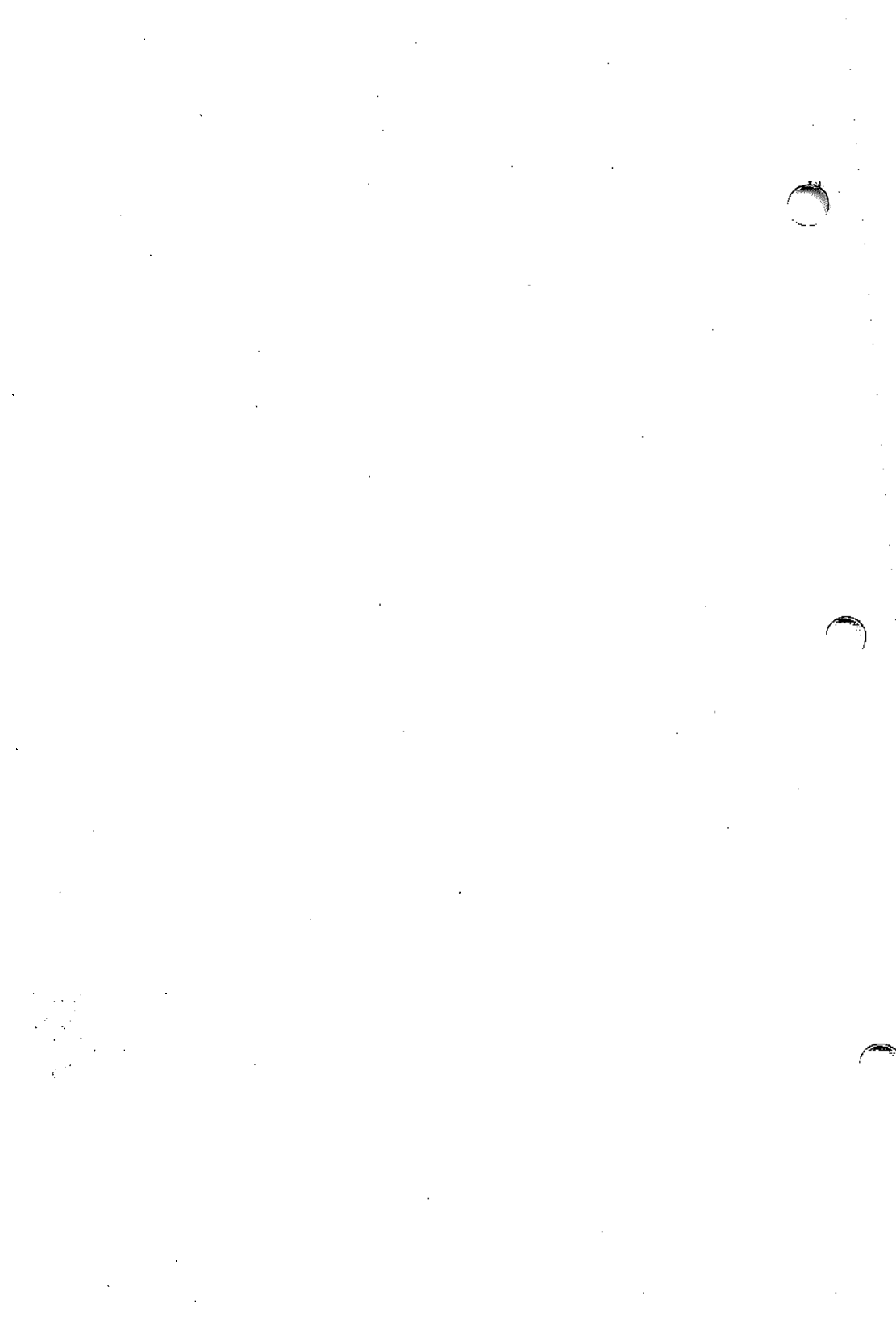
word wrap

A feature that eliminates the need for you to use **ENTER** at the end of each line. When a word you are typing extends past the right-hand margin, the entire word is moved to the left-hand margin of the next line.



INDEX

- /, slash character 4
- alternate file, establishing 24
- arrow keys, cursor-positioning 4
- ASCII file 6
- BACKSPACE function 4
- BEGIN-LINE function 22
- BREAK function 23
- CANCEL function 8
- changing margins 22
- changing tabs 22
- copying directories 14
- copying files 14
- copying text 13
- creating a document 5
- creating directories 9
- cursor 3
- cursor, moving to beginning of line 22
- cursor, moving to end of line 22
- cursor-positioning functions 4
 - deleting directories 14
 - deleting files 14
 - description field 3
 - directory, copying 14
 - directory, creating 9
 - directory, deleting 14
 - directory, editing 11
 - directory, home 2
 - directory, moving 14
 - editing directories 11
 - editing files 11
 - END-LINE function 22
 - entering TEN/PLUS Environment 2
 - file, alternate 24
 - file, ASCII 6
 - file, copying 14
 - file creation 5
 - file, deleting 14
 - file, editing 11
 - file field 3
 - file, moving 14
 - file name characters 4
 - file name, full 4
 - file name size 3
 - file, structured 6
 - FORMAT function 15
 - formatting text 15
 - full file name 4
 - functions, cursor-positioning 4
 - getting help 8
 - HELP function 8
 - home directory 2
 - INSERT function 15
 - INSERT-MODE function 23
 - inserting space for new text 15
 - inserting text 23
 - interrupting a search 23
 - keyboard template 2
 - LEFT function 22
 - LOCAL-MENU function 19
 - logging in 2
 - margins, changing 22
 - MENU function 18
 - menus, removing 8
 - menus, using 18
 - LINE function 7
 - PAGE function 7
 - SEARCH function 23
 - TAB function 4
 - moving directories 14
 - moving files 14
 - moving screen to left 22
 - moving screen to right 22
 - moving text 11
 - New Task Menu 18
 - OVERWRITE function 23
 - overwriting text 23
 - password 2
 - PICK-COPY function 13
 - PICK-UP function 11
 - +LINE function 7
 - +PAGE function 7
 - +SEARCH function 23
 - popup boxes, removing 8
 - PRINT function 24
 - printing documents 24
 - PUT-COPY function 13
 - PUT-DOWN function 11
 - RIGHT function 22
 - scrolling 7
 - searching text 23
 - SET-TAB function 22
 - slash (/) 4
 - starting TEN/PLUS 2
 - structured file 6
 - TAB function 4
 - tabs, changing 22
 - tabs, setting 22
 - TEN/PLUS functions, summary 26
 - text, formatting 15
 - text, inserting 23
 - text, inserting space for 15
 - text, overwriting 23
 - text, searching 23
 - text, searching backwards 23
 - USE function 24
 - using menus 18
 - word wrap 3
 - ZOOM-IN 4
 - ZOOM-OUT 4



TEN/PLUS Tutorial

CONTENTS

1. INTRODUCTION	1
1.1 Scope of This Tutorial	1
1.2 Overview of This Tutorial	1
1.3 Accessing TEN/PLUS Functions	3
2. GETTING STARTED	4
2.1 Logging In	4
2.2 Using MENU to Return to Your Home Directory	5
2.3 Creating a File	6
2.4 Using the Keyboard	8
2.5 Using the Cursor- and Window-Positioning Functions	10
2.6 Using Menus	11
2.7 Using FUNCTIONS	11
2.8 Using the TEN/PLUS ENTER Function	12
2.9 Using the TEN/PLUS HELP and CANCEL Functions	13
2.10 Exiting a File	13
2.11 Accessing Existing Files	13
2.12 Logging Out	14
3. TYPING AND FORMATTING TECHNIQUES	15
3.1 Changing the Left-Hand Margin	15
3.2 Changing the Right-Hand Margin for Word Wrap	17
3.3 Changing Both Margins Simultaneously	19
3.4 Readjusting Lines on New Margins	20
3.5 Centering Text	21
3.6 Changing Tab Stops	23
3.7 Underscoring With FONT	23
4. BASIC EDITING TECHNIQUES	25
4.1 Deleting Characters or Words Within a Line	26
4.2 Adding Characters or Words Within a Line	27
4.3 Correcting Transpositions and Typographical Errors	29
4.4 Moving Windows	30
4.5 Deleting or Moving Lines	31
4.5.1 Using DELETE	32

4.5.2	Using PICK-UP	32
4.6	Adding Blank Lines and Splitting Lines	33
4.7	Joining Lines	35
4.8	Defining Rectangular Areas Using BOX-MARK	35
4.9	Adding Lines Using BOX-MARK	37
4.10	Deleting and Duplicating Lines Using BOX-MARK	39
4.11	Defining Nonrectangular Areas Using TEXT-MARK	40
4.12	Deleting or Moving Sentences Using TEXT-MARK	41
4.13	Moving Words, Lines, and Paragraphs	43
4.14	Duplicating Words, Lines, and Paragraphs	44
4.15	Inserting, Moving, Deleting, and Duplicating a Specific Number of Lines	46
4.16	Opening, Moving, Deleting, and Duplicating Columns	47
4.17	Moving to a Specific Line in a File	49
4.18	Searching for Specific Text	50
4.19	Performing Individual Search and Replace	51
4.20	Accessing Two or More Files	52
4.20.1	Viewing an Alternate File	52
4.20.2	Creating Multiple Windows	53
5.	THE FILE MANAGER	56
5.1	Copying Files	56
5.2	Renaming Files	59
5.3	Deleting Files	59
5.4	Path Names and Directory Structure	60
5.5	Creating Directories	61
5.6	Moving Files Between Directories	62
5.7	Copying Files Between Directories	63
5.8	Alternate Methods of Changing Directories	64
5.8.1	Full and Relative Path Names	64
5.8.2	Parent Directories	64
5.9	Removing Directories	65
5.10	Recovering Directories From <code>.putdir</code>	65
5.11	File and Directory Permissions	68
5.11.1	Changing Permissions on Files and Directories	69
5.12	New Task Menu Options	75
5.12.1	Executing a Command From the Editor Subshell	76
5.12.2	Executing a Command in a Pop-up Box	77
5.12.3	Showing Your Profiles Directory	78
5.12.4	Editing Your Editor Profile	78
5.12.5	Housekeep	79

5.12.6	Displaying the History of the Current File	79
5.13	The File Manager Profile	80
6.	THE HISTORY DISPLAY	82
6.1	Types of INed Files	82
6.2	Creating a Structured File	82
6.2.1	Creating Versions of a File	84
6.3	Accessing the History of a File	85
6.3.1	The History Display Local Menu	87
6.4	Removing History	89
6.5	Structured Nontext Files	90
7.	TEN/PLUS C PROGRAMMING HELPER	93
7.1	Using The C Programming Helper	93
8.	ADDITIONAL FUNCTIONS	96
8.1	More About HELP and the Help Menu	96
8.2	Communicating With the System While in the TEN/PLUS Environment	97
8.2.1	Running System Commands	97
8.3	Exiting the TEN/PLUS Environment	98
8.4	Sorting Columns	99
8.5	Performing Global Search-and-Replace	100
8.6	Using the Spell Program	101
8.7	Printing a Document	103
8.8	Splitting a File	104
8.9	Searching for a Subject Within a Directory	105
8.10	Changing Your Password	105
8.11	Recovering From System Crashes	106
8.12	Removing Line-Noise Characters and Broadcast Messages	107
8.13	Using Wildcards	107
Appendix:	TEN/PLUS FUNCTIONS	108
1.	CONTROL AND MENU FUNCTIONS	108
2.	WINDOW-POSITIONING FUNCTIONS AND SEQUENCES	109
3.	CURSOR-POSITIONING FUNCTIONS	110
4.	TAB SETTING/RELEASING FUNCTIONS AND SEQUENCES	111

5. TEXT MANIPULATION FUNCTIONS AND SEQUENCES 111

6. MULTIPLE FILE/WINDOW FUNCTIONS AND SEQUENCES 115

7. EXIT/SAVE FUNCTIONS AND SEQUENCES 116

TEN/PLUS* Tutorial

1. INTRODUCTION

1.1 Scope of This Tutorial

This tutorial is a self-paced training guide for beginners who wish to learn how to use the TEN/PLUS system to create, edit, and manage text files. It is designed for users having no system management responsibilities. Use of this tutorial requires that the TEN/PLUS User Interface has been installed and is available for use.

This tutorial introduces the basic components of the TEN/PLUS User Interface and describes how it provides a bridge to the UNIX* Operating System. It explains procedures for entering, exiting, and reentering the TEN/PLUS system that facilitate moving back and forth between the TEN/PLUS system and the UNIX System shell. It details only the fundamental UNIX System commands, such as those used for logging in and logging out. Refer to your user's manual for detailed information about UNIX System commands.

This tutorial is designed as an introduction to the TEN/PLUS User Interface and does not provide a complete reference to all of its extensive facilities. Refer to the "TEN/PLUS Reference Manual" for more detailed information about available commands. Refer to other sections of this tutorial for information about editing and formatting documents.

1.2 Overview of This Tutorial

This tutorial describes the three main features of the TEN/PLUS User Interface: the INed* editor, the File Manager, and the History Display. The sections included are:

1. INTRODUCTION

This section provides a general overview of the tutorial.

2. GETTING STARTED

This section describes how to start up and log in to the TEN/PLUS system. It introduces the INed editor and teaches the user to:

- create a file
- use the keyboard

scroll within a file
modify functions
access the HELP facilities
ZOOM-IN to a file
ZOOM-OUT of a file
access existing files

This section ends with information about exiting the TEN/PLUS system and logging out.

3. TYPING AND FORMATTING TECHNIQUES

This section explains techniques for formatting text. It describes how to change margins and tabs, center and align text, and access alternate fonts.

4. BASIC EDITING TECHNIQUES

This section covers techniques for inserting, deleting, moving, copying, and searching for text. It also explains how to delete, copy, and move blocks of text, columns, and sentences.

This section ends with a discussion of two procedures (an alternate file approach and a multiple window approach) used to access several files during the same editor session. These procedures facilitate copying or moving text between different files, or between different portions of the same file.

5. USING THE FILE MANAGER

This section describes how to use the File Manager to manipulate files and directories. It explains procedures for copying, deleting, combining, moving, and renaming files and directories. It also describes how to change permissions on files and directories either to limit or give access to others for reading and editing. An explanation of how directories are structured is provided along with a description of how to create and remove directories.

6. STRUCTURED FILES AND THE HISTORY DISPLAY

This section describes TEN/PLUS structured files and explains briefly why many of the TEN/PLUS functions and features can be used only on structured files. It

explains how to convert files from structured to ASCII and ASCII to structured.

This section also introduces the History Display utility. It explains how to use the History Display to recall previous versions of structured files.

7. THE C PROGRAMMING HELPER

This section describes how the TEN/PLUS C Programming Helper can be used. It is intended for C programmers who want to develop, compile, and test programs without leaving the TEN/PLUS environment. The C Programming Helper is designed for use with the KNR Compiler only.

8. ADDITIONAL FUNCTIONS

This section contains both system and editor information. The emphasis, however, is on using UNIX System filters from within the INed editor. UNIX System filters, used to modify text while using the editor, facilitate such functions as global replacing, sorting, and justifying.

This section also describes editor facilities, such as the help and print menus. It concludes with procedures for changing passwords, changing the editor profile file, and using wildcards.

APPENDIX: TEN/PLUS FUNCTIONS

This appendix describes the capabilities of each of the TEN/PLUS functions and sequences.

1.3 Accessing TEN/PLUS Functions

Accessing TEN/PLUS functions requires different keystroke sequences on different keyboards. Refer to the appropriate section of “TEN/PLUS Keyboard Information” for an alphabetic listing of the TEN/PLUS functions and the keystroke sequences required for your keyboard.

Note that references to the **ENTER** key indicate the key labelled RETURN or ENTER on your keyboard. References to the TEN/PLUS ENTER function (ENTER) indicate the key labelled ENTER on your TEN/PLUS template.

2. GETTING STARTED

2.1 Logging In

You should be familiar with the terms *prompt*, *file*, and *directory* before logging in. A prompt is a signal that prints on the display; it indicates that the system is ready to receive information. A file is a collection of data records stored on the computer. For example, a file might contain a phone list, a letter, or a report. A directory is a collection of files. For example, all files related to a given topic might be located in the same directory. Directories are explained in detail in §5. These terms are used frequently throughout this tutorial. Additional examples are provided later in the text.

The system prompts for user identification before granting access. Although the prompt can be personalized from system to system, it usually asks for a user name and a password. To log in:

1. Type your user name and press the **ENTER** key. (The information is “sent” to the computer when the **ENTER** key is used.)
2. The system may prompt for a password (private security identification) after a user name is received. If a password has been issued and the password prompt displays, type the password and press the **ENTER** key. For security purposes, the password does *not* appear on the display when typed. If a password has not been issued and the password prompt displays, press the **ENTER** key to complete the login procedure.

If your system administrator has arranged for you to automatically log in to the TEN/PLUS environment, a formatted screen, listing the name and description of each of your files and directories, will appear. This listing is called your *home directory*. This screen is an example of a home directory:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
/usr/larry	
INSERT Line 1 (4)	

(Note that screens may vary on different displays.) If your screen does not look similar to this one, your cursor should be located to the right of a system prompt, typically a \$. Type e . and then press the **ENTER** key to enter the TEN/PLUS environment.

2.2 Using MENU to Return to Your Home Directory

If at any time you get lost in the TEN/PLUS environment, you can use **MENU** to return to your home directory. **MENU** provides an easy way to perform actions that would otherwise require more effort. When you use **MENU**, a menu of options appears. Position the cursor on the desired option, then use **EXECUTE** to perform the indicated operation, or **CANCEL** to remove the menu. (The list of menu options is installation-dependent and can be overridden by a user in his editor profile file. Refer to “TEN/PLUS Profiles” for more information about customizing your editor profile file.) To use **MENU** to return to your home directory:

1. Use **MENU**:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                Description
-----
1988.cal            Calendar

```

New Task Menu

Select a menu option (for example, move the cursor to an item and EXECUTE); otherwise, CANCEL to remove the menu, or HELP to display help information.

- Show home directory
- Execute UNIX shell commands
- Run a shell command in a box
- Show your profiles directory
- Edit your editor profile
- Housekeep
- Display history of current file

```

/usr/larry                INSERT Line      1 (      4)

```

2. With the cursor on Show home directory, use **EXECUTE**; you will return to the File Manager display in your home directory. You can move quickly to your home directory from any other location by selecting this option from the New Task Menu. See §5 for a discussion of the other options on the New Task Menu.

2.3 Creating a File

To create and access a new file using the TEN/PLUS system:

1. With the cursor at the beginning of the first blank line in the File field, type the file name. If you make a typographical error, use **BACKSPACE** to correct the error, then continue typing. Use only alphanumeric characters and the symbols . and , and _ when naming a file. File names should not contain spaces. Use the name report for this example.
2. **TAB** or use **→** to move the cursor to the first position in the Description field.
3. Type a brief description in the Description field. For example, type First Report on the QuikSell Project.
4. **ZOOM-IN**. Since the report file does not yet exist, a menu will appear:

```

                INTERACTIVE  TEN/PLUS FILE MANAGER
                Move the cursor to an item below and ZOOM-IN to see it.

                File                Description
    1988.cal      Calendar
    msg          My Incoming Messages
    phone        Company Telephone Book
    policies     Company Policies
    report       First Report on the QuikSell Project

    You are attempting to create file "/usr/larry/report"

    Select a menu option (for example, move the
    cursor to an item and EXECUTE); otherwise,
    CANCEL to remove the menu, or HELP to display
    help information.

    Create an ASCII file (without history)
    Create a structured file (with history)
    Create a directory
    Re-enter the file name

    /usr/larry                INSERT      Line      5 (      5)

```

Your options are to create an ASCII file, a structured file,¹ or a directory, or to re-enter the file name in case you mistyped it. For these exercises, you will create ASCII files.

5. With the cursor on **Create an ASCII file (without history)**, use **EXECUTE** to create the new file. A blank window will appear on the display.

The window is a view into the file and often displays only a portion of the text in the file. The window indicates that you are in the INed text editor and may now enter or revise data in the file. Since this file is empty, the window is blank.

On most displays, the window is 20 or 21 lines high by 78 characters wide (approximately one-third of an 11-inch page). The cursor indicates the current position in the window. On most displays, it is represented by a blinking or solid box, or a blinking or solid underline. Keyboard input is implemented or displayed at the cursor

-
1. ASCII files store text and programs. The computer treats the material in an ASCII file like a stream of characters. Structured files store specialized data, such as the help menus you will use later.

position. When a file is created, the cursor is positioned on the first line of the new file.

Certain file-specific information, such as the file name, appears below the window. System instructions and comments appear whenever necessary in popup boxes in the window. Use **CANCEL** to remove popup boxes.

HELP provides general editing information and specific information pertaining to a popup box when one is displayed. **HELP** is available whenever you are in the TEN/PLUS environment. Refer to this function occasionally while reviewing this tutorial to become familiar with its use.

2.4 Using the Keyboard

Some keyboards differ slightly from regular typewriters. Some symbols may be located in different positions and others may be unique to keyboards. On some keyboards, the numeric keypad can be used for rapid entry of numbers when NUM-LOCK is in effect.

Follow the steps outlined below to acquaint yourself with the keyboard. The words automatically wrap at the right-hand margin indicated at the top of the window; it is not necessary to press **ENTER** at the end of a line. The text automatically scrolls up when the bottom of the window is reached.

1. Examine your keyboard and note the different symbols.
2. Find the key for the numeral 1. Note that it is different from the letter 1.
3. **BACKSPACE** to erase the characters just typed, then type:

```

l   t   t   t   t   t   t   t   t   t   r
Preliminary report by Pat:

As you can see, I got the hang of the system right away.  It's so fast and
easy, I find it hard to believe I used to rely so heavily on personnel and
paper in the past (not so reliable, either).

I have reviewed all copy and ad designs for QuikSell as you requested.
John and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme,
but the visuals are just too stuffy for a progressive young company like
QuikSell Corp.  The photo of business people in a conference room setting
doesn't have the impact we're looking for--this client deserves something
as dynamic as their product.

I recommend a less conservative approach, although still using photography
rather than illustrations.  I envision a scene with one prop against a white
background.  Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here."  A half a dozen or more
business people are crowded around in front of the booth, waiting for their
turns to buy.  They're writing out checks or waving their checkbooks
impatiently.  The salesman is taking orders and checks as fast as he can

/usr/larry/report                               INSERT   Line   20 (   20)

```

If you make a typographical error, **BACKSPACE** to correct the error, then continue typing. If there is an error on a previous line, ignore it; you will learn how to revise it later. If you type text to the left of text already on a line, the existing text moves to the right.

4. When the text reaches the bottom of the screen the window scrolls to allow continuous typing. Continue typing the rest of the paragraph:

```

l   t   t   t   t   t   t   t   t   t   r
as dynamic as their product.

I recommend a less conservative approach, although still using photography
rather than illustrations.  I envision a scene with one prop against a white
background.  Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here."  A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy.  They're writing out checks or waving their checkbooks
impatiently.  The salesman is taking orders and checks as fast as he can
(life should be so good!).  This, I believe, is more consistent with the
image we want and certainly has the impact we're looking for.

/usr/larry/report                               INSERT   Line   22 (   22) ^

```

Note that the symbol ^ appears below the window to indicate that there is text above the text that is visible in the window. The v

symbol indicates that there is text below what appears in the window. The > symbol indicates that text extends to the right of the window, and the < symbol indicates that text extends to the left of the window. On some systems, a different set of symbols may be used to indicate text extending beyond the window borders.

2.5 Using the Cursor- and Window-Positioning Functions

All typing and editing occurs at the cursor position. The cursor- and window-positioning functions move the cursor easily within the file.

This list summarizes the cursor- and window-positioning functions:

- ↓ Moves the cursor down.
- ↑ Moves the cursor up.
- Moves the cursor to the right.
- ← Moves the cursor to the left.
- TAB** Moves the cursor forward to the next tab stop.
- TAB** Moves the cursor backward to the previous tab stop.
- HOME** Moves the cursor to the upper left-hand corner of the window.
- +LINE** Moves the text one-third of a window forward within the file.
- LINE** Moves the text one-third of a window back within the file.
- +PAGE** Moves the text one window forward within the file.
- PAGE** Moves the text one window back within the file.
- GO-TO** Moves the cursor to the first line of the file; if at the first line, moves the cursor to the last line of the file.
- LINE-FEED** Positions the cursor at the beginning of the next line of a form.
- LEFT** Moves the window to the left, displaying any text previously to the left of the window.
- RIGHT** Moves the window to the right, displaying any text previously to the right of the window.

+PAGE, **-PAGE**, **+LINE**, **-LINE**, **LEFT**, and **RIGHT** enable you to scroll through a file. **HOME** (which positions the cursor at the upper left-hand corner of the window) and **GO-TO** (which positions the cursor at the first or last line of the file) are also helpful for quick movement. When the cursor-positioning functions **↓**, **↑**, **→**, and **←** are used and the cursor reaches the boundary of the screen, it wraps around and moves to the opposite border. When the cursor is on the last line of the window and the **RETURN** key is used, the screen scrolls one line. This allows text to be entered continuously at the end of a file without having to move the window to bring in new lines.

Practice positioning the cursor within the window by using the functions described above. Use each function until you are comfortable with it.

2.6 Using Menus

Many operations in the TEN/PLUS environment are simplified through the use of menus. To select an option from a menu that appears on your screen, either position the cursor on the desired option and use **EXECUTE**, or use any of the functions **(1)** through **(8)**, depending on the number that corresponds to the line on which your choice is listed. For example, to select Create a structured file (with history) from the file creation menu, use **(2)**, since this is the second option on that menu.

2.7 Using FUNCTIONS

The ten basic functions included with the TEN/PLUS environment are **MENU**, **LOCAL-MENU**, **INSERT**, **PICK-COPY**, **PUT-COPY**, **PICK-UP**, **PUT-DOWN**, **FORMAT**, **ZOOM-IN**, and **ZOOM-OUT**. Each of these functions can only be used in certain situations and, consequently, only a subset of these functions may be active at any given time. **FUNCTIONS** displays the set of functions that are currently active. (On some systems, more than ten functions may be displayed.) For example, when editing an ASCII file, **ZOOM-IN** does not display on the menu displayed by **FUNCTIONS** because you cannot **ZOOM-IN** from the editing window of a text file.

The complete menu displayed by **FUNCTIONS** typically looks like this:

```

      Functions Menu

Move the cursor to the desired
function and touch EXECUTE.
Touch CANCEL to do nothing,
HELP for help.

MENU
LOCAL-MENU
INSERT
PICK-COPY
PUT-COPY
PICK-UP
PUT-DOWN
FORMAT
ZOOM-IN
ZOOM-OUT

```

2.8 Using the TEN/PLUS ENTER Function

ENTER is used to modify the meaning of other functions. Although you will not be using **ENTER** extensively until later, this exercise provides an example of how to use it:

1. **GO-TO** the beginning of the report file.
2. Locate the key(s) used for **ENTER** on the TEN/PLUS template.
3. Locate the key(s) used for **+LINE**. (Remember, **+LINE** scrolls the text forward in the file.)
4. Use **ENTER**; the cursor moves into a popup box and **ENTER:** displays in the box.
5. Type the number 30. Note that on some keyboards the numeric keypad is used only to access functions, in which case numeric data is entered via the numbered keys on the main keyboard. Refer to the layout template for your keyboard.
6. Use **+LINE**. The cursor advances 30 lines to the 31st line of the file.
7. The effect of **ENTER** *number* **+LINE** is different from that of **+LINE** alone. Use only **+LINE**; the text scrolls about one-third of a window forward.

Note that you can use **CANCEL** or use **ENTER** again to cancel an **ENTER** sequence at any time before completion. You can use any of these functions to edit the text typed into a popup box:

←, →
BACKSPACE
BEGIN-LINE, **END-LINE**
DELETE-CHARACTER
INSERT-MODE

2.9 Using the TEN/PLUS HELP and CANCEL Functions

As mentioned previously, **HELP** provides either general information during an editing session or specific information pertaining to the message in a popup box. Although there is no need to explore all available **HELP** topics now, it is important to review the Help Menu and become familiar with its facilities. Refer to §8.1 for additional information about **HELP**. Practice using **HELP** and **CANCEL**:

1. Use **HELP**. The Help Menu will appear on the screen. Note that various topics are covered.
2. Use **CANCEL** to remove the Help Menu.
3. Use **ENTER**. The **ENTER:** popup box will appear on the screen.
4. Use **HELP**. Note that the information now focuses on **ENTER**.
5. Use **CANCEL** to return to the **ENTER:** popup box.
6. Use **CANCEL** again to remove the popup box from the display.

2.10 Exiting a File

The screen that displays your home directory is the File Manager screen. The File Manager is the primary utility used to create, access, and delete files and directories.

When you are finished editing a file, **ZOOM-OUT** to return to the File Manager screen. You can **ZOOM-OUT** no matter where the cursor is located in the file.

2.11 Accessing Existing Files

To access a file in your home directory, position the cursor on the line on which the file is listed and **ZOOM-IN**. Because the file already exists, no menu appears.

2.12 Logging Out

If your system administrator has arranged for you to automatically log in to and log out of the TEN/PLUS environment, you can log out of the system by using **EXIT**. You know you have logged out because the system prompts for the user name of the next user with the login prompt.

If your system is not configured to allow direct entry into the TEN/PLUS environment when you log in, you must exit the TEN/PLUS environment before logging out. To exit the TEN/PLUS environment, use **EXIT**. This causes the system prompt to display. Once the prompt appears, log out by using the logout sequence for your keyboard, provided in the appropriate section of “TEN/PLUS Keyboard Information.”

Log in again with your user name and, if you have one, your password.

3. TYPING AND FORMATTING TECHNIQUES

Until now, all of the exercises in this tutorial have used unformatted text with preset margins and tabs. The INed text editor also lets you set margins and tabs and provides basic formatting facilities. These typing and formatting techniques are covered in this section.

3.1 Changing the Left-Hand Margin

When a file is created or opened, the left-hand margin is in the first column to the right of the left-hand window border. The setting is indicated by the placement of the letter l on the tab grid above the window. Access your report file:

```

l   t   t   t   t   t   t   t   t   t   r
Preliminary report by Pat:

As you can see, I got the hang of the system right away. It's so fast and
easy, I find it hard to believe I used to rely so heavily on personnel and
paper in the past (not so reliable, either).

I have reviewed all copy and ad designs for QuikSell as you requested.
John and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme,
but the visuals are just too stuffy for a progressive young company like
QuikSell Corp. The photo of business people in a conference room setting
doesn't have the impact we're looking for--this client deserves something
as dynamic as their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can

/usr/larry/report                               INSERT   Line   1 ( 22)   v

```

The left-hand margin can be set at any column to the right of column 1. The cursor must, however, be somewhere within the window. All text entered after the margin is changed wraps to the new left-hand margin.

The simplest way to set a new left-hand margin is to use **MARGIN**:

1. Move the cursor across the display to the column at which the new left-hand margin is to be set. The line on which the cursor is located does not affect the setting. The new margin affects text typed after the margin is changed regardless of where the text is typed.

2. Use **MARGIN**. The position of the 1 on the tab grid above the window will change.

To move the left-hand margin back to the left-hand edge of the window:

1. Use **←** or **-TAB** to position the cursor at the left-hand border.
2. Use **MARGIN**.

Practice changing the left-hand margin on the report file:

1. Use **ENTER** **GO-TO** to move to the end of the file.
2. **TAB** to the first tab stop and use **MARGIN**.
3. Add an indented paragraph to the end of your file:

```
      1      t      t      t      t      t      t      t      t      r
I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can
(life should be so good!). This, I believe, is more consistent with
the image we want and certainly has the impact we're looking for.

      Also, about the Cornerstone Condos concept--"Homes Built To Last"
misses the mark. These are inexpensive condos whose biggest selling
point is their cheap price and impressive list of standard features.

/usr/larry/report                INSERT      Line      26 (      26) ^
```

4. **TAB** to the second tab stop and use **MARGIN**.
5. Continue typing:

```

t   i   t   t   t   t   t   t   t   r
I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can
(life should be so good!). This, I believe, is more consistent with the
image we want and certainly has the impact we're looking for.

Also, about the Cornerstone Condos concept--"Homes Built To Last"
misses the mark. These are inexpensive condos whose biggest selling
point is their cheap price and impressive list of standard features.
They're really a "first home" investment, not the type a
buyer would plan to live in for more than 2-3 years.

/usr/larry/report                               INSERT   Line 28 ( 28) ^

```

6. Position the cursor in the column to the right of the left-hand window border and use **MARGIN**. The left-hand margin is restored to its original position.

3.2 Changing the Right-Hand Margin for Word Wrap

The right-hand margin controls the point at which text begins to wrap onto the next line. Word wrap increases typing speed by eliminating the need to press the **ENTER** key at the end of every line.

When a file is created, the right-hand margin is automatically set. The setting is indicated by the placement of the letter **r** on the tab grid above the window. Note the position of the default right-hand margin setting above the window on your display.

The right-hand margin can be removed or changed to any column between columns 2 and 200. The cursor must, however, be somewhere within the window. All text entered after the margin is changed wraps at the new right-hand margin. Having no right-hand margin set turns off word wrap. Note that to set a right-hand margin, a left-hand margin must already be set.

To create a new right-hand margin:

1. Move the cursor to the column at which the new right-hand margin is to be set. The line on which the cursor is located when the margin is set does not affect the setting. The new margin affects text typed after the margin is changed regardless of where the text is typed.

New left- and right-hand margins set during an editing session last only until you exit the TEN/PLUS environment. If you exit the TEN/PLUS environment and then reaccess a file, the margins are set to the default columns. Reset the left- and right-hand margins as required.

3.3 Changing Both Margins Simultaneously

You can change both the left- and right-hand margins in one step by cursor-defining the width and positioning of the desired text block:

1. Move the cursor to the column in which the new left-hand margin is to be located, then use **ENTER**.
2. Use **→** or **TAB** to move the cursor to the column in which the new right-hand margin is to be set.
3. Use **MARGIN**.

You can turn off word wrap by clearing the margins. To clear the margins:

1. Position the cursor in the same column as the left-hand margin.
2. Use **ENTER MARGIN**. Note that the **l** and **r** markers on the tab grid are removed.

To reset the default margins after clearing them:

1. Move the cursor to the default right-hand margin position.
2. Use **ENTER MARGIN**. Note that the margins must be cleared before **ENTER MARGIN** can be used to restore the default margins.

Practice clearing the margins and setting new ones:

1. With the cursor at the left-hand margin, use **ENTER MARGIN**.
2. **TAB** to the first tab stop, then use **ENTER**.
3. **TAB** to the eighth tab stop, then use **MARGIN**.

3.4 Readjusting Lines on New Margins

FORMAT is used to adjust lines according to a new margin setting. When **FORMAT** is used, the lines from the cursor position to the next blank line are adjusted according to the current margin settings.

To readjust lines within new margins:

1. **GO-TO** the beginning of your file.
2. Position the cursor on the first line of the second paragraph. One or more text lines followed by a blank line are considered to be a paragraph.
3. Use **FORMAT**. Note that **FORMAT** does not change the indentation of the first line of the paragraph. Move the line to the new margin by using the space bar.
4. Use **FORMAT** to fill the text again:

```

1      t      t      t      t      t      t      r      t
Preliminary report by Pat:

    As you can see, I got the hang of the system right away.
    It's so fast and easy, I find it hard to believe I used
    to rely so heavily on personnel and paper in the past
    (not so reliable, either).

I have reviewed all copy and ad designs for QuikSell as you requested.
John and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme,
but the visuals are just too stuffy for a progressive young company like
QuikSell Corp. The photo of business people in a conference room setting
doesn't have the impact we're looking for--this client deserves something as
dynamic as their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks

/usr/larry/report                INSERT      Line      3 ( 39)  v

```

5. Reset both margins so that they are each indented approximately two inches.
6. Move the cursor to the third line of the next paragraph in the file, then use **FORMAT**. Note that the lines above the cursor are not affected.

7. Reset the margins to the default columns and use **FORMAT** to reformat the text, including the paragraphs at the end of the file.
8. Where necessary, adjust the first line in each paragraph by positioning the cursor on the first character in the line and using **BACKSPACE** to move the line to the left-hand margin; use **FORMAT** again.

3.5 Centering Text

CENTER centers a single line of text between the current left- and right-hand margins. To center a single line:

1. Position the cursor on the line to be centered.
2. Use **CENTER**.

To center multiple lines, cursor-define the lines to be centered before using **CENTER**:

1. Move the cursor to the first line to be centered, then use **ENTER**.
2. Use **↓** to move the cursor down to the last line to be centered, then use **CENTER**. Note that the lines defined by the cursor are centered.

Practice this procedure at the end of the `report` file:

1. Use **ENTER** **GO-TO** to go to the end of the file.
2. Type the new text exactly as shown below, pressing the **ENTER** key at the end of each line:

```

1   t   t   t   t   t   t   t   t   t   t   r
"first home" investment, not the type a buyer would plan to live in for more
than 2-3 years.

We've got to go for the first-time buyer, convince him that this is the
opportunity he's been waiting for to make the jump from renter to homeowner.
He couldn't care less about how long the home will last. He wants a
lifestyle and a low down payment. I say we go with a headline that expands
this message, such as "Our long list of standard features will raise your
standard of living more than you ever thought possible--for just $1,500
down." (I'll buy two for my kids right now!)

Preliminary comments from Larry:

After a brief review of the accounts, it
seems to me Pat's insight and recommendations
are all valid. I'll begin working up some
new copy for the campaign.

/usr/larry/report                INSERT   Line   43 (   43) ^ v

```

3. With the cursor on the line containing the word Preliminary, use **ENTER**.
4. Move the cursor down to the last line of new text, then use **CENTER**:

```

1   t   t   t   t   t   t   t   t   t   t   r
"first home" investment, not the type a buyer would plan to live in for more
than 2-3 years.

We've got to go for the first-time buyer, convince him that this is the
opportunity he's been waiting for to make the jump from renter to homeowner.
He couldn't care less about how long the home will last. He wants a
lifestyle and a low down payment. I say we go with a headline that expands
this message, such as "Our long list of standard features will raise your
standard of living more than you ever thought possible--for just $1,500
down." (I'll buy two for my kids right now!)

                Preliminary comments from Larry:

                After a brief review of the accounts, it
                seems to me Pat's insight and recommendations
                are all valid. I'll begin working up some
                new copy for the campaign.

/usr/larry/report                INSERT   Line   38 (   43) ^ v

```

3.6 Changing Tab Stops

The editor has default tab stops indicated by the markers above the window. The default tabs stops are located eight columns apart. During an editing session, individual tabs can be cleared or set, but the default tab stops are reinstated whenever you exit the TEN/PLUS environment and then reaccess a file.

To clear a tab stop:

1. **TAB** to the appropriate tab stop.
2. Use **ENTER SET-TAB**.

To set individual tabs:

1. Position the cursor at the appropriate column within the window.
2. Use **SET-TAB**. Note that **SET-TAB** will only work when the cursor is positioned within the window.

Practice clearing and setting tab stops:

1. **TAB** to the first tab stop.
2. Use **ENTER SET-TAB**.
3. Repeat for the next two tab stops.
4. Set tabs 10 and 20 characters from the left-hand margin.

3.7 Underscoring With FONT

Most displays are equipped with four fonts: one for typing nonunderscored words; one for typing underscored words, spaces, and punctuation (continuous underscoring); one for typing underscored words only (word underscoring); and one for accessing a graphics character set (if one is available on your display). **FONT** is used in conjunction with **ENTER** to select the various fonts for input. Text already entered cannot be modified with **FONT**. On displays that cannot show underlining, underlined characters are added correctly to the file.

To select continuous underscoring, use **ENTER c FONT**. To select word underscoring, use **ENTER w FONT**. To select nonunderscoring, use **ENTER FONT**. The graphics characters are described in the “TEN/PLUS Reference Manual.” Although graphics characters may be viewed on the display, the ability to

print them is printer dependent. To select the graphics font, use **ENTER** g **FONT**.

Once a font has been selected, **FONT** can be used to switch between that font (the “active” font) and the font selected immediately beforehand (the “previously active” font). The nonunderscore (or Roman) font is active whenever you enter the editor. The default previously active font is continuous underscore.

Selection of active and previously active fonts permits all possible combinations of fonts. If, for example, you select word-underscore font while the nonunderscore font is active, you switch between nonunderscored text and word-underscored text each time you use **FONT**. If you select the word-underscore font while the continuous-underscore font is active, you switch between word underscoring and continuous underscoring each time you use **FONT**. Remember, you can always return to the default nonunderscore font by using **ENTER** **FONT**.

It is important for you to feel comfortable using the techniques you have just learned. Do this exercise to be sure you know how to use all of these facilities:

1. Use **ENTER** **FONT** to return to nonunderscore font, then **ZOOM-OUT** of your file.
2. Create a new file named memo.
3. Practice typing text in the various fonts. When you are accustomed to switching between the various fonts, enter some text as follows:
 - a. Insert a centered, underscored title.
 - b. Type one paragraph using the default margins.
 - c. Indent the left- and right-hand margins and type another paragraph.
 - d. Underline some of the text as you are typing.
 - e. Change the margins back to their original setting.
 - f. Change the tab stops and type a chart.

4. BASIC EDITING TECHNIQUES

The previous sections explained how to create, type, and format files. This section shows how to revise (or *edit*) files.

These functions are used when making revisions:

DELETE-CHARACTER

Deletes characters.

INSERT-MODE

Alternates between insert mode and overwrite mode.

INSERT

Creates a blank line.

PICK-UP

Picks up a full line from the text and adds it to the “pick buffer.”

PUT-DOWN

Returns last “picked” line to the file.

DELETE

Deletes a full line and adds it to the “wastebasket buffer.”

RESTORE

Returns last **DELETE**d line to the file.

PICK-COPY

Copies a full line and adds it to the pick buffer.

PUT-COPY

Copies the last picked line to the file.

The purpose of each of these functions, when used alone or in conjunction with **ENTER**, is explained more fully later in this section.

To revise a file:

1. Access the file requiring revision.
2. Position the cursor at the location of the desired revision using the cursor- and window-positioning functions.
3. Make the revision.

Before continuing, locate the keys used for **DELETE-CHARACTER**, **INSERT-MODE**, **INSERT**, **PICK-UP**, **PUT-DOWN**, **DELETE**, **RESTORE**, **PICK-COPY**, and **PUT-COPY**. Note that some of these functions can also be invoked using **FUNCTIONS**, which displays a menu showing which of the ten basic TEN/PLUS functions are active.

4.1 Deleting Characters or Words Within a Line

DELETE-CHARACTER deletes individual characters or, when used repetitively, strings of characters within a line. As the characters are deleted, the rest of the line to the right of the deletion point moves toward the cursor.

Practice using **DELETE-CHARACTER**:

1. Access the report file.
2. With the cursor on the third line of the file, delete the words right away by positioning the cursor at the space before the r in right, then using **DELETE-CHARACTER** repeatedly until the words are removed and the period is at the cursor position. Note that the characters to the right of the cursor move one space to the left for each character deleted:

```

1      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:

As you can see, I got the hang of the system. It's so fast and
easy, I find it hard to believe I used to rely so heavily on personnel and
paper in the past (not so reliable, either).

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like
QuikSell Corp. The photo of business people in a conference room setting
doesn't have the impact we're looking for--this client deserves something as
dynamic as their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can

/usr/larry/report                INSERT   Line      3 (   43)  v

```

3. On the fifth line of the file, position the cursor at the space just after the t in past and use **DELETE-CHARACTER** to remove (not so reliable either). Note that strings can be removed more quickly when **DELETE-CHARACTER** is held down.

4.2 Adding Characters or Words Within a Line

When a file is opened, the editor is in insert mode. (The system informs you of this by displaying `INSERT` below the window.)

When the editor is in insert mode, all input is inserted at the cursor position and existing text is moved to the right (or wrapped onto the next line). To add characters or words within a line without overstriking characters, use insert mode.

If the right-hand margin has been removed (word wrap is off) and the editor is in insert mode, a line of text can extend beyond the right-hand window border. You can view the part of the line that has moved off the display by using `RIGHT` to move the window. If there is a right-hand margin (word wrap is on) and the editor is in insert mode, the text wraps at the margin. Use `FORMAT` as required to reformat the text within each paragraph according to the margins set on the display.

When insert mode is in effect, you can correct typing errors by typing the correct characters and using `DELETE-CHARACTER` to remove the incorrect ones. You may prefer to keep insert mode in effect throughout revision.

1. Deactivate word wrap by removing the margins as described in §3.3.
2. On the third line, position the cursor on the period after the word `system`.
3. Insert the text `in just a few short hours`. The text to the right of the insert moves beyond the right-hand window border:

```

t      t      t      t      t      t      t      t      t
Preliminary report by Pat:

As you can see, I got the hang of the system in just a few short hours. It's
easy, I find it hard to believe I used to rely so heavily on personnel and
paper in the past.

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like
QuikSell Corp. The photo of business people in a conference room setting
doesn't have the impact we're looking for--this client deserves something as
dynamic as their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can

/usr/larry/report                               INSERT   Line   3 (   43)  v  >

```

Note that the > symbol appears below the window to indicate that there is text to the right of the border, and that the text so fast and is not visible in the window.

- Use **RIGHT** to see the text to the right of the window:

```

t      t      t      t      t      t      t      t      t
hang of the system in just a few short hours. It's so fast and
lieve I used to rely so heavily on personnel and

nd ad designs for QuikSell as you requested. John
Makes the Hard Sell Easy" is a great theme, but
tuffy for a progressive young company like
of business people in a conference room setting
're looking for--this client deserves something as

ative approach, although still using photography
I envision a scene with one prop against a white
the picture is a salesman standing behind a
ppears a sign "Buy Here." A half a dozen or more
d around in front of the booth, waiting for their
ting out checks or waving their checkbooks
is taking orders and checks as fast as he can

/usr/larry/report                               INSERT   Line   3 (   43)  v  <

```

Note that the < symbol now appears below the window to indicate that there is text to the left of the window.

5. Use **LEFT** to move the window back to its original position.
6. Reset the default margins to reactivate word wrap.

4.3 Correcting Transpositions and Typographical Errors

When the editor is in overwrite mode it is possible to correct typographical errors by typing the correct characters over the incorrect ones. As each character is typed, it *overwrites* the incorrectly typed character at the cursor position. Use **INSERT-MODE** to switch into overwrite mode. This causes the word **INSERT** to be replaced by the word **OVERWRITE** below the window. Use **INSERT-MODE** to alternate between insert mode and overwrite mode. Delete extra characters by using **DELETE-CHARACTER**. Use **ENTER** **DELETE-CHARACTER** to delete text from the cursor position to the end of a line.

1. Switch into overwrite mode by using **INSERT-MODE**. Note that **OVERWRITE** replaces **INSERT** below the window.
2. Position the cursor at the period after the word *past* on the fifth line, use **→** to move the cursor three columns to the right, and type (including the transposition in *thnigs*):

```
I can already tell that this system is going to speed  
thnigs up immensely on my part.
```

3. Position the cursor at the *i* in *is* on the fifth line and type *will*. Note that each character typed overwrites the character at the cursor position.
4. Use **DELETE-CHARACTER** to delete the extra characters at the end of *going* (*oing*) and the word *to*.
5. Position the cursor at the *n* in *thnigs* and type *in*.
6. On the sixth line, position the cursor at the space after *immensely* and type a period.
7. Use **ENTER** **DELETE-CHARACTER** to delete the end of the line.
8. Press the **ENTER** key, then use **INSERT**. The screen should now look like this:

```

1      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:

As you can see, I got the hang of the system in just a few short hours.  It's
easy, I find it hard to believe I used to rely so heavily on personnel and
paper in the past.  I can already tell that this system will speed
things up immensely.

I have reviewed all copy and ad designs for QuikSell as you requested.  John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like
QuikSell Corp.  The photo of business people in a conference room setting
doesn't have the impact we're looking for--this client deserves something as
dynamic as their product.

I recommend a less conservative approach, although still using photography
rather than illustrations.  I envision a scene with one prop against a white
background.  Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here."  A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy.  They're writing out checks or waving their checkbooks

/usr/larry/report          OVERWRITE   Line      6 (   44)  v  >

```

9. Use **INSERT-MODE** to switch back into insert mode.

4.4 Moving Windows

If a right-hand margin has not been set and words are inserted within a line, the line often extends past the right-hand window border. Use **RIGHT** to move the window to the right so that the text beyond the border is visible. Use **LEFT** to move the window back to its original position. File width can exceed the width of the window. In such cases, only part of the width appears on the display at any one time.

RIGHT and **LEFT** also allow you to type text, such as a statistical chart, that is wider than the window. You may even set a right-hand margin (**ENTER MARGIN**) that is wider than the window. The window, in this case, automatically scrolls right and left as the system word wraps the text.

Like the other scrolling functions, **RIGHT** and **LEFT** accept numeric arguments when used in conjunction with **ENTER**. For example, to scroll the window 50 columns to the right, use **ENTER 50 RIGHT**. To return the window to its original position, in lieu of repeatedly using **LEFT**, use **ENTER 50 LEFT**. (Any number greater than 50 will work as well.)

This exercise demonstrates the procedures for moving windows:

1. Use **RIGHT** to move the window so that you can see the text extending beyond the right-hand window border.
2. Use **LEFT** to move the window back to its original position.
3. Use **ENTER** 50 **RIGHT** to move the window 50 columns to the right:

```

t   t   t   r   t   t   t   t   t   t
-----
st a few short hours. It's so fast and
heavily on personnel and
system will speed

ll as you requested. John
y" is a great theme, but
young company like
conference room setting
ient deserves something as

still using photography
h one prop against a white
n standing behind a
" A half a dozen or more
booth, waiting for their
g their checkbooks

/usr/larry/report                INSERT   Line   1 ( 44)  v <

```

4. Use **ENTER** 50 **LEFT** to return the window to its original position.
5. Move the cursor to the first line of the paragraph beginning with **As you can see** and use **FORMAT**. Note that the text beyond the window border is now wrapped within the right-hand margin.

4.5 Deleting or Moving Lines

There are two ways to delete lines from a file: **DELETE** and **PICK-UP**. Use **DELETE** if you wish to remove a line permanently. Use **PICK-UP** if you plan to put the line down somewhere else. A line that has been picked up with **PICK-UP** and has not been **PUT-DOWN** or copied has effectively been deleted. Note that you can recover **DELETE**d lines with **RESTORE**.

4.5.1 Using DELETE

DELETE removes the line containing the cursor, adds it to the wastebasket buffer, and moves the remaining lines up. To recover text deleted the last time **DELETE** was used, use **RESTORE**. **RESTORE** removes text from the wastebasket buffer and replaces it in the file. Text that has not been **RESTORE**d remains in the buffer.

Practice moving or deleting lines using **DELETE**:

1. Use **DELETE-CHARACTER** to delete the phrase *It's so fast and easy*, beginning on the third line, then use **DELETE** to delete the line beginning *As you can see*.
2. Use **DELETE** to remove the next four lines. The screen should now look like this:

```

1      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:

I have reviewed all copy and ad designs for QuikSell as you requested.  John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like QuikSell
Corp.  The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations.  I envision a scene with one prop against a white
background.  Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here."  A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy.  They're writing out checks or waving their checkbooks
impatiently.  The salesman is taking orders and checks as fast as he can
(life should be so good!).  This, I believe, is more consistent with the
image we want and certainly has the impact we're looking for.

Also, about the Cornerstone Condos concept--"Homes Built To Last" misses the
/usr/larry/report                               INSERT      Line      3 (      39)      v

```

3. Use **RESTORE** five times to recover the **DELETE**d lines, then use **FORMAT**.

4.5.2 Using PICK-UP

Like **DELETE**, **PICK-UP** removes the line containing the cursor and moves the remaining lines up, but it places the line in the pick buffer rather than in the wastebasket buffer. Recover picked-up text with **PUT-DOWN**, which removes the text from the pick buffer

and replaces it in the file. Any text that is not **PUT-DOWN** remains in the buffer.

Practice deleting and restoring lines using **PICK-UP**:

1. Position the cursor on the third line in the file and use **PICK-UP** five times. All five lines are removed:

```

1      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like QuikSell
Corp. The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can
(life should be so good!). This, I believe, is more consistent with the
image we want and certainly has the impact we're looking for.

Also, about the Cornerstone Condos concept--"Homes Built To Last" misses the

/usr/larry/report                               INSERT   Line      3 (   39)   v

```

2. Use **PUT-DOWN** five times. All five lines are retrieved in reverse order from the pick buffer and the file is restored to its previous state.
3. Repeat steps 1 and 2 using **DELETE** and **RESTORE** instead of **PICK-UP** and **PUT-DOWN**.

Although **PICK-UP** and **DELETE** are similar in function, they store text in different buffers. This permits you to **PICK-UP** text for a move and continue editing while making permanent deletions with **DELETE** without disturbing your pick buffer. You can then **PUT-DOWN** the text at the appropriate location.

4.6 Adding Blank Lines and Splitting Lines

INSERT inserts a blank line above the cursor line and moves the rest of the lines down. **ENTER** **INSERT** splits the current line at the cursor position and moves the text to the right of the cursor to the next line. **ENTER** *number* **INSERT**, where *number* is the

number of lines, inserts the specified number of lines at the cursor position and moves the rest of the lines down.

Practice adding blank lines and splitting lines:

1. With the cursor on the last I on the third line, use **ENTER** **INSERT**. The line is split at the cursor position.
2. With the cursor on the fourth line, use **INSERT**. A blank line appears at the cursor position and the text on and below the cursor line is moved down one line. The cursor is positioned on the blank line at the column in which it was located when **INSERT** was used.
3. Position the cursor at the beginning of the blank line and type
It is so easy to use.
4. Position the cursor at the I on the third line, and split the line by using **ENTER** **INSERT**. The part of the line beginning with I moves to the next line and the cursor is located one space after the comma:

```

1      t      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:

As you can see,
I got the hang of the system in just a few short hours.
It is so easy to use
I
find it hard to believe I used to rely so heavily on personnel and paper in
the past. I can already tell that this system will speed things up
immensely.

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like QuikSell
Corp. The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a

/usr/larry/report                INSERT      Line      3 ( 47)  v

```

5. Type so far things are going very well.
6. On the second line, use **ENTER** 5 **INSERT**. Five lines are inserted at the cursor position and the other lines in the file are moved down.

- Use **ENTER** 5 **PICK-UP** to remove the five lines and move the rest of the lines in the file up.

4.7 Joining Lines

Sometimes it is necessary to join two adjacent lines. This can be done with **FORMAT**, but **FORMAT** fills all lines in the paragraph to the current margins. Use **ENTER** **PICK-UP** or **ENTER** **DELETE** in situations where specific lines are to be joined and the rest of the text should not be filled, as in a list:

- Position the cursor at the end of the fifth line, two spaces after the e in use.
- Use **ENTER** **PICK-UP**. The line is joined with the line below:

```

1      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:

As you can see, so far things are going well.
I got the hang of the system in just a few short hours.
It is so easy to use I
find it hard to believe I used to rely so heavily on personnel and paper in
the past. I can already tell that this system will speed things up
immensely.

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like QuikSell
Corp. The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more

/usr/larry/report                               INSERT   Line   5 ( 46) v

```

- Position the cursor on the third line and use **FORMAT**. Note that all lines in the paragraph are joined.

4.8 Defining Rectangular Areas Using BOX-MARK

It is possible to use the cursor-positioning functions to cursor-define a rectangular area to be moved, inserted, deleted, or copied. (Copying is covered later in this tutorial.) Cursor-defining is used extensively to move or duplicate words, columns, lines, or paragraphs.

Note that you can use **CANCEL** or **ENTER** to cancel a **BOX-MARK** sequence at any time before completion.

To cursor-define and then move a rectangular area:

1. Position the cursor at the beginning of the area to be defined and use **BOX-MARK**. A **BOX-MARK** indicator appears on the character under the cursor; depending on the display, the indicator may consist of highlighting, blinking, or the replacement of the character under the cursor by a graphics character. Also, ****** BOX/LINE ****** appears below the window on the left-hand side of the display.
2. Use the cursor-positioning functions to define the area to be affected. Words, lines, or columns can be cursor-defined. To define complete lines, use **↓** to move the cursor down through the area to be affected. Use **+LINE** or **+PAGE** to scroll forward if the area extends beyond the display. To define partial lines, use **→** to move the cursor across the area to be defined. Note that when you define and **PUT-DOWN** a rectangular area of text that includes partial lines, such as columns, text in the cursor column and to the right is displaced. Refer to §4.16 for more information about using **BOX-MARK** to define columns. Some displays highlight the text being defined; others only indicate where the definition begins and ends.
3. Use **PICK-UP**. The text is removed and the cursor returns to its original position.
4. Move the cursor to a new location and use **PUT-DOWN**. The text is removed from the pick buffer and inserted at the cursor location.

Practice moving text using **BOX-MARK**:

1. With the cursor at the beginning of the third line, use **BOX-MARK**. Note that the defined area begins with the line on which the cursor is initially positioned.
2. Use **↓** to move the cursor four lines down to the blank line, then use **PICK-UP**. The five lines are removed:

```

1   t   t   t   t   t   t   t   t   t   t   r
Preliminary report by Pat:

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like QuikSell
Corp. The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with one prop against a white
background. Prominent in the picture is a salesman standing behind a
tradeshow booth on which appears a sign "Buy Here." A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can
(life should be so good!). This, I believe, is more consistent with the
image we want and certainly has the impact we're looking for.

Also, about the Cornerstone Condos concept--"Homes Built To Last" misses the

/usr/larry/report                INSERT Line      3 ( 39)  v

```

3. Use **+PAGE**, position the cursor on line 34, and use **PUT-DOWN**. The lines in the pick buffer are restored and moved to the cursor position.

4.9 Adding Lines Using BOX-MARK

Usually **INSERT-MODE** is used for typing words within lines, and **INSERT** is used to add one line at a time. For major additions, either add text while insert mode is in effect or **INSERT** a few lines before beginning to type.

INSERTing a large area helps prevent typing over the material that follows the insertion. If there are extra blank lines remaining when the insertion has been typed, **DELETE** them. To **INSERT** a large area quickly, cursor-define the area by using **I**. Use **+LINE** or **+PAGE** to scroll forward if the defined area extends beyond the display. Cursor-defining to create blank lines requires three steps:

1. Position the cursor at the beginning of the area to be opened and use **BOX-MARK**.
2. Use **I** to move the cursor down the remaining number of lines to be inserted.
3. Use **INSERT**. The cursor returns to its original position and blank lines are available for input.

Practice inserting lines using **BOX-MARK**:

1. Use **+LINE**.
2. Position the cursor on line 39 and use **BOX-MARK**.
3. Use **↓** to move the cursor six lines down.
4. Use **INSERT**. Seven blank lines are inserted in the text:

```

l   t   t   t   t   t   t   t   t   t   r
He couldn't care less about how long the home will last. He wants a
lifestyle and a low down payment. I say we go with a headline that expands
this message, such as "Our long list of standard features will raise your
standard of living more than you ever thought possible--for just $1,500
down." (I'll buy two for my kids right now!)

As you can see, so far things are going well. I got the hang of the system
in just a few short hours. It is so easy to use I find it hard to believe I
used to rely so heavily on personnel and paper in the past. I can already
tell that this system will speed things up immensely.

Preliminary comments from Larry:

/usr/larry/report          INSERT   Line   39 (   50) ^ v

```

5. Move the cursor to the beginning of the current line and type the new text as shown below:

```

l   t   t   t   t   t   t   t   t   t   r
He couldn't care less about how long the home will last. He wants a
lifestyle and a low down payment. I say we go with a headline that expands
this message, such as "Our long list of standard features will raise your
standard of living more than you ever thought possible--for just $1,500
down." (I'll buy two for my kids right now!)

As you can see, so far things are going well. I got the hang of the system
in just a few short hours. It is so easy to use I find it hard to believe I
used to rely so heavily on personnel and paper in the past. I can already
tell that this system will speed things up immensely.

Preliminary comments from Janet:

I read it all, and it looks ok to me. Pat's ideas on the campaign seemed a
bit odd to me. Are we going to get together in the next few days to discuss
all this?

Preliminary comments from Larry:

/usr/larry/report          INSERT   Line   43 (   50) ^ v

```

4.10 Deleting and Duplicating Lines Using BOX-MARK

There are two methods that can be used to delete a paragraph or section from a file. The first is to delete each line individually, and the second is to delete all lines at one time. The first, which uses **DELETE**, may be sufficient for short blocks of text, but is time-consuming if the block is lengthy. The second, which uses **BOX-MARK**, is more efficient for larger blocks of text.

To use **BOX-MARK** to delete a section of text:

1. Position the cursor at the beginning of the section to be deleted and use **BOX-MARK**.
2. Move the cursor down through the end of the section to be deleted. When deleting a paragraph, include the blank line at the end of the paragraph.
3. Use **DELETE**.

The section is deleted and put into the wastebasket buffer. Use **RESTORE** to recover text stored in the wastebasket buffer. Later in this tutorial, you will use this procedure in conjunction with **PICK-UP** and **PUT-DOWN** to move paragraphs, and you will use **PICK-COPY** and **PUT-DOWN** to duplicate paragraphs.

Practice using **BOX-MARK** to delete text:

1. Move the cursor to the beginning of line 34 and use **BOX-MARK**.
2. Use **↓** to move the cursor four lines down to the next blank line and use **DELETE**. The paragraph is deleted.

4.11 Defining Nonrectangular Areas Using TEXT-MARK

A different method of cursor-defining is used to insert, delete, or copy nonrectangular areas of text. This method is usually used on sentences but can also be used on words or lines. Note that you can use **CANCEL** or **ENTER** to cancel a **TEXT-MARK** sequence at any time before completion.

To cursor-define and move a nonrectangular area:

1. Position the cursor at the beginning of the area to be defined and use **TEXT-MARK**. A display-dependent indicator appears at the cursor position and ******* TEXT ******* appears below the window on the left-hand side of the display whenever **TEXT-MARK** is used.
2. Use the cursor-positioning functions to define the area to be affected. Use **↓** or press the **ENTER** key to define full lines and **→**, **←**, **TAB**, or **-TAB** to define partial lines. When partial lines are defined, the character at the final cursor position is not part of the defined area. Some displays highlight the text being defined; others only indicate where the definition begins and ends.
3. Use **PICK-UP**. The text is removed, and the cursor returns to its original position.
4. Move the cursor to the insertion location and use **PUT-DOWN**. The text is removed from the pick buffer and inserted at the cursor location. Text at and to the right of the cursor position is moved right.

Practice using **TEXT-MARK** to move a nonrectangular area:

1. With the cursor on line 36, position the cursor at the P in Pat and use **TEXT-MARK**.
2. Press the **ENTER** key, then use **→** to move the cursor to the A in Are .
3. Use **PICK-UP**. The sentence is removed, and the text that was previously located after the sentence moves to the cursor

position. Notice that line 36 extends beyond the right-hand window margin.

4. Use **FORMAT** to readjust the text.
5. Position the cursor on line 37, three spaces after the question mark, and use **PUT-DOWN**. The text is removed from the wastebasket buffer and is inserted in the file.

4.12 Deleting or Moving Sentences Using TEXT-MARK

When deleting or moving text areas that include both full and partial lines, the fastest and most efficient way to cursor-define the area is to use **TEXT-MARK**. The procedure for deleting or moving a block of text is:

1. Position the cursor on the first character to be deleted or moved and use **TEXT-MARK**.
2. Press the **ENTER** key through any full lines to be deleted or moved.
3. Position the cursor on the first character *not* to be deleted or moved.
4. Use **DELETE** to remove the block; use **PICK-UP** to move the block.

If **DELETE** is used, the section is placed in the wastebasket buffer. Recover text stored in the wastebasket buffer with **RESTORE**. If **PICK-UP** is used, the section is placed in the pick buffer. Recover text stored in the pick buffer with **PUT-DOWN**.

Practice using **TEXT-MARK** on the report file:

1. **GO-TO** the beginning of the file.
2. Position the cursor on the I on line 11 and use **TEXT-MARK**.
3. Press the **ENTER** key, then use **→** to move the cursor to the P in Prominent.
4. Use **PICK-UP**.
5. Position the cursor at the A on line 12 and use **PUT-DOWN**. The sentence is inserted:

```

1   t   t   t   t   t   t   t   t   t   t   r
Preliminary report by Pat:

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like QuikSell
Corp. The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. Prominent in the picture is a salesman standing be
tradeshow booth on which appears a sign "Buy Here." I envision a scene with o
background. A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can
(life should be so good!). This, I believe, is more consistent with the
image we want and certainly has the impact we're looking for.

Also, about the Cornerstone Condos concept--"Homes Built To Last" misses the

/usr/larry/report                INSERT   Line   12 (   45)  v  >

```

Note that the end of the third line in the paragraph is outside the right-hand window border.

6. Use **FORMAT** to move the text inside the window.
7. Position the cursor at the P in Prominent on line 11 and use **TEXT-MARK**.
8. Press the **ENTER** key, then use **→** to move the cursor to the I at the beginning of the next sentence.
9. Use **PICK-UP**. The marked text is deleted:


```

1      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:

I have reviewed all copy and ad designs for QuikSell as you requested. John
and I agree that "QuikSell Makes the Hard Sell Easy" is a great theme, but
the visuals are just too stuffy for a progressive young company like QuikSell
Corp. The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a
scene with one prop against a white background. A half a dozen or more
business-people are crowded around in front of the booth, waiting for their
turns to buy. They're writing out checks or waving their checkbooks
impatiently. The salesman is taking orders and checks as fast as he can
(life should be so good!). This, I believe, is more consistent with the
image we want and certainly has the impact we're looking for.

Also, about the Cornerstone Condos concept--"Homes Built To Last" misses the
mark. These are inexpensive condos whose biggest selling point is their

/usr/larry/report                                INSERT      Line      11 (      44)      v

```

10. Use **FORMAT** to fill the paragraph.

4.13 Moving Words, Lines, and Paragraphs

PICK-UP is used to move words, lines, or sentences. It removes the text and places it in the pick buffer. To relocate the text, position the cursor at the new location and use **PUT-DOWN**. The last picked-up text is placed at the cursor position. To move a single line, use **PICK-UP**, position the cursor at the new location, and use **PUT-DOWN**. To move words, sentences, lines, or paragraphs, use **BOX-MARK** or **TEXT-MARK** to cursor-define the area before using **PICK-UP**.

Practice using **PICK-UP** and **PUT-DOWN** to move single lines:

1. Use **ENTER** **GO-TO** to move to the end of the file.
2. Beginning on line 47, type:

```

1   t   t   t   t   t   t   t   t   t   t   r
I read it all, and it looks ok to me. Are we going to get together in the
next few days to discuss all this? Pat's ideas on the campaign seemed a
bit odd to me.

        Preliminary comments from Larry:

                After a brief review of the accounts, it
                seems to me Pat's insight and recommendations
                are all valid. I'll begin working up some
                new copy for the campaign.

We should meet with the following people before the end of the week:

J. Winkler
L. Larson
D. Adams
L. Pantages
P. Sternig

/usr/larry/report                INSERT   Line   53 (   54) ^

```

3. Alphabetize the list by using **PICK-UP** on each line to be moved, positioning the cursor at the location where you want the line moved, and using **PUT-DOWN**:

```

1   t   t   t   t   t   t   t   t   t   t   r
I read it all, and it looks ok to me. Are we going to get together in the
next few days to discuss all this? Pat's ideas on the campaign seemed a
bit odd to me.

        Preliminary comments from Larry:

                After a brief review of the accounts, it
                seems to me Pat's insight and recommendations
                are all valid. I'll begin working up some
                new copy for the campaign.

We should meet with the following people before the end of the week:

D. Adams
L. Larson
L. Pantages
P. Sternig
J. Winkler

/usr/larry/report                INSERT   Line   52 (   52) ^ v

```

4.14 Duplicating Words, Lines, and Paragraphs

PICK-COPY and **PUT-COPY** are used to pick up a copy of text (*without* altering the file) and subsequently put one or more copies

down. **PUT-COPY** differs from **PUT-DOWN** in that **PUT-COPY** does not remove the text from the pick buffer. To copy a single line, use **PICK-COPY**, position the cursor at the new location, and use **PUT-COPY** or **PUT-DOWN**. To copy words, sentences, multiple lines, or paragraphs, use **BOX-MARK** or **TEXT-MARK** to cursor-define the area before using **PICK-COPY**.

If multiple copies of the text are needed, position the cursor where the picked text is to be placed and use **PUT-COPY**. The picked text can be copied into the file as many times as needed; it stays at the top of the pick buffer until you:

1. Use **PICK-UP** or **PICK-COPY** to place new text at the top of the pick buffer.
2. Use **PUT-DOWN**, thereby removing it from the pick buffer.

Practice using **PICK-COPY** and **PUT-COPY** to duplicate sentences and paragraphs:

1. Position the cursor at the w in We on line 47, then use **BOX-MARK**.
2. Use **↓** to move the cursor six lines down to the J in J. Winkler, then use **PICK-COPY**. The lines are stored in the pick buffer.
3. Position the cursor at the beginning of line 39, then use **PUT-COPY**. The lines are copied from the pick buffer into the file:

```

1      t      t      t      t      t      t      t      t      t      t      r
I read it all, and it looks ok to me.  Are we going to get together in the
next few days to discuss all this?  Pat's ideas on the campaign seemed a
bit odd to me.

We should meet with the following people before the end of the week:

D. Adams
L. Larson
L. Pantages
P. Sternig
J. Winkler

                Preliminary comments from Larry:

                After a brief review of the accounts, it
                seems to me Pat's insight and recommendations
                are all valid.  I'll begin working up some
                new copy for the campaign.

We should meet with the following people before the end of the week:

/usr/larry/report                                INSERT   Line   39 (   60) ^ v

```

4.15 Inserting, Moving, Deleting, and Duplicating a Specific Number of Lines

A simple way to **INSERT** blank lines when inserting a block of text is to use a numeric argument specifying the number of lines to be inserted. (Excess blank lines can be **DELETE**d after typing the insertion.) Numeric arguments can also be used when moving, deleting, or duplicating a specific number of lines.

To insert, move, delete, or duplicate a specific number of lines:

1. Use **ENTER**.
2. Type a numeric argument indicating the number of lines to be affected.
3. Use **INSERT**, **PICK-UP**, **DELETE**, or **PICK-COPY**.

If this method is used to **PICK-COPY**, **DELETE**, or **PICK-UP** lines, **PUT-COPY**, **RESTORE**, or **PUT-DOWN** can be used to move or copy the lines.

Practice using **ENTER** with a numeric argument to delete a specific number of lines:

1. Use **+LINE**.
2. Position the cursor at the W in We on line 54.

3. Use **ENTER**.
4. Type the number 7.
5. Use **DELETE**. The seven lines beginning with the cursor line are removed from the file and stored in the wastebasket buffer:

```

1   t   t   t   t   t   t   t   t   t   r
L. Larson
L. Pantages
P. Sternig
J. Winkler

      Preliminary comments from Larry:

      After a brief review of the accounts, it
      seems to me Pat's insight and recommendations
      are all valid. I'll begin working up some
      new copy for the campaign.

/usz/larry/report                INSERT   Line   54 (   53) ^

```

4.16 Opening, Moving, Deleting, and Duplicating Columns

BOX-MARK can be used to open, move, delete, and duplicate columns. To mark a column using **BOX-MARK**:

1. Position the cursor at the top left-hand corner of the column, then use **BOX-MARK**.
2. Use **↓** to move the cursor to the last line in the column.
3. Use **→** to move the cursor across the width of the column.

To open the column, use **INSERT**. To move the column, use **PICK-UP**, position the cursor at the location where the column is to be moved, and use **PUT-DOWN**. To copy the column, use **PICK-COPY**, position the cursor at the location where the copy is to be placed, and use **PUT-DOWN**.

Practice using **BOX-MARK** to copy, move, and delete columns:

1. Use **-LINE**.
2. Position the cursor on line 41 and complete the list, adding titles and extensions:

```

1      t      t      t      t      t      t      t      t      t      r
next few days to discuss all this? Pat's ideas on the campaign seemed a
bit odd to me.

We should meet with the following people before the end of the week:

D. Adams      ext 25      Director
L. Larson     ext 17      Graphic Artist
L. Pantages   ext 21      Consultant
P. Sternig    ext 14      Marketing Manager
J. Winkler    ext 19      Consultant

          Preliminary comments from Larry:

          After a brief review of the accounts, it
          seems to me Pat's insight and recommendations
          are all valid. I'll begin working up some
          new copy for the campaign.

/usr/larry/report                INSERT   Line   45 ( 53) ^

```

3. Position the cursor on the e in ext 25, then use **BOX-MARK**.
4. Use **↓** to move the cursor to the last line in the list.
5. Use **→** to move the cursor across to the C in Consultant and use **PICK-COPY**.
6. Copy the column by positioning the cursor at the D in Director and using **PUT-DOWN**:

```

1   t   t   t   t   t   t   t   t   t   t   r
next few days to discuss all this? Pat's ideas on the campaign seemed a
bit odd to me.

We should meet with the following people before the end of the week:

D. Adams      ext 25          ext 25          Director
L. Larson     ext 17          ext 17          Graphic Artist
L. Pantages   ext 21          ext 21          Consultant
P. Sternig    ext 14          ext 14          Marketing Manager
J. Winkler    ext 19          ext 19          Consultant

                Preliminary comments from Larry:

                After a brief review of the accounts, it
                seems to me Pat's insight and recommendations
                are all valid. I'll begin working up some
                new copy for the campaign.

We should meet with the following people before the end of the week:

/usr/larry/report                INSERT   Line   41 (   53) ^

```

Note that when you define and **PUT-DOWN** a rectangular area of text in this way, text in the cursor column and to the right is displaced to the right. Follow the same procedure using **PICK-UP** and **PUT-DOWN** to move the column, or **DELETE** to remove the column. It is also possible to **INSERT** a blank area within text by using the same procedure.

7. Remove the second column by positioning the cursor at the e in the first occurrence of ext 25.
8. Use **BOX-MARK** **↓** to move the cursor to the last line in the list.
9. Use **→** to move the cursor across to the e in the second occurrence of ext 19, then use **DELETE**. The column is deleted.

4.17 Moving to a Specific Line in a File

To move directly to a specific line in a file:

1. Use **ENTER**.
2. Type the number of the desired line.
3. Use **GO-TO**.

Practice moving to a specific line in the file. When you feel comfortable with the procedure, use **ENTER** **GO-TO** to return to the end of the file.

4.18 Searching for Specific Text

The fastest way to locate specific text throughout a document is by searching. To initiate a search for occurrences of a character string, word, or group of words, search forward with **+SEARCH** or backward with **-SEARCH**.

To search for other occurrences of a specific word showing in the window:

1. Position the cursor at the beginning of the word.
2. Use **ENTER** **+SEARCH** to search from the cursor position to the end of the file. Use **ENTER** **-SEARCH** to search from the cursor position to the beginning of the file.

The system only searches for one word, the word at the cursor position. A word includes any punctuation within or immediately following the word at the cursor position. Thus, if the cursor is positioned on a word followed by a punctuation mark and a search is initiated, the system only searches for occurrences of the word that include the same punctuation.

To search for a particular word or words at which the cursor is not yet positioned:

1. Use **ENTER**.
2. Type the word or words.
3. Use **+SEARCH** or **-SEARCH**.

Avoid unnecessary typing by searching for the shortest possible unique string within the word or string you wish to locate. For example, type **SPE** to search for **SPECIAL** or **The sing** to search for **The single** when the word **single** appears often in the file. This is an efficient way to reach the next revision point.

During a search, the searched-for word or string is stored in the search buffer. Thereafter, search for the next occurrence of the word or string by using **+SEARCH** or **-SEARCH**, without **ENTER**. When no further occurrences of the word or string are found, the system responds **Search failed on string "string"**.

The system searches for an exact character match. For example, a search for the word `office` finds `offices`, but not `Office` (because of the capital O). To find both occurrences of the word, search for `ffice`. Searches are done within single lines; words or strings that split across lines are not found. For example, a search for `home directory` will not find occurrences of that string where a line ends with `home` and the next line begins with `directory`.

Practice searching for specific text using `+SEARCH` and `--SEARCH`:

1. `GO-TO` the beginning of the file.
2. Use `ENTER` `Condos` `+SEARCH` to search forward for the word `Condos`.
3. Continue using `+SEARCH`. When all occurrences of a string have been found, the system responds with the message `Search failed on string "Condos"`. Use `CANCEL` to remove the message.
4. `GO-TO` the beginning of the file and search for the word `condos` by using `ENTER` `condos` `+SEARCH`. Note that for search purposes, `Condos` and `condos` are two different words. A search for one does not locate the other.
5. `GO-TO` the beginning of the file, then look for an occurrence of the string `ondos`. Note that the system finds the string in both `Condos` and `condos`. It is not necessary to type complete words; only the portion that makes the character string unique is necessary.

4.19 Performing Individual Search and Replace

At times you may want to replace many, but not all, instances of a word or words with another word or words. This is possible by putting the replacement string in the replace buffer. This buffer is associated with `REPLACE`.

To replace a string with another string:

1. Search for the word or string to be replaced by using `ENTER` *string* `+SEARCH`. The word or string is put in the search buffer and the search is initiated.
2. When an occurrence of the string to be replaced is located by the system, use `ENTER` *replacement string* `REPLACE`. The

replacement string is put in the replace buffer and the replacement occurs.

3. Use **+SEARCH** to continue searching for the string to be replaced. Use **REPLACE** when you wish to substitute the replacement string.

Practice using **REPLACE** by replacing **Condos** with **condos**:

1. **GO-TO** the beginning of your file.
2. Use **ENTER** **Condos** **+SEARCH**.
3. Use **ENTER** **condos** **REPLACE**.

4.20 Accessing Two or More Files

Occasionally it is helpful to view two or more files simultaneously. When moving or copying text from one file to another the ability to view both files at once eliminates the need to type the same material twice. At other times it is necessary to refer to text in another file for ideas and guidance.

There are two ways to access multiple files. The first permits you to jump between two files, viewing each file full-display, and the second permits you to split the editing window into smaller windows for simultaneous viewing of multiple files.

Any two files can be used as alternate files. You can also use this feature to view two portions of the same file. If the alternate files are the same file, a change made to one copy is also made to the other. Multiple alternate files can be accessed in a single editing session.

4.20.1 Viewing an Alternate File

To create or view an alternate file:

1. Access the first file.
2. Use **ENTER**, then type the name of the second file.
3. Use **USE** to bring the second file onto the screen. If the second file does not exist, or if you mistyped the name of an existing file, the file creation menu will appear on the screen. Either select one of the options to create a new file, or select the **Re-enter the file name** option, type the correct file name into the popup box that displays, and use **EXECUTE**. The second file will appear on the screen.

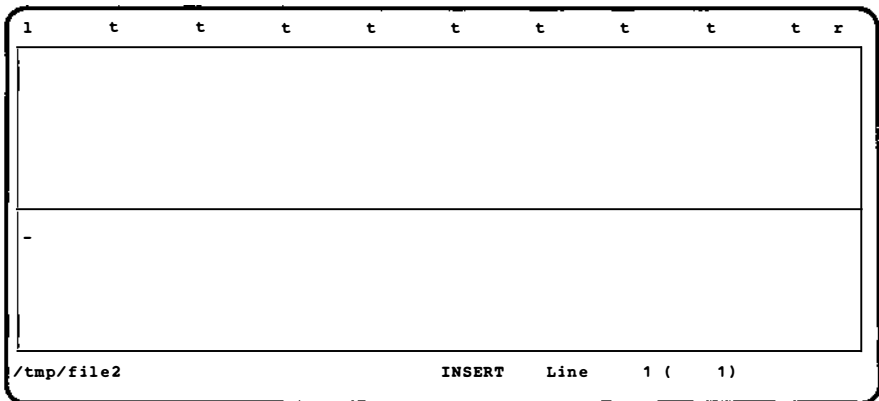
- 4. Use **USE** to alternate between the two files. The name and line number of the displayed file are printed at the bottom of the screen.

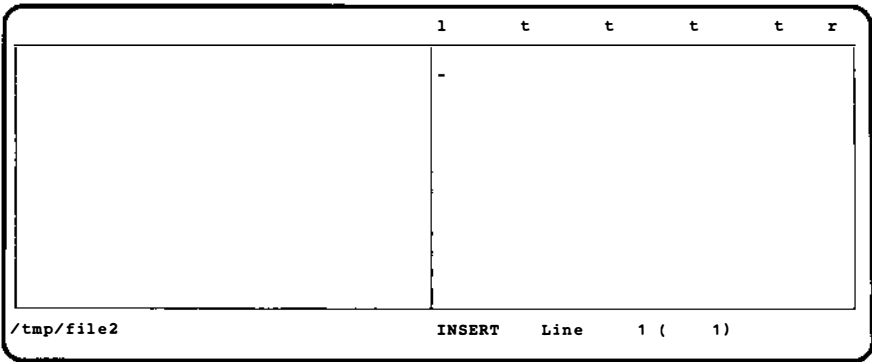
4.20.2 Creating Multiple Windows

Two (or more) files or several different sections of the same file may be in constant view by dividing the window into smaller, multiple windows.

To create a window, the cursor must be positioned at the division point. To create a horizontal window, position the cursor on the line where the window is to be divided. To create a vertical window, position the cursor on the first line of the window at the desired division point.

The following diagrams show the cursor positions for creating vertical and horizontal windows. These particular cursor positions are only examples; you may create windows of any size you choose.





To create a window:

1. Position the cursor at the appropriate location, use **ENTER**, then type a file name.
2. Use **WINDOW**. If the file is an existing file, that file appears in the window. If the file is the file you are currently editing, that file appears in the window and you may edit or view another section of it. If you type the name of a file you wish to create, the instruction box for creating new files appears.
3. Use **EXECUTE** to create the file if it does not yet exist.
4. Use **NEXT-WINDOW** to change windows. If there are more than two windows, **NEXT-WINDOW** moves the cursor from window to window in the order in which the windows were created.
5. Use **ENTER WINDOW** to remove windows. All windows except the one in which the cursor currently resides are removed.

Use this exercise to practice making windows and using alternate files:

1. In the `report` file, use **ENTER** `report` **USE**. The `report` file is now the current and alternate file.
2. Use **ENTER GO-TO** to move to the end of the current file.
3. Use **USE** to switch back and forth between the current and alternate files.

4. In either the current or alternate file, position the cursor part way down the window, then use **ENTER** report **WINDOW** to create a horizontal window containing the report file.
5. Use **NEXT-WINDOW** to switch back and forth between windows.
6. Use **+PAGE**, then use **NEXT-WINDOW**. Note that different parts of the file are displayed in each window.
7. **ZOOM-OUT** of the files. Although this exercise used the file **report** as the alternate and “window” files, you can use **USE**, **WINDOW**, and **NEXT-WINDOW** to access many different files at one time.

5. THE FILE MANAGER

The File Manager is the primary utility used to create, access, and delete files and directories. The directory you entered at login is called your home directory (`$HOME`); the screen used to display a directory is called the File Manager display. The File Manager display should be visible on your screen now. If you are in a file, **ZOOM-OUT** to return to the File Manager display of your home directory:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project

/usr/larry INSERT Line 1 (5)

5.1 Copying Files

Sometimes a duplicate of a file that already exists is needed. Instead of retyping the text, you may prefer to copy the original document. The File Manager allows you to copy a file simply and easily using **PICK-COPY** and **PUT-DOWN**. You can also copy multiple files using **BOX-MARK** or **ENTER** motion with **PICK-COPY** and **PUT-DOWN**.

1. Move the cursor to the line describing your **report** file and use **PICK-COPY**. The cursor will automatically move to the next line.
2. Use **PUT-DOWN**; a popup box will appear:

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project
report	First Report on the QuikSell Project

File name "report" already exists.
Press CANCEL to abort file restore,
or enter a new name:

/usr/larry INSERT Line 6 (6)

3. Use **HELP** for additional information about the messages in popup boxes on your screen:

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project
report	First Report on the QuikSell Project

You are trying to restore a file into a directory that already contains a file by that name. You can type in a different name or type CANCEL to abort the restore.

/usr/larry INSERT Line 6 (6)

4. Use **CANCEL** to remove the **HELP** popup box. Since file names must be unique, type a new file name:

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project
report	First Report on the QuikSell Project

File name 'report' already exists.
Press CANCEL to abort file restore,
or enter a new name: test

/usr/larry INSERT Line 6 (6)

5. Use **EXECUTE**. The copy is renamed as specified:

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project
test	First Report on the QuikSell Project

/usr/larry INSERT Line 6 (6)

6. Change the descriptive information, using **ENTER** **DELETE-CHARACTER** to remove extra characters:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project
test	Test File
/usr/larry	
INSERT Line 6 (6)	

7. You now have two copies of your `report` file, one called `report` and one called `test`. **ZOOM-IN** to your new `test` file if you wish to verify that it contains the same information as your `report` file.

5.2 Renaming Files

You can rename a file by typing over the current file name:

1. Switch to overwrite mode by using **INSERT-MODE**, then change the name of the `test` file to `copy` by typing over the name.
2. Use **INSERT-MODE** to switch back to insert mode.

5.3 Deleting Files

There are two ways to delete a file, **PICK-UP** and **DELETE**. **PICK-UP** and **DELETE** can also be used with **BOX-MARK** or **ENTER** motion to delete multiple files.

PICK-UP deletes a file and places it in the pick buffer. **PUT-DOWN** removes the file from the pick buffer and replaces it in your directory. You will use **PICK-UP** to move a file from one directory to another in §5.7. It is recommended that you use **PICK-UP** when you wish to move a file from one location to another.

DELETE removes a file and places it in the wastebasket buffer. **RESTORE** removes the file from the wastebasket buffer and replaces it in your directory. It is recommended, however, that you use **DELETE** to permanently remove a file.

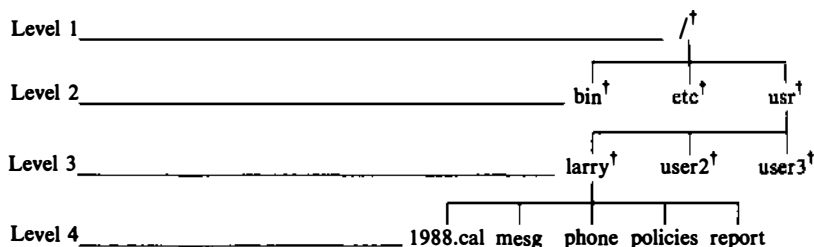
1. Position the cursor on the line describing the copy file.
2. Use **DELETE** to remove the copy file from your directory:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project
/usr/larry	
INSERT Line 6 (5)	

5.4 Path Names and Directory Structure

The system identifies you by your user name. Similarly, the system identifies your files by the file names you assign and stores them in a directory structure (filing system) that you arrange. The name of the directory where you are currently located is indicated at the bottom left of the File Manager display. In our example the name of the current directory is /usr/larry. The name /usr/larry is also the “path name” of this directory. A path name consists of a / followed by one or more directory names separated by /s. The / that begins the path name is the first directory in the directory structure—the “root directory.” The last directory name in the series is the current directory—in this case, your home directory, larry. A path name is the sequence of directories from the root directory to the file or directory you wish to reference.

The diagram below illustrates how a directory structure is configured (each directory is indicated by a †). Only the files in the directory /usr/larry are shown:



The diagram illustrates how the path name of a directory identifies that directory's placement in the directory structure. As you create and delete files and directories, your directory structure will change to reflect the additions and deletions that you make. You can **ZOOM-IN** and **ZOOM-OUT** to move around the directory structure:

1. **ZOOM-OUT** to the directory above your home directory:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                Description
-----
larry               Larry's Home Directory
user2               User2's Home Directory
user3               User3's Home Directory

/usr                INSERT      Line      1 (      3)
  
```

2. Practice using **ZOOM-IN** and **ZOOM-OUT** to explore the directory structure. If you get lost, you can always return to your home directory via the Show home directory **MENU** option discussed in §2.3.

5.5 Creating Directories

So far you have worked on files in your home directory. Now you will create a subdirectory to contain files for a particular project:

1. Position the cursor on the line describing the file `report`, then use **INSERT**. Type a directory name and some descriptive information:

```

                INTERACTIVE TEN/PLUS FILE MANAGER
      Move the cursor to an item below and ZOOM-IN to see it.

      File                                   Description

1988.cal   Calendar
mesg       My Incoming Messages
phone     Company Telephone Book
policies   Company Policies
quiksell   QuikSell Project Directory
report     First Report on the QuikSell Project

/usr/larry                               INSERT   Line   5 (   6)

```

2. **ZOOM-IN** on the line describing the new directory; a menu will appear:

```

                INTERACTIVE TEN/PLUS FILE MANAGER
      Move the cursor to an item below and ZOOM-IN to see it.

      File                                   Description

1988.cal   Calendar
mesg       My Incoming Messages
phone     Company Telephone Book
policies   Company Policies
quiksell   QuikSell Project Directory
report     First Report on the QuikSell Project

You are attempting to create file "/usr/larry/quiksell"

Select a menu option (for example, move the cursor to an item and EXECUTE); otherwise, CANCEL to remove the menu, or HELP to display help information.

Create an ASCII file (without history)
Create a structured file (with history)
Create a directory
Re-enter the file name

/usr/larry                               INSERT   Line   5 (   6)

```

3. Position the cursor on Create a directory and **EXECUTE**. A blank File Manager display for the quiksell directory will appear on the screen.
4. **ZOOM-OUT** to your home directory.

5.6 Moving Files Between Directories

You can move a file from one directory to another with **PICK-UP** and **PUT-DOWN**. Move the report file from your home directory to your new quiksell directory:

1. With the cursor on the line describing the `report` file, use **PICK-UP**.
2. Move the cursor to the line describing the `quiksell` directory, then **ZOOM-IN**.
3. Use **PUT-DOWN** to complete the move.

You can also move multiple files between directories by using **BOX-MARK** or **ENTER** motion with **PICK-UP** and **PUT-DOWN**.

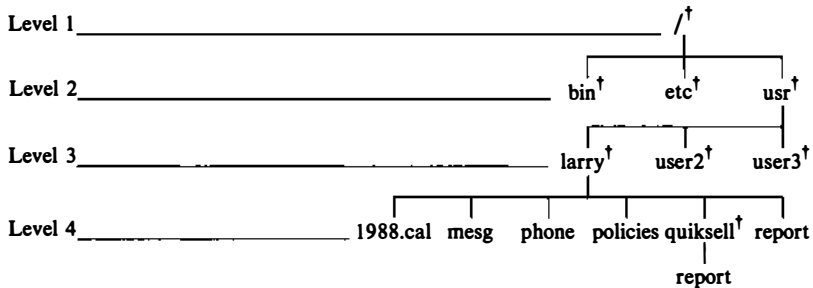
5.7 Copying Files Between Directories

Use **PICK-COPY** and **PUT-DOWN** to make a copy of the `report` file and place it in your home directory:

1. With the cursor on the line describing the `report` file, use **PICK-COPY**.
2. **ZOOM-OUT** to your home directory and use **PUT-DOWN**. You now have a copy of the `report` file in both your home directory and your `quiksell` directory.

You can also copy multiple files between directories by using **BOX-MARK** or **ENTER** motion with **PICK-COPY** and **PUT-DOWN**.

Note that the system does not ask you to type a new file name as it did when you tried to create a copy of the `report` file in the same directory as the original. The system allows you to have files of the same name in different directories because the path name of each remains unique. In this case, the first path name is `/usr/larry/report` and the second path name is `/usr/larry/quiksell/report`:



5.8 Alternate Methods of Changing Directories

So far, you have used **ZOOM-IN** and **ZOOM-OUT** to change directories. There are, however, alternate methods for doing this.

5.8.1 Full and Relative Path Names

You can move from one directory to another with **ENTER** *path-name* **USE**. A path name that does not begin with a / is called a “relative” path name and tells the system to search downward from your current directory. You can give the system a relative path name if the target directory is on a direct path below your current directory.

For example, the `quiksell` directory is in your home directory and is therefore on a path below your home directory:

1. In your home directory, use **ENTER** and type the directory name `quiksell`.
2. Use **USE**; the File Manager display for the `quiksell` directory will appear on your screen.

In this example, you have given the path name of the target directory relative to your location within the directory structure; that is, a path name that does not begin with a /.

A full path name begins with a / and tells the system to begin its search from the root directory. To access a directory above your current directory, use the full path name to the target directory. For example, to move from the `quiksell` directory to the home directory of `user2` in the above diagram, you should provide the system with the full path name to `user2`'s home directory. This is necessary because `user2`'s home directory is located above your current location, the `quiksell` directory. In our example, the full path name to `user2`'s home directory is `/usr/user2`. To move to that directory, use **ENTER** `/usr/user2` **USE**.

5.8.2 Parent Directories

A “parent” directory is the directory located immediately above your current directory. You can use **ENTER** `..` **USE** to change from a directory to its parent directory. **ENTER** `..` **USE** moves you one level up in the directory structure. For example, you can move from the `quiksell` directory to its parent directory, `larry`, with the sequence **ENTER** `..` **USE**. The two dots take the place of the parent directory name. Practice the sequences

ENTER *directoryname* **USE** and **ENTER** . . **USE** to move back and forth between the `quiksell` directory and its parent directory. Because the `quiksell` directory and its parent directory are now the current and alternate files, respectively, you can move between them with **USE**.

Note that you can create a new directory or file with **ENTER** *filename* **USE** by providing the system with a name for a new directory or file. The system will display a popup box asking you to choose the type of file you wish to create. Select the appropriate option to create the directory or file. Use **CANCEL** to remove the popup box without creating a directory or file.

5.9 Removing Directories

You can remove a directory just as you remove a file, with either **PICK-UP** or **DELETE**. It is suggested that you use **PICK-UP** and **PUT-DOWN** to move a directory from one location to another and **DELETE** to remove a directory. When you remove a directory, all files and subdirectories located within that directory are also removed:

1. With the cursor on the line describing your `quiksell` directory, use **DELETE**. The message Saving directory hierarchy "quiksell" will appear. Both the `quiksell` directory and the `report` file contained within it are removed.

You can use **RESTORE** to recover a directory deleted the last time you used **DELETE** but only during the same editing session in which the directory was removed. Once you have exited the TEN/PLUS environment, the buffers that hold deleted files and directories are cleared.

2. Use **RESTORE** to restore the deleted `quiksell` directory. Both the `quiksell` directory and the `report` file within it are restored.

PICK-UP and **DELETE** can also be used with **BOX-MARK** and **ENTER** *motion* to remove multiple directories.

5.10 Recovering Directories From `.putdir`

The TEN/PLUS File Manager maintains a safety back-up area where deleted files and directories are stored. This safety area allows you to recover files and directories that you have mistakenly deleted. The default location is a directory called `.putdir` in

your home directory. Files and directories placed in `.putdir` by the system remain there until you specifically remove them. Files and directories placed in `.putdir` are removed whenever you select Housekeep from your New Task Menu.

1. With the cursor on the line describing the `quiksell` directory, use **DELETE**.

A copy of the `quiksell` directory and all files and sub-directories contained within it is now located in `/usr/larry/.putdir`. You have not seen `.putdir` listed in your home directory because files and directories whose names begin with a `.` are not usually visible in the directory display; they are hidden by the system. **LOCAL-MENU** provides a way to list such hidden files.

2. Use **LOCAL-MENU**:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                Description
-----
1988.cal                            Calendar
mesg                                 My Incoming Messages
phone                                Company Telephone Book
P
x
File Manager
Select a menu option (for example, move the
cursor to an item and EXECUTE); otherwise,
CANCEL to remove the menu, or HELP to display
help information.

(1)   Display "visible" files
(2)   Display all files
---   Return to normal directory display
(4)   Show details about files
(5)   Show more details about this file
---   Show more details about this file
ct

/usr/larry                            INSERT   Line   5 (   5)

```

3. Select (2) Display all files.
4. **GO-TO** the top of the directory listing:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
.*	
..	
....pr6ceb962e	
....scc005b79f	
.estate	
.index	
.llog	
.profile	
.putdir	
.scomb	
1988.cal	Calendar
mesg	My Incoming Messages
phone	Company Telephone Book
policies	Company Policies
report	First Report on the QuikSell Project
/usr/larry	INSERT Line 16 (16)

The directory listing now includes the names of a number of “hidden” files, most of which are created and used internally by the system. The hidden files that will be discussed in this tutorial are .putdir, .bak, .old, .index, and .* files. See §5.13 for more information about these hidden files.

- With the cursor on the line containing the name .putdir, **ZOOM-IN**. The File Manager display for the .putdir directory will appear on the screen:

INTERACTIVE TEN/PLUS FILE MANAGER	
Move the cursor to an item below and ZOOM-IN to see it.	
File	Description
0.copy	
0.quiksell	
/usr/larry/.putdir	INSERT Line 1 (2)

Files and directories are saved in .putdir under their original file names prefixed with a number. The system uses such a prefix in order to avoid duplicate file names in .putdir.

The `quiksell` directory, for example, is saved in `.putdir` under the name `0.quiksell`. The descriptive text from the `Description` field is not saved.

6. To recover the `quiksell` directory, move the cursor to the line containing the directory name and use **PICK-COPY**. Do not use **PICK-UP** in the `.putdir` directory unless you wish to completely remove the file or directory from the system. Files and directories removed from `.putdir` with **PICK-UP** or **DELETE** cannot be **PUT-DOWN** again.
7. **ZOOM-OUT** to your home directory, then use **PUT-DOWN**. The `quiksell` directory is restored. Remove the number prefix from the directory name, if you wish, and replace the descriptive text. Note that your home directory listing still shows the names of your hidden files. You can remove hidden file names from the listing by using **LOCAL-MENU** and selecting (1) Display "visible" files.

Since all files and directories deleted from the File Manager display are placed in your `.putdir` directory, it is a good practice to periodically clean up that directory with **PICK-UP** or `Housekeep`. This will free disk space for your current files. Files and directories removed from `.putdir` with **PICK-UP** or `Housekeep` are permanently removed from the system. On some systems, the system administrator may have `.putdir` removed periodically.

5.11 File and Directory Permissions

Each TEN/PLUS file or directory has certain default permissions or protections that specify who will have access to it. The owner can change the permissions on a file or directory at any time. There are three different types of permissions: "read," "write," and "execute."

The combination of read and execute permissions on a directory allows you to **ZOOM-IN** and view the listing of files and directories stored there. Read permission on a file allows you to view and copy the contents of that file. The defaults allow read permission on files and directories to all users.

Write permission on a directory allows you to create and delete files in that directory. Write permission on a file allows you to modify that file. The defaults allow write permission on files and directories only to the owner.

The combination of execute and read permissions on a directory allows you to **ZOOM-IN** to that directory and view the listing of files and directories stored there. Execute permission on a directory allows you to access a file in that directory, although without read permission you cannot see the listing of the directory on your screen. If a file is an executable program, execute permission allows you to run that program. The defaults allow execute permission on directories to all users and execute permission on files to no one.

Each set of permissions applies to one of three categories of users: "owner," "group," and "others."

The owner is the person who creates the file or directory. A group often consists of people working on the same project or in the same department. You are usually assigned to a group when you are added as a user to the system. Others refers to everyone on the system who is neither the owner nor in the owner's group.

5.11.1 Changing Permissions on Files and Directories

You can change the permissions on your files and directories through the **LOCAL-MENU** for the File Manager:

1. Use **LOCAL-MENU**:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                     Description
-----
1988.cal                               Calendar
msg                                     My Incoming Messages
phone                                  Company Telephone Book
p
q
r
File Manager
Select a menu option (for example, move the
cursor to an item and EXECUTE); otherwise,
CANCEL to remove the menu, or HELP to display
help information.

(1)   Display "visible" files
(2)   Display all files
---   Return to normal directory display
(4)   Show details about files
(5)   Show more details about this file
---   Show more details about this file
ct

/usr/larry                               INSERT   Line   5 (   6)

```

A dashed line in front of an option indicates that the option is not available in this situation. The third and sixth options

have dashed lines in front of them and are, therefore, not available in this situation. They will become available when you look at details about a particular file or directory as described in step 3 below. You have already used options (1) and (2) to display files in your directory listing. The other options are described below:

--- Return to normal directory display

Returns you to the directory display from either the file status information display or the Detailed File Status Information display.

(4) Show details about files

Displays the file status information display for an entire directory.

(5) Show more details about this file

Can be used to view the Detailed File Status Information display for a particular file either from the directory listing or from the the file status information display.

--- Show more details about this file

Provides an alternate way to move to the Detailed File Status Information display for a particular file when you are viewing the file status information display.

2. Select option (4) Show details about files. A display showing details about the files and directories in your directory listing will appear:

INTERACTIVE TEN/PLUS System						
To see detailed file status information, select an item and ZOOM-IN.						
File Name	T	Owner	Size	Modification Date		Permissions
1988.cal	r	larry	18	Mar 4 1988	16:46:08	rw- r-- r--
mesg	r	larry	18	Mar 4 1988	17:05:38	rw- r-- r--
phone	r	larry	10	Mar 5 1988	10:49:42	rw- r-- r--
policies	r	larry	25	Mar 5 1988	12:48:51	rw- r-- r--
quiksell	d	larry	80	Mar 11 1988	12:04:34	rx- r-x r-x
report	r	larry	2488	Mar 11 1988	10:25:32	rw- r-- r--

`/usr/larry` INSERT Line 6 (6)

This is a file status information display. The fields in the display contain this information:

File Name

Indicates the name of the file or directory.

T

Indicates the type of the entry; an `r` indicates a regular file and a `d` indicates a directory.

Owner

Indicates the owner of the file or directory.

Size

Indicates the size of the file or directory in bytes.

Modification Date

Indicates the date and time the file or directory was last modified.

Permissions

Indicates the permissions for owner, group, and others. An `r` indicates read permission, a `w` indicates write permission, and an `x` indicates execute permission.

You can change the permissions on a file or directory by typing over the existing permissions at this level, or you can use `LOCAL-MENU` to move to a more detailed level, the Detailed File Status Information display.

- With the cursor on the line describing the `report` file, use `LOCAL-MENU`:

5. With the cursor in the field labeled Permissions, use **INSERT-MODE** to enter overwrite mode. Change the permissions so that users in your group have write permission on the file. Do this by typing a w over the first - in the Group field:

```

                Detailed File Status Information
Description    Full path name: /usr/larry/report
               File Size in Bytes:  2488 427   File Size in Blocks: 5
               File Type: Ordinary           Number of Links: 1
               Inode: 2053   Directory ID: 2, 10   Device ID: N. A.

Ownership     Owner of the file: larry           Uid: 229
               Group of the file: staff       Gid: 1

Permissions   Owner: rw-           Group: rw-           Other: r--
               Set Uid: -           Set Gid: -           Save text image: -

File          Time of last file access:           Mar 11 1988 10:25:32
Access       Time of last data modification:      Mar 11 1988 10:25:32
Times        Time of last file status change:     Mar 11 1988 10:25:32

/usr/larry                                OVERWRITE   Line    1 (  1)

```

6. Use **LOCAL-MENU**:

Detailed File Status Information

Description	Full path name: /usr/larry/report	
	File Size in Bytes: 2488 427	File Size in Blocks: 5
	File Type: Ordinary	Number of Links: 1
	Inode: 2053	Directory ID: 2, 10 Device ID: N. A.

Ownership	Owner of t	<p style="text-align: center;">File Manager</p> <p>Select a menu option (for example, move the cursor to an item and EXECUTE); otherwise, CANCEL to remove the menu, or HELP to display help information.</p>	
	Group of t		
Permissions	Owner: rw-		
	Set Uid: -		
File Access Times	Time of la		
	Time of la		
	Time of la		
	---		Display "visible" files
	---		Display all files
	(3)		Return to normal directory display
(4)	Show details about files		
---	Show more details about this file		
(ZOOMOUT)	Show details about files		
---	Show more details about this file		

/usr/larry OVERWRITE Line 1 (1)

7. Select (4) Show details about files or (ZOOMOUT) Show details about files. You will again be viewing the file status information display.

To deny access to all of the files in a particular directory, you can change the permissions on the directory itself. For example, if users in your group do not have read permission on your quiksell directory, they cannot view the files in that directory even if they have read permission on a particular file.

8. Change the permissions on your quiksell directory so that other users do not have read or execute permission on it:

INTERACTIVE TEN/PLUS System

To see detailed file status information, select an item and ZOOM-IN.

File Name	T	Owner	Size	Modification Date	Permissions
1988.cal	r	larry	18	Mar 4 1988 16:46:08	rw- r-- r--
mesg	r	larry	18	Mar 4 1988 17:05:38	rw- r-- r--
phone	r	larry	10	Mar 5 1988 10:49:42	rw- r-- r--
policies	r	larry	25	Mar 5 1988 12:48:51	rw- r-- r--
quiksell	d	larry	80	Mar 11 1988 12:04:34	rwX --- ---
report	r	larry	2488	Mar 11 1988 10:25:32	rw- rw- r--

/usr/larry OVERWRITE Line 6 (6)

You can bypass **LOCAL-MENU** once you have become familiar with the numbers that correspond to the choices on that menu. **(3)** corresponds to the File Manager **LOCAL-MENU** option (3) Return to normal directory display, therefore, you can use **(3)** to return to the File Manager display.

9. Use **LOCAL-MENU** option (3) Return to normal directory display to return to the File Manager display.

To change permissions or other options on a single file, you can use **LOCAL-MENU** option (5) Show more details about this file from the normal directory display to see the Detailed File Status Information display. In large directories, this is much faster than displaying status information for all files.

5.12 New Task Menu Options

You have already used **MENU** to move from another directory to your home directory. This option is always available to you, no matter where you are located in the directory structure. There are several other options also available to you through **MENU**:

1. Use **MENU** to see the New Task Menu:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                Description
-----
1988.cal            Calendar

```

```

New Task Menu

Select a menu option (for example, move the
cursor to an item and EXECUTE); otherwise,
CANCEL to remove the menu, or HELP to display
help information.

Show home directory
Execute UNIX shell commands
Run a shell command in a box
Show your profiles directory
Edit your editor profile
Housekeep
Display history of current file

```

```

/usr/larry                INSERT        Line        1 (        6)

```

The options **Execute UNIX shell commands** and **Run a shell command in a box** allow you to communicate with your system. The options **Show your profiles directory** and **Edit your editor profile** allow you to customize your profiles. The option **Housekeep** provides an easy way to remove history from structured files and remove unwanted files from your directories. The option **Show history of current file** allows you to view the history of the file that is currently displayed (if the current file is structured). These options are discussed in the following paragraphs.

5.12.1 Executing a Command From the Editor Subshell

The TEN/PLUS system eliminates most of the requirements for communicating directly with the system by allowing you to perform tasks with functions instead of commands. However, there are many available system commands in the TEN/PLUS environment. **MENU** provides two methods of communicating with the system. The first one allows you to run a command from the editor subshell:

1. Select the option **Execute UNIX shell commands**. The display will clear and the editor subshell prompt will appear near the top left of your screen:

```

Execute UNIX shell commands
Touch CNTL-D to continue editing
eshell>

```

Line 4

You are now at the editor subshell level, and you can type commands to the system. The *date(1)* command causes the system to display the current date and time.

2. Type *date*, then press the **ENTER** key:

```

Execute UNIX shell commands

Touch CNTL-D to continue editing
eshell> date
Fri Mar 11 14:41:07 PST 1988
eshell>

```

Line 6

- Use **CTRL** **d**, then press the **ENTER** key, to return to the File Manager display.

5.12.2 Executing a Command in a Popup Box

The second method of communicating with the system is to execute a command in a popup box:

- Use **MENU**. Select the option Run a shell command in a box; a popup box will appear:

```

                INTERACTIVE TEN/PLUS FILE MANAGER
                Move the cursor to an item below and ZOOM-IN to see it.

```

File	Description
1988.cal	Calendar
Shell command:	
quiksell report	QuikSell Project Directory First Report on the QuikSell Project

```

/usr/larry                INSERT    Line    1 (    6)

```

- Type **date**, then use **EXECUTE**:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                     Description
-----
1988.cal                               Calendar
sages
Fri Mar 11 14:45:34 PST 1988          ne Book
S
quiksell                               QuikSell Project Directory
report                                First Report on the QuikSell Project

/usr/larry                               INSERT   Line   1 (   6)

```

The output for the command is displayed in the popup box. If there is no specific output for the command (for example, if the *cp*(1) command is used to copy a file), a popup box with the message No output from "Run a shell command in a box" appears.

3. Use **CANCEL** to remove the popup box.

Note, however, that interactive commands (commands requiring additional input from the user) cannot be run in popup boxes. If you accidentally enter an interactive command, try using **BREAK** to stop the command from executing. The *date* command is just one example of a command that can be executed from the editor sub-shell or in a popup box. For a complete listing of system commands, see your user's manual.

5.12.3 Showing Your Profiles Directory

The TEN/PLUS User Interface includes three profiles that you can use to customize the TEN/PLUS environment to suit your particular needs. Custom versions of the TEN/PLUS profiles must be stored in a `profiles` directory in your home directory. The `Show your profiles directory` option moves you from a file or another directory to your `profiles` directory. If your home directory does not already contain a `profiles` directory, it will be created.

5.12.4 Editing Your Editor Profile

The editor profile, `editorprf`, is a standard profile intended for beginning users. The TEN/PLUS `editorprf` file lets you specify what the New Task Menu will look like, what the Help Menu will look like, which files the editor should watch, and what the

editor search paths should be. Use the **Edit your editor profile** option to customize your `editorprf` to suit your needs. If this file does not already exist, it will be created. If a `profiles` directory does not already exist in your home directory, it will be created as well. The top level of the `editorprf` file looks like this:

```

                                Editor Profile File

This file allows you to modify the behavior of the editor to suit your
preferences. Put the cursor on the line of interest and ZOOM-IN to specify
your choice of options.

    MENU Options
    HELP Options
    Files the Editor Should Watch
    Editor Search Paths

/usr/larry/profiles/editorprf      INSERT   Line   1 (   4)

```

Your system may have additional profile items. Refer to §6 for a brief discussion of how to modify the **MENU Options** section of your `editorprf` file; for more information about this and the other sections of your `editorprf` file, refer to “TEN/PLUS Profiles.”

5.12.5 Housekeep

The **Housekeep** option on your **New Task Menu** runs a `housekeep` command that removes `$HOME/.putdir`, removes history from all structured files in your directories, and removes all `...` and `.bak` files from your directories. (Note that you cannot restore files from `.putdir` if they were deleted prior to using this option.)

5.12.6 Displaying the History of the Current File

The **Display history of current file** option on your **New Task Menu** displays a list of previous versions of the current file if the current file is structured. This list can be used to view or restore previous versions (see §6).

5.13 The File Manager Profile

The File Manager has its own profile file that allows you to customize your directory listing. This profile file is called `indexprf`. This display shows the `indexprf` file with the system defaults:

```

Directory Helper Options

Synchronize the index file with the directory: x

Directory for deleted files: $HOME/.putdir

Files
to be
hidden
    *.bak
    *.old
    *.index
    .*

/usr/larry/profiles/indexprf          INSERT   Line      1 (      1)

```

If you wish to change any of the defaults shown in the sample `indexprf` file above, you must create an `indexprf` file in your `$HOME/profiles` directory, and edit the file. If you already have a `profiles` directory, you can access that directory, type `indexprf` in the **File** field, **ZOOM-IN**, and select the option to create a structured file. If you do not have a `profiles` directory, use **ENTER** `$HOME/profiles/indexprf` **USE**, and select the option to create a structured file. The system will create a `profiles` directory, then create an `indexprf` file in that directory and display the new `indexprf` file on your screen.

The **Synchronize** option in this section allows you to specify whether your directory listing will be updated (that is, synchronized with the actual state of the system) whenever you create files through **ENTER** `filename` **USE**. The `x` in this field specifies that synchronization will take place; this is the default. Unless there is an `x` in this field, synchronization will not occur. In that case, files created through **ENTER** `filename` **USE** will not show in the directory listing until you manually list them by selecting the File Manager local menu option (1) **Display "visible" files**.

The **Directory for deleted files** option specifies the directory in which the system places your deleted files and directories. The default is the `.putdir` directory located in your home directory. You can change this default by replacing it with a new directory name. For example, you might decide to place deleted files and directories in a directory called `removed` in your home directory. In this case, you would type the path name `$HOME/removed` over the existing path name in this option. A directory name must be specified for this option or you will be unable to delete files or directories.

The **Files to be hidden** option specifies those files that are to be hidden by the system. By default, all files beginning with a `.` or ending in `.bak`, `.old`, or `.index`, will be hidden. The asterisk you see in the display is used as a wildcard, which means that it is equivalent to any character or characters. Files ending in `.bak` are created by the system each time you complete an editing session on an ASCII file (such as those you have been using). The previous version of that file is not deleted but is renamed by truncating the file name to 10 characters (if necessary) and appending `.bak`. Any previous `.bak` file is overwritten. A `.index` file is a structured file created and used by the File Manager to edit a directory; you cannot access a `.index` file directly. You can specify which files are to be hidden in your directory listing by modifying the list of file names contained in this option. You can, however, always view a listing of all files by selecting the File Manager local menu option (2) **Display all files**.

6. THE HISTORY DISPLAY

The TEN/PLUS system makes use of “helpers” to perform specific computations on data that has been stored with a predefined structure. The structure of data used by a helper is defined by a form associated with the data whenever it is saved or retrieved. The TEN/PLUS History Display helper provides a way to inspect the history of a structured file. You can use this helper to determine details about the changes made to the file, and you can recreate versions of the file as needed. **LOCAL-MENU** for the History Display allows you to copy a previous version to a new file for further editing.

6.1 Types of INed Files

The INed editor can edit two types of files: ASCII and structured. All TEN/PLUS files are either ordinary ASCII files or structured files. An ASCII file is composed of a sequence of ordinary text characters.

A structured file reflects a specific data structure defined by an associated form, and it keeps a record of changes that have been made to the file. This record is called the history of the file. A structured file automatically saves versions of the file between its original creation and the time that it is ultimately deleted. Although you cannot use some system commands directly with structured files, you can convert the files to ASCII if necessary. (See *readfile*(1) for more information about converting structured files to ASCII.) It should be noted that structured files may grow much larger than the corresponding ASCII files over a lifetime of changes to the files. However, it is less time-consuming to save structured files. You can use the History Display only with structured files, therefore you should use structured files whenever you wish to maintain a history of changes made to a file.

6.2 Creating a Structured File

You must first create and modify a structured file to use the History Display:

1. **ZOOM-IN** to the `quiksell` directory, then type a file name and descriptive information:


```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                     Description
-----
report                                  First Report on the QuikSell Project
sample                                  Sample Flyer for QuikSell

/usr/larry/quiksell                      INSERT   Line   2 (   2)

```

2. **ZOOM-IN** on the line containing the new file name:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                     Description
-----
report                                  First Report on the QuikSell Project
sample                                  Sample Flyer for QuikSell

You are attempting to create file "/usr/larry/quiksell/sample"

Select a menu option (for example, move the
cursor to an item and EXECUTE); otherwise,
CANCEL to remove the menu, or HELP to display
help information.

Create an ASCII file (without history)
Create a structured file (with history)
Create a directory
Re-enter the file name

/usr/larry/quiksell                      INSERT   Line   2 (   2)

```

3. Since history is kept only for structured files, select the option to create a structured file. A blank file will appear on your screen; this is considered to be the first version of the file.
4. Type the following text:

```

l   t   t   t   t   t   t   t   t   t   r
-----
QUIKSELL MAKES THE HARD SELL EASY

Software tools for the sales industry
that provide a revolutionary approach
to sales.

      from the
    QuikSell Corporation

/usr/larry/quiksell/sample      INSERT   Line   12 (   12)

```

6.2.1 Creating Versions of a File

When you terminate an editing session, the system saves the modified file. **SAVE** provides a way of saving a file, including all changes, without terminating the editing session. **SAVE** three versions of the sample file:

1. Use **SAVE** to save this version of the file. Now add a new line at the top of the file as shown:

```

l   t   t   t   t   t   t   t   t   t   r
-----
Many Fortune 1000 companies have already discovered that

QUIKSELL MAKES THE HARD SELL EASY

Software tools for the sales industry
that provide a revolutionary approach
to sales.

      from the
    QuikSell Corporation

/usr/larry/quiksell/sample      INSERT   Line   2 (   12)

```

2. Use **SAVE**, then change the text again, as indicated:

```
l   t   t   t   t   t   t   t   t   t   r
Many Fortune 1000 companies have already discovered that

      QUIKSELL MAKES THE HARD SELL EASY

Software tools that make sales management efficient and accurate.

      from the
      QuikSell Corporation

/usr/larry/quiksell/sample          INSERT   Line      8 (    10)
```

3. Use **SAVE** a third time. You now have four different versions of the sample file. (Remember that the original blank file is considered to be a version.)

6.3 Accessing the History of a File

Because you have created several versions of the sample file, you can now view its history:

1. Use **MENU**:

```
l   t   t   t   t   t   t   t   t   t   r
Many Fortune 1000 companies have already discovered that

      QUIKSELL MAKES THE HARD SELL EASY

s
<  New Task Menu
   Select a menu option (for example, move the
   cursor to an item and EXECUTE); otherwise,
   CANCEL to remove the menu, or HELP to display
   help information.
   Show home directory
   Execute UNIX shell commands
   Run a shell command in a box
   Show your profiles directory
   Edit your editor profile
   Housekeep
   Display history of current file
ent and accurate.

/usr/larry/quiksell/sample          INSERT   Line      8 (    10)
```

2. Select the option **Display history of current file**. A form, similar to the one shown below, will list the versions of the file:

```
History of file: /usr/larry/quiksell/sample
```

User	Modification Time	Ins	Del	New
larry	Fri Mar 11 13:33:59 1988	0	0	0
larry	Fri Mar 11 13:35:25 1988	0	0	6
larry	Fri Mar 11 13:37:02 1988	0	0	1
larry	Fri Mar 11 13:37:38 1988	1	3	1

HISTORY-MODE INSERT Line 1 (4)

This is the **History of file** form. The form shows the name of the user who edited the file; the date and time the editing session started; and the number of lines (or records in a structured nontext file) that were inserted, deleted, or changed. To see a particular version of the file, **ZOOM-IN** on the line describing that version.

3. With the cursor on the third line, **ZOOM-IN** to see the third version of the sample file:

```
1 t t t t t t t t t r
```

```
Many Fortune 1000 companies have already discovered that
```

```
QUIKSELL MAKES THE HARD SELL EASY
```

```
Software tools for the sales industry
```

```
that provide a revolutionary approach
```

```
to sales.
```

```
from the
```

```
QuikSell Corporation
```

HISTORY-MODE INSERT Line 1 (12)

6.3.1 The History Display Local Menu

LOCAL-MENU for the History Display provides access to several special options for working with versions of a file. You can use **LOCAL-MENU** while viewing a version of a file to print the time of the version in a popup box, show the next version, show the previous version, redisplay the History of file form, or save the version.

Use **LOCAL-MENU** to see the History Display Options:

```

1      t      t      t      t      t      t      t      t      t      r
-----
History Display
-----
Select a menu option (for example, move the
cursor to an item and EXECUTE); otherwise,
CANCEL to remove the menu, or HELP to display
help information.

(1) show time of this version of the file
(2) show next time
(3) show previous time
(4) redisplay history
(5) save current version of file
-----
HISTORY-MODE                                INSERT   Line   1 ( 12)

```

Option (1) displays the last modification time of the current version in a popup box. You can select this option instead of using **ZOOM-OUT** to view the modification time on the History of file form. Use **CANCEL** to remove the popup box containing the modification time of the current version.

Options (2) and (3) show the next or previous versions of a file, respectively.

Option (4) allows you to move from a version of a file back to the History of file form.

Option (5) allows you to save a previous version of a file. This is necessary if you need to edit a previous version. In order to edit a version, you must save it under another name, or make it the current version:

1. Use **LOCAL-MENU** and select (5) save current version of file:

```

1      t      t      t      t      t      t      t      t      t      t      r
-----
*
Enter file name (/usr/larry/quiksell/sample):
QUIKSELL MAKES THE HARD SELL EASY
Software tools for the sales industry
that provide a revolutionary approach
to sales.

      from the
      QuikSell Corporation

HISTORY-MODE                               INSERT   Line      1 ( 12)

```

2. To save a copy of this version under a new file name, type the new name in the space provided:

```

1      t      t      t      t      t      t      t      t      t      t      r
-----
*
Enter file name (/usr/larry/quiksell/sample): sample2
QUIKSELL MAKES THE HARD SELL EASY
Software tools for the sales industry
that provide a revolutionary approach
to sales.

      from the
      QuikSell Corporation

HISTORY-MODE                               INSERT   Line      1 ( 12)

```

3. Use **EXECUTE**. The new file, `sample2`, consists of just one version. It will have no history until changes are made to it. The old `sample` file retains all of its history.
4. You can also save this version under the original file name. To do this, simply use **EXECUTE** without typing in a new file name. The current version of the file, with all of its history, will be placed in `file.bak`, and the version of the file you are looking at will become the current version. All other

history will be removed. (See *rmhist(1)* for more information about removing history from a file.)

5. Use **ZOOM-OUT** to return to the History of file form, then **ZOOM-OUT** again to return to your original file.

6.4 Removing History

As noted previously, a structured file may grow quite large over a lifetime of changes. In order to save disk space, it's a good idea to remove the history from a structured file when the history is no longer needed. You can copy your files to diskette or tape before removing history if you wish. Your New Task Menu provides a convenient way to remove the history from all structured files in your home directory and all of its subdirectories.

1. In your sample file, use **MENU**:

```

1      t      t      t      t      t      t      t      t      t      r
Many Fortune 1000 companies have already discovered that

QUIKSELL MAKES THE HARD SELL EASY

S<-----ent and accurate.

New Task Menu

Select a menu option (for example, move the
cursor to an item and EXECUTE); otherwise,
CANCEL to remove the menu, or HELP to display
help information.

Show home directory
Execute UNIX shell commands
Run a shell command in a box
Show your profiles directory
Edit your editor profile
Housekeep
Display history of current file

/usr/larry/quiksell/sample          INSERT      Line      8 ( 10)

```

2. Select Housekeep. The message Executing "Housekeep" will appear in a popup box and your keyboard will be locked while the system removes the history from all structured files in your directories, removes all ... and .bak files in your directories, and removes \$HOME/.putdir.
3. When Housekeep has completed, a popup box with the message No output from "Housekeep" will appear.

1. Use **MENU** and select Edit your editor profile.

```

                                Editor Profile File

This file allows you to modify the behavior of the editor to suit your
preferences. Put the cursor on the line of interest and ZOOM-IN to specify
your choice of options.

MENU Options
HELP Options
Files the Editor Should Watch
Editor Search Paths

/usr/larry/profiles/editorprf          INSERT   Line   1 (   4)

```

This is the top level of your editorprf. Each line corresponds to a record. Each record listed here contains more detailed levels, which can be accessed with **ZOOM-IN**.

2. **ZOOM-IN** on MENU Options:

```

                                MENU Options

Description shown in menu      Type      Name of file or program

Show home directory           file      $HOME
Execute UNIX shell commands  screen   echo "Touch CNTL-D to continue ed
Run a shell command in a box  popbox   "Shell command:"
Show your profiles directory  file     $HOME/profiles
Edit your editor profile      file     $HOME/profiles/editorprf
Housekeep                     popbox   housekeep
Display history of current file helper    history

/usr/larry/profiles/editorprf          INSERT   Line   1 (   6)

```

3. With the cursor on Show home directory, **ZOOM-IN**:

Details of MENU Option		
Description shown in menu	Type	Name of file or program
Show home directory	file	\$HOME
Flags: any non-space character means true		
<input type="checkbox"/> Sync and reopen file	<input type="checkbox"/>	Save all files

/usr/larry/profiles/editorprf INSERT Line 1 (1)

Each of these more detailed levels is a part of the MENU Options record, which in turn is part of the editorprf file.

You can use the History Display to recreate a version of a structured nontext file such as editorprf. Remember that you will be recreating the entire file and not just one portion of it. The recreated version will include both the top level listing of all the records and the more detailed levels of text contained in each of those records.

It should be noted that versions of structured nontext files, such as editorprf, must be saved in files with specific names. For editor profile files, the name must either be editorprf or end in .editorprf. For example, you might save a version of your editorprf under a name such as old.editorprf. For File Manager profile files, the name must be either indexprf or end in .indexprf. For print profile files, the name must be either printprf or end in .printprf.

7. TEN/PLUS C PROGRAMMING HELPER

The C Programming Helper is designed to simplify compiling and debugging C source code. The INed editor automatically calls the C Programming Helper whenever you edit a file of C code. The editor considers any file named with the extension `.c` to be a C program.

The C Programming Helper keeps track of compiler errors and their line numbers, allowing you to debug code without having to exit the editor. The helper reduces the time it takes to debug a C program, since it records compiler output and uses the output to search for errors.

The C Programming Helper uses **LOCAL-MENU** options to position the cursor on each of the compiler errors and displays the compiler diagnostics in a popup box. Since the helper keeps track of the locations of subsequent errors even when their line numbers change, you can insert and delete lines as necessary. This allows you to fix errors in order, from the beginning of the code to the end. The helper also provides an option used to check for matching delimiters. When errors are corrected, you can use the helper from the editor to run `make(1)`.

This section assumes that you are familiar with the C programming language and the UNIX `make` facility. In addition, you should understand the procedures for entering and exiting the TEN/PLUS environment.

7.1 Using The C Programming Helper

The C Programming Helper provides seven options to facilitate compiling and making C programs. The options displayed by using **LOCAL-MENU** for the C Helper are:

- (1) compile current file
- (2) goto next compiler error
- (3) goto first compiler error
- (4) show compiler output
- (5) show C program
- (6) run make
- (7) show matching parenthesis

To exercise these options, you can either use **LOCAL-MENU**, or use one of the functions (1) through (7) (i.e., using function (1) compiles the current file). You might want to use **LOCAL-MENU** until you are familiar with these options and their associated functions.

Using these options, you can compile the current file, view compiler errors and the compiler output, run *make*, switch between the temporary compiler output file and the C source file, and check for matching delimiters. The following is a summary of the use of the C Helper to compile source code.

1. Create a file with the `.c` extension, such as `code.c`. This file can have either a structured or ASCII file type.
2. Enter code into `code.c`.
3. Compile `code.c`, using option (1) `compile current file`. You can stop the process at any time with **BREAK**. To see the output and diagnostics, use option (4) `show compiler output`. This puts you in a temporary file that contains the compiler output; this file is removed when you exit the editor.

Before compiling the code, the helper saves the source file and all ASCII files, including the Makefile.

4. Correct any errors. Use options (2) `goto next compiler error`, and (3) `goto first compiler error`, to move to the places in your file where the compiler caught errors. The use of **ENTER** with these options reverses their meaning, so that **ENTER** (3) goes to the last compiler error, and **ENTER** (2) goes to the previous compiler error. The helper moves the cursor to the line with an error, and displays the error message in a popup box. When you use these options, the C Programming Helper looks at the error list in the temporary file; this file is displayed by using option (4) `show compiler output`.

Use option (7) `show matching parenthesis` to check for matching delimiters. To use this option, move the cursor to either a parenthesis (`(` or `)`), a bracket (`[` or `]`), or a brace (`{` or `}`), then use (7). The helper either moves the cursor to the matching delimiter, or, if a matching delimiter is not found, displays an error message.

5. If necessary, recompile using option (1). When the program compiles without errors, a `.o` file is created, e.g., `code.o`.
6. Use option (6) `run make`. The helper asks you, in a popup box, which target program to make. You can either type in the target's name and **EXECUTE**, or accept the default target

name displayed in parentheses. To accept the default, **EXECUTE** without entering a target name. The initial default is the current source code name without the `.c` extension (e.g., `code` is the default for `code.c`). Otherwise, the default is the last target name you entered. The helper shows you the operations *make* is performing and their diagnostics. The *make* can be interrupted at any time using **BREAK**.

7. After *make* has finished, you are looking at a temporary file containing the output of the compiler and the loader. If you wish, you can switch back and forth between the source file and this file using options (4) and (5).
8. You can run the program without exiting the TEN/PLUS environment by: dropping into a subshell (if the program is interactive), running the program in a popup box, or running the program as a filter on a temporary file.

Options (1) and (6) can both be used to compile a file. There are, however, differences between the two. Option (1) compiles the file, but does not link it; option (6) can do both. With option (1) the helper moves the cursor to the locations in your file where the compiler has caught errors; the helper cannot do that if you use option (6). Another significant difference between options (1) and (6) is how these two options handle structured files. Since the C compiler does not understand structured files, it is important to convert the file to ASCII before doing the *make*. This can be done by using option (1) (which also does a *readfile*(1) if the file is structured) or by adding an explicit *readfile* command to *Makefile*. You can avoid considerations about whether your files are structured or ASCII and how to keep track of lines with compiler errors by always using option (1) before using option (6).

8. ADDITIONAL FUNCTIONS

8.1 More About HELP and the Help Menu

HELP provides additional information about TEN/PLUS functions and facilities. It is available at any time. There are two distinct situations in which **HELP** is useful:

1. When additional information about a currently displayed popup box is required. Further information about popup boxes is available by using **HELP** at any time while the popup box is displayed.
2. When additional information about any of the topics in the Help Menu is required.

To use the Help Menu:

1. Use **HELP**. The Help Menu will display:

```

INTERACTIVE TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                Description
-----                                -
1988.cal                            Calendar
                                     Help Menu
Select a menu option (for example, move the
cursor to an item and EXECUTE); otherwise,
CANCEL to remove the menu, or HELP to display
help information.

Alphabetic List of Editor Commands
How do I ... ?
How to Customize the Editing System
Suggestions for Your MENU
Suggestions for Your Print Menu
Keyboard Layouts

/usr/larry                            INSERT   Line   1 ( 7)

```

2. Select the desired topic. The help information for the topic selected displays on the screen. For example, if the Alphabetic List of Editor Commands is selected, a display containing an alphabetic list of editor commands will appear. Additional information about each of the editor commands can be accessed by moving the cursor to the line on which the desired command is listed and using **ZOOM-IN**. **ZOOM-OUT** to return to the list of editor commands.

3. Use **USE** to return to the File Manager display or the editing session, depending on where you were when **HELP** was used.

Practice using **HELP** until you are comfortable with its use:

1. **ZOOM-IN** to the report file, then use **HELP**.
2. Select Keyboard Layouts. A display showing a list of keyboards will appear. You can access a particular keyboard layout by moving the cursor to the line on which the desired keyboard is listed and using **ZOOM-IN**. **ZOOM-OUT** to return to the list of keyboards.
3. Use **USE** to return to the editing session.

8.2 Communicating With the System While in the TEN/PLUS Environment

The TEN/PLUS system eliminates most of the requirements for communicating directly with the UNIX System since most TEN/PLUS operations are performed with functions instead of commands. Files and directories can be created, edited, copied, moved, and deleted with the basic TEN/PLUS functions. There are, however, a number of system commands available while in the TEN/PLUS environment that perform a variety of useful functions and provide specific information about the system. A few of the more common system commands are provided in this section as examples. Others are provided in your user's manual.

There are three methods of communicating with the system while in the TEN/PLUS environment. The first two involve the use of **MENU** and have been discussed in §5.12.1 and §5.12.2. The third involves the use of **ENTER**.

8.2.1 Running System Commands

The procedure for running a command using **ENTER** is:

1. Use **ENTER**. The cursor moves into a popup box containing the **ENTER:** prompt.
2. Type the command.
3. Use **DO** or **MENU**. **DO** causes the output of the command to print at the cursor position, replacing the text from the cursor position to the next blank line. **MENU** causes the output to print in the popup box.

The output for the command is displayed in the file or the popup box, depending on which function is used. If **MENU** is used and there is no specific output for the command (for example if the *cp(1)* command is used to copy a file), a popup box with the message No output from "command" will display on the screen. Use **CANCEL** to remove the popup box.

1. Use **ENTER** date **MENU** to run the *date(1)* command and display the results in a popup box:

```

1      t      t      t      t      t      t      t      t      t      r
Preliminary report by Pat:
[Fri Mar 11 19:47:32 PST 1988] d designs for QuikSell as you requested. John
visuals are just too stuffy for a progressive young company like QuikSell
Corp. The photo of business people in a conference room setting doesn't have
the impact we're looking for--this client deserves something as dynamic as
their product.

I recommend a less conservative approach, although still using photography
rather than illustrations. I envision a scene with
one prop against a white background. A half a dozen or more business people
are crowded around in front of the booth, waiting for their turns to buy.
They're writing out checks or waving their checkbooks impatiently. The
salesman is taking orders and checks as fast as he can (life should be so
good!). This, I believe, is more consistent with the image we want and
certainly has the impact we're looking for.

Also, about the Cornerstone Condos concept--"Homes Built To Last" misses the
mark. These are inexpensive condos whose biggest selling point is their cheap
/usr/larry/report          INSERT      Line      1 ( 53)

```

2. Use **CANCEL** to remove the popup box.
3. Use **ENTER** **GO-TO** to go to the end of the file.
4. Use **ENTER** date **DO** to run the *date* command and print the results in the file. Because you are at the end of the file, the output is inserted without replacing existing text. If you wish to insert the output in the middle of a file without replacing text, use **INSERT** to open blank lines at the desired location before running the command.
5. **ZOOM-OUT**.

8.3 Exiting the TEN/PLUS Environment

Up to this point, the discussions in this tutorial have focused on using the TEN/PLUS system to perform basic system and editor

functions. If your system administrator has not arranged for you to automatically log in to the TEN/PLUS environment, you can exit from TEN/PLUS to the UNIX System shell. Although it is not usually necessary to exit the TEN/PLUS environment to perform system functions, there may be times when this is desirable.

The UNIX System shell is accessed by using **EXIT**. **EXIT** can be used from within a file or from the File Manager display. If **EXIT** is used from within a file, the file is saved before the system is brought to the command level.

The command level is indicated by a system prompt. Commands to the system are typed at the command level by typing the necessary characters from the keyboard and pressing the **ENTER** key. Commands must be typed accurately or the system will not recognize them. Type `e $HOME` to reenter the TEN/PLUS environment in your home directory. Type `e .` to reenter the TEN/PLUS environment in the current directory. Type `e` to reenter the TEN/PLUS environment in the file you were editing before exiting.

Exit and reenter the TEN/PLUS environment:

1. Use **EXIT**. The system displays the system prompt.
2. Type `date`, then press the **ENTER** key. The current date and time are printed and the system prompt returns.
3. Type `e $HOME` to reenter the TEN/PLUS environment in your home directory.

8.4 Sorting Columns

The `sort(1)` command is used on ASCII files to sort single-spaced lines or columns either alphabetically or numerically. It is used from the editor via **ENTER** and **DO**. The `sort` command acts on one paragraph unless otherwise specified. (The system interprets a blank line following text to be a paragraph separator.)

To sort a column, place the cursor on the first line to be sorted and use **ENTER** `sort` **DO**. To sort only a given number of lines, indicate the number of lines followed by the letter `l`. For example, to sort ten lines, use **ENTER** `10l sort` **DO**. Lists can be sorted on specific columns if each entry within a column is preceded by the same number of blank characters. To sort a specific column, type the number of columns to be skipped, preceded by a `+`, after the `sort` command. For example, to sort the third column, use **ENTER** `sort +2` **DO**. The `-b` option is used to instruct the system to

ignore leading blanks and tabs. For example, use `[ENTER] sort -b +2 [DO]`. To sort a numeric column, insert `-n` anywhere on the command line. For example, use `[ENTER] sort -b -n +2 [DO]`. For a discussion of more advanced sorting techniques, refer to *sort*(1). Type some text to be sorted and practice the variations of *sort* that apply to your work.

8.5 Performing Global Search-and-Replace

The global search-and-replace command, *rpl*(1), is used on text files to replace a word or string of words with another word or string of words throughout a line, paragraph, or document. Like the *sort* command, it is used through the editor via `[ENTER] command [DO]` and acts on one paragraph unless otherwise specified.

When searching for and replacing a string of words that contains a space (or spaces), enclose the string in quotation marks to hold the words together; for example, `[ENTER] 1931 rpl "XXX Corporation" "YYY Company" [DO]`. (Note that the number 193 is followed by the letter `l` to indicate the number of lines to be affected.)

To identify one form of a word to be replaced, without replacing variations, put a space before and/or after the word. For example, `[ENTER] 20 rpl "tree " "shrub " [DO]`. This example ensures that `tree` is replaced, but not `trees`.

To initiate a global replacement use `[ENTER] n1 rpl target-string replacement-string [DO]`, where *target-string* is the original string being searched for and *replacement-string* is the string that is to replace the target string. The argument *n* without `l` is the specific number of paragraphs, and *n1* is the specific number of lines.

To use the global search-and-replace command:

1. Determine the area where the word or string is to be replaced. If *n* is not specified, the replacement is made within the paragraph where the cursor is positioned. To modify the default, specify the number of paragraphs (*n*) or lines (*n1*) from the cursor position in which the replacement should be made.

For example, to replace all occurrences of a particular string throughout a document:

- a. Use `[ENTER] [GO-TO]` to go to the end of the report file.

- b. Note the number of lines in the document.
 - c. Instruct the system to search that number of lines, or use a very large number, such as 999. Alternatively, estimate the number of lines in your file, and use a large number.
2. Position the cursor at the beginning of the area where the “search-and-replace” is to begin.
 3. Use **ENTER**.
 4. Enter the number of paragraphs or lines (if required), the target string, and the replacement string.
 5. Use **DO**.

For example, to replace `xxx` with `yyy` in the next three paragraphs, use **ENTER** `3 rpl xxx yyy` **DO**. To replace `xxx` with `yyy` in the next 100 lines, use **ENTER** `100l rpl xxx yyy` **DO**. (Note that the number 100 is followed by the letter l.)

These characters have special meanings to `rpl`:

`. [] () { } * ^ $`

Use these characters with caution; they must all be preceded by a `\` to escape their special meanings. (See `rpl(1)` for details.)

8.6 Using the Spell Program

The `spell(1)` program is used to check the spelling of all words in a text file against a list of commonly used words. Words not on the list or derivable by adding certain prefixes, suffixes, or inflections to a word in the list, are output as possible errors.

The `spell` program helps discover typographical errors; it does not proof the document. It searches only for misspelled words; a correctly spelled word that is used incorrectly is ignored. For example, if `in` is typed instead of `is`, the `spell` program will not detect it as an error. In addition, the list of “error” words may include correctly spelled words that are not in the system dictionary; for example, a person’s last name. Thus, the entries in the output file are not necessarily misspelled words.

The `spell` program can be run using Execute UNIX shell commands or Run a shell command in a box from the New Task Menu as described in §5.13, or it can be run using **ENTER** and **MENU**. To run `spell` using **ENTER** and **MENU**:

1. Use **ENTER**.
2. Type `spell filename > typofile`, where *filename* is the name of the file being checked and *typofile* is the name of the file in which possible errors will be written. If *typofile* does not already exist, it is created. If *typofile* does exist, its existing contents are deleted before anything is written. If differentiation between words not on the list and words with plausible derivatives is desired, the `-v` option should be specified before *filename* in the `spell` command (`spell -v filename > typofile`).
3. Use **MENU** (not **DO**).

Check the spelling in a file called `test` and output the errors to a file called `typos`:

1. Use **ENTER**.
2. Type `spell test > typos`, then use **MENU**. The system confirms that the command is being executed by displaying the message `Executing "spell test > typos"` in a popup box.
3. The message `No output from "spell test > typos"` appears in the popup box when the program is finished running. Use **CANCEL** to remove the popup box.
4. In the `typos` file, position the cursor at the first typographical error listed, then use **ENTER** **+SEARCH**. This places the typographical error in the search buffer. If the typographical error is listed in the `typos` file only once, the system responds `Search failed on string "string"`. The typographical error remains in the search buffer.
5. Switch to your original file with **USE** or **NEXT-WINDOW**, then use **+SEARCH** and **-SEARCH** to locate the search string. Continue searching until the system responds with `Search failed on string "string"`.

Repeat this procedure until all typographical errors are corrected.

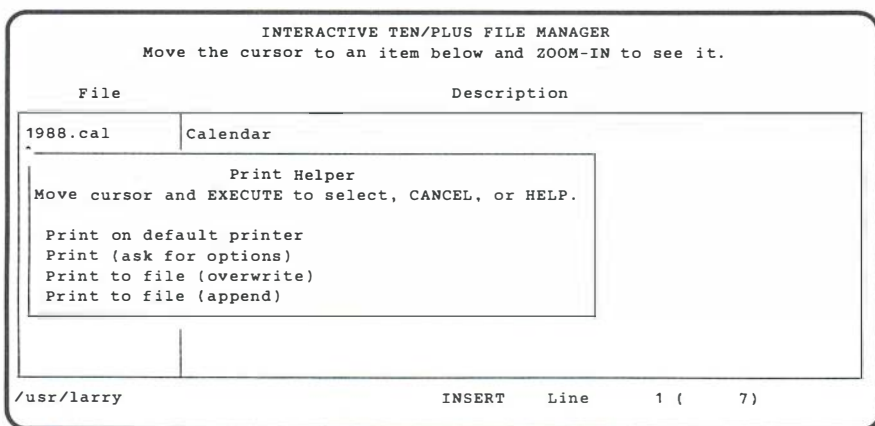
Use this procedure to check the spelling in the report file (**ENTER** `spell report > typos` **MENU**). Find and correct any errors.

8.7 Printing a Document

Documents created with the INEd editor can be printed on a variety of devices by using **PRINT**. **PRINT** produces a reproduction of a file exactly as it is displayed on-line, regardless of how much is visible on the screen when **PRINT** is used. **PRINT** can also be used to print a directory listing by using **PRINT** while the listing is displayed.

To print a document:

1. Access the file or directory to be printed.
2. Use **PRINT**. A menu similar to this one appears on the screen:



3. Select the desired print option. Standard print options are described below:

Print on default printer

Causes the file or directory to be printed on the printer assigned to your work station. Page numbers and the file name are printed at the top of each page, and the title on the header page is set to the name of the current file. Each work station should have an assigned default printer.

Print (ask for options)

Pipes the **PRINT** output through the print command for your system, sets the title on the header

page to the name of the current file, and prompts you for print options.

Print to file (overwrite)

Prompts for a file name and overwrites the specified file with the output.

Print to file (append)

Prompts for a file name and writes the output to the end of the specified file.

If you wish to modify or add to the default options on your print menu, you must copy the system print profile, `printprf`, into your `profiles` directory and edit it. The system `printprf` file is located in the `$SYS/profiles` directory. For more information about modifying your `printprf` file, refer to “TEN/PLUS Profiles.”

8.8 Splitting a File

Sometimes it is helpful to split a lengthy document into two or more shorter documents. The shorter files can then be edited by more than one person and joined together when editing is complete.

The `split(1)` command splits the original file and creates new temporary files with `aa`, `ab`, `ac`, and so on, appended to each new file name. If a new file name is not designated, the system assigns the file name `x`.

The `split` program can be run using `Execute UNIX shell commands` or `Run a shell command in a box` from the `New Task Menu` as described in §5.13, or it can be run using `ENTER`. To split a file using `ENTER`:

1. Determine the number of lines each new file should contain and choose a new file name to be used for the split files. (One way to determine the number of lines the new files should contain is to access the original file and use `ENTER` `GO-TO` to move to the end of the file. Note the number of lines in the document and divide that number by the number of temporary files the document is to be split into. This will result in new files of approximately equal length when the `split` command is issued.)
2. Use `ENTER`.

3. Type `split -n filename newfilename`, where *n* is the number of lines each file should contain, *filename* is the name of the original file, and *newfilename* is the name to be used for the new (split) files.
4. Use **MENU**. After a brief pause, the message "No output from "split -n filename newfilename" appears in a popup box. Use **CANCEL** to remove the popup box.

When all the new temporary files have been revised, use the `cat(1)` command (**ENTER** `cat list of filenames > filename` **MENU**) to combine them all back into one file. For example, to combine `rev2aa`, `rev2ab`, `rev2ac`, and `rev2ad` into a file named `revision2`, use:

```
cat rev2aa rev2ab rev2ac > revision2
```

Split the `report` file into three files using `split`, then rejoin them using `cat`.

8.9 Searching for a Subject Within a Directory

The `grep(1)` command can be used to find a file in a specific directory when the subject is known but the file name is not. Use this procedure to locate references to a subject within a directory:

1. Change to the directory in which the search is to be conducted.
2. Use **ENTER**.
3. Type `grep subject *`, where *subject* is the word or phrase to be searched for. If the subject consists of more than one word, enclose it in quotation marks ("). For example, to search for the names of all files that mention the Widget Corporation, type `grep "Widget Corporation" *`.
4. Use **MENU**. The system lists all files in which the subject is mentioned, followed by the lines containing the word or phrase *subject*.

Practice using `grep` by searching for the subject `QuikSell` in your home directory.

8.10 Changing Your Password

Your password should be changed periodically to ensure the security of your files. To change your password:

1. Use **MENU**.
2. Select **Execute UNIX shell commands**, then use **EXECUTE**. The window is removed and the editor subshell prompt displays. The cursor is positioned after the prompt.
3. Type `passwd`, then press the **ENTER** key. These prompts ensure that only the user can change the password and that there are no errors in setting the new password:

```
Old password:
New password:
Re-enter new password:
```

4. Enter the old password, then the new password, then reenter the new password. Remember that you must press the **ENTER** key after each entry before typing the next.

If an incorrect old password is given, the system responds **Sorry**, and the editor subshell prompt returns. To change your password, you must reenter the `passwd(1)` command and press the **ENTER** key.

If there is an error when typing in the new password, the system responds with **They don't match; try again** and prompts for the new password. The new password should be entered and then reentered as prompted.

8.11 Recovering From System Crashes

The INed editor automatically recovers an editing session if the system crashes while the session is in progress. The next time the file is accessed, it should be in the same state as it was when the system crashed.

Although automatic recovery is part of the editing system, it is recommended that files be **SAVE**d periodically as an extra precaution in case of failure. If the recovery process fails, the file can be accessed as it was when it was last **SAVE**d by editing the `.bak` file.

The message `Saving file "/pathname/filename" . . .` displays in a popup box when **SAVE** is used. When the popup box disappears, file editing can resume.

8.12 Removing Line-Noise Characters and Broadcast Messages

Occasionally, the system prints warning messages in the editor window or displays characters not keyed in the file. When this occurs, use **REFRESH**.

When **REFRESH** is used, the window goes blank for a few seconds, and then previous text is rewritten exactly as it was prior to the interference. When **REFRESH** is completed, the cursor returns to the position it occupied before the display was cleared.

8.13 Using Wildcards

The `*` is a wildcard character that matches any string of characters. It saves typing and makes it possible to name groups of files as arguments to a UNIX System command.

When typing a command that applies to several files, use the `*` to denote zero, one, or more characters that appear in all the file names, rather than typing the name of each file. For example the `*` can be used with `ls(1)`:

```
ls *
```

lists *all* files within a directory.

```
ls intro*
```

lists all files whose names begin with the characters `intro`.

```
ls *.bak
```

lists all files whose names end with the characters `.bak`.

Another wildcard is `?`, which represents exactly *one* character. For example, `ls h?t` lists all three-letter files within a directory that begin with an `h` and end with a `t`, so that files named `hat`, `hot`, and `hit` are listed, but `hunt` is not.

Brackets (`[]`) can be used to indicate a range. For example `ls draft[1-5]` lists all files within a directory that begin with the letters `draft` and end with `1`, `2`, `3`, `4`, or `5`, so that files named `draft1` or `draft3` are listed but `draft7` is not.

Appendix: TEN/PLUS FUNCTIONS

This appendix describes the capabilities of the TEN/PLUS functions and sequences. Refer to the “TEN/PLUS Reference Manual” for additional information about these functions and sequences.

1. CONTROL AND MENU FUNCTIONS

- DO** Invokes system commands from within the editor when used with **ENTER**. **ENTER** *command* **DO** places output in the current file, replacing text from the cursor line to the end of paragraph.
- ENTER** Modifies the capabilities of most functions. Used in conjunction with **DO** or **MENU**, it invokes system commands from within the TEN/PLUS Environment.
- EXECUTE** Invokes functions that appear in menus or help popup boxes. Used in conjunction with the cursor-positioning functions to indicate the desired function.
- CANCEL** Removes comment, instruction, help, or other popup boxes.
- BREAK** Stops a **+SEARCH** or **-SEARCH**, or a command started via **EXECUTE**.
- HELP** Displays additional help instructions relating to the editor or to the comments in the current popup box. Often used in conjunction with **EXECUTE**, **ZOOM-IN**, and **ZOOM-OUT** for more detail on a particular subject.
- MENU** Accesses non-editor functions without having to **EXIT** the TEN/PLUS Environment. Used in conjunction with **EXECUTE** to choose menu options. **ENTER** *command* **MENU** places the output of *command* in a popup box.
- PRINT** Accesses a print menu.
- LAST-ARG** Displays the last **ENTER** popup box argument.

QUOTE Allows the explicit insertion of control characters into the text.

REFRESH Clears the editor window of “garbage” characters or broadcast messages. Use **REFRESH** to rewrite the editor window and position the cursor as it was before the interruption.

ENTER REFRESH If TEN/PLUS is running in a windowing environment that supports windows, use **ENTER REFRESH** to rewrite the editor window, using the new window size.

FUNCTIONS Displays a menu listing which of the ten basic TEN/PLUS functions are active.

LOCAL-MENU Displays the local, data-specific task menu.

(1)–(8) Local menu bypass functions.

2. WINDOW-POSITIONING FUNCTIONS AND SEQUENCES

+LINE
-LINE Moves the text one-third of a window forward (+) or backward (-) within the file.

ENTER *n* **+LINE**
ENTER *n* **-LINE** Moves the text the specified number of lines forward (+) or backward (-) within the file, rather than the one-third window default.

ENTER **+LINE**
ENTER **-LINE** Moves the cursor line to the top (+) or the bottom (-) of the window.

+PAGE
-PAGE Moves the text one window forward (+) or backward (-) within the file.

ENTER *n* **+PAGE**
ENTER *n* **-PAGE** Moves the text the specified number of windows forward (+) or backward (-) within the file, rather than the one window default.

RIGHT
LEFT Moves the window one-third of its width to the right or left, to permit viewing text that is off the display.

ENTER *n* **RIGHT**
ENTER *n* **LEFT**

Moves the window the specified number of characters to the right or left, rather than the default number of characters.

ENTER **+SEARCH**
ENTER **-SEARCH**

Puts the string of characters beginning at the cursor position and ending at the first blank space in the search buffer and initiates a forward (+) or backward (-) search.

ENTER *string* **+SEARCH**
ENTER *string* **-SEARCH**

Puts *string* in the search buffer and initiates a forward (+) or backward (-) search.

+SEARCH
-SEARCH

Searches forward (+) or backward (-) for the current contents of the search buffer.

GO-TO

Moves the cursor to line 1 of the file. If the cursor is at line 1, moves the cursor to the last line of file.

ENTER **GO-TO** Moves the cursor to the last line of the file.

ENTER *n* **GO-TO**
 Moves the cursor to the specified line in the file.

3. CURSOR-POSITIONING FUNCTIONS

↑ **←** **↓** **→**

Cursor-positioning functions. Used in conjunction with **BOX-MARK** and **TEXT-MARK** and the text manipulation functions **INSERT**, **PICK-UP**, **DELETE**, and **PICK-COPY** to define a specific area of text.

RETURN Moves the cursor to the beginning of the next line.

HOME Moves the cursor to the upper left-hand corner of the current window.

TAB
-TAB Moves to the next tab position, to the right (**TAB**) or to the left (**-TAB**).

- BACKSPACE** Corrects typing errors by backspacing over (thereby erasing) them.
- BEGIN-LINE** Moves the cursor to the first nonblank character on the current line.
- END-LINE** Moves the cursor immediately to the right of the last nonblank character on the current line.
- LINE-FEED** Positions the cursor at the beginning of the next line of a form.

4. TAB SETTING/RELEASING FUNCTIONS AND SEQUENCES

- SET-TAB** Sets a tab at the cursor position.
- ENTER SET-TAB** Removes the tab (if any) at the cursor position.

5. TEXT MANIPULATION FUNCTIONS AND SEQUENCES

- MARGIN** Sets a left-hand margin at the cursor position.
- ENTER MARGIN** Sets a right-hand margin at the cursor position.
- ENTER → MARGIN** Sets left- and right-hand margins. Sets the left-hand margin at the cursor location where defining began and sets the right-hand margin at the cursor location where defining ended.
- FORMAT** Reformats text from the cursor position to the end of the paragraph according to the margins indicated on the display.
- ENTER c FONT** Selects continuous-underscore font.
- ENTER w FONT** Selects word-underscore font.
- ENTER FONT** Selects nonunderscore font.
- ENTER r FONT** Selects Roman font (equivalent to **ENTER FONT**).

ENTER g **FONT**

Selects graphics font (if this font is available on your display).

FONT

Switches between two fonts.

CENTER

Centers the cursor line between the margins.

ENTER ↓ **CENTER**

Centers the defined text between the margins.

ENTER n **CENTER**

Centers a specific number of lines starting with the cursor line.

INSERT-MODE

Switches between insert mode and overwrite mode. When insert mode is on, insertion of text is allowed (text to the right of the insertion moves to the right). When overwrite mode is on, text can be deleted by striking over it.

DELETE-CHARACTER

Deletes the character at the cursor position.

ENTER **DELETE-CHARACTER**

Deletes the portion of a line at and to the right of the current cursor position.

INSERT

Creates one blank line at the cursor position by moving the cursor line and all succeeding lines down.

ENTER n **INSERT**

Creates the specified number of blank lines immediately below the cursor line.

BOX-MARK ↓ **INSERT**

Creates a blank space equal to the number of defined lines, moving text at the cursor position down.

BOX-MARK ↓ → **INSERT**

Creates a rectangular area of blank space.

ENTER **INSERT**

Moves text at and to the right of the cursor to the new line below.

PICK-UP Deletes text and adds it to the pick buffer.

ENTER *n* **PICK-UP** Deletes the specified number of lines and adds them to the pick buffer.

BOX-MARK ↓ **PICK-UP** Deletes all of the defined lines and adds them to the pick buffer.

TEXT-MARK ↓ **PICK-UP** Deletes the defined area and adds it to the pick buffer.

BOX-MARK → **PICK-UP**
TEXT-MARK → **PICK-UP** Deletes the defined portion of the line and adds it to the pick buffer.

BOX-MARK ↓ → **PICK-UP** Deletes a rectangular area of text and adds it to the pick buffer.

TEXT-MARK **RETURN**
 → **PICK-UP** Deletes the defined area and adds it to the pick buffer.

ENTER **PICK-UP** Replaces text at and to the right of the cursor with text on the line below the cursor.

PUT-DOWN Places the text most recently added to the pick buffer at the cursor position, removing it from the pick buffer.

DELETE Deletes text and adds it to the wastebasket buffer.

ENTER *n* **DELETE** Deletes the specified number of lines and adds them to the wastebasket buffer.

BOX-MARK ↓ **DELETE** Deletes the defined lines and adds them to the wastebasket buffer.

BOX-MARK → **DELETE**

TEXT-MARK → **DELETE**

Deletes the defined portion of the line and adds it to the wastebasket buffer.

BOX-MARK ↓ → **DELETE**

Deletes a rectangular area of text and adds it to the wastebasket buffer.

TEXT-MARK **RETURN**

→ **DELETE**

Deletes the defined area and adds it to the wastebasket buffer.

ENTER **DELETE**

Replaces text at and to the right of the cursor with text on the line below the cursor.

RESTORE

Removes only the most recently **DELETE**d text from the wastebasket buffer and places it at the cursor position.

ENTER *n* **RESTORE**

Places the specified number of copies of the most recently **DELETE**d text at the cursor position.

PICK-COPY

Adds a copy of the line at the current cursor position to the pick buffer.

ENTER *n* **PICK-COPY**

Adds a copy of the specified number of lines to the pick buffer.

BOX-MARK ↓ **PICK-COPY**

Adds a copy of the defined lines to the pick buffer.

TEXT-MARK ↓ **PICK-COPY**

Adds a copy of the defined area to the pick buffer.

BOX-MARK → **PICK-COPY****TEXT-MARK** → **PICK-COPY**

Adds a copy of the defined portion of the line to the pick buffer.

BOX-MARK ↓ → **PICK-COPY**

Adds a copy of a defined rectangular area to the pick buffer.

TEXT-MARK **RETURN****PICK-COPY**

Adds a copy of the defined area to the pick buffer.

PUT-COPY

Places a copy of the text most recently added to the pick buffer at the cursor position, leaving it in the buffer.

ENTER *string* **REPLACE**

Puts *string* in the replace buffer. If the string at the cursor position matches the string in the search buffer, it is replaced with the string in the replace buffer.

REPLACE

If the string at the cursor position matches the string in the search buffer, it is replaced with the string in the replace buffer.

6. MULTIPLE FILE/WINDOW FUNCTIONS AND SEQUENCES

USE

Accesses the alternate file.

ENTER *filename* **USE**

Makes *filename* the current file. (*Filename* may be an existing or a new file.) The current file becomes the alternate file.

WINDOW

Makes a new window showing the current file.

ENTER *filename* **WINDOW**

Creates a window and puts *filename* into that window. (*Filename* may be an existing or a new file.)

NEXT-WINDOW

Changes the current window, in the order in which windows were created.

ENTER WINDOW	Removes all windows except the one in which the cursor currently resides.
NEXT	Displays the next item in a list.
PREVIOUS	Displays the previous item in a list.
ZOOM-IN	Displays more detailed information.
ZOOM-OUT	Displays more summary information.

7. EXIT/SAVE FUNCTIONS AND SEQUENCES

EXIT	Exits the TEN/PLUS environment and returns to the command level, saving the exited file as created or revised.
QUIT	Abnormally terminates the editing session, attempting to save all files. QUIT produces a core file and should be used only if EXIT fails.
ENTER q EXIT	Exits the editor and cancels all changes made to all ASCII files during the current editing session unless they have been explicitly SAVE d.
SAVE	Saves a file without exiting the editor. Each SAVE creates a new .bak file, overwriting the previous .bak file.

TEN/PLUS Reference Manual

CONTENTS

1. INTRODUCTION	1
1.1 Accessing TEN/PLUS Functions	1
2. TEN/PLUS SYSTEM OVERVIEW	2
2.1 Components and Capabilities	2
2.1.1 The INEd Editor	2
2.1.2 The File Manager	3
2.1.3 The Profile Helper	3
2.1.4 The Print Helper	3
2.1.5 The History Display	3
2.2 File Types	4
2.3 Initializing the TEN/PLUS System	4
2.4 Exiting the TEN/PLUS Environment	5
3. THE FILE MANAGER	6
3.1 Creating Files	6
3.2 Exiting Files	6
3.3 Accessing Existing Files	6
3.4 Copying Files	6
3.5 Moving Files	7
3.6 Renaming Files	7
3.7 Deleting Files	7
3.8 Creating Directories	7
3.9 Accessing Directories	8
3.10 Copying Directories	8
3.11 Moving Directories	9
3.12 Renaming Directories	9
3.13 Removing Directories	9
3.14 Using LOCAL-MENU With the File Manager	9
4. THE HISTORY DISPLAY	10
4.1 Accessing the History Display	10
4.2 Accessing the Versions of a File	10
4.3 Using LOCAL-MENU with the History Display	10
4.4 Saving a Version of a File	10
5. CURSOR- AND WINDOW-POSITIONING FUNCTIONS	11

5.1	Cursor-Positioning Functions	11
5.2	Window-Positioning Functions	12
6.	TEN/PLUS SYSTEM FUNCTIONS	14
6.1	Canceling Operations: BREAK	14
6.2	Error Message and Other Explanations: HELP	14
6.3	Accessing the Operating System and Other Files: MENU	14
6.4	Removing Boxes: CANCEL	15
6.5	Controlling Detail: ZOOM-IN and ZOOM-OUT	15
6.6	Printing a Document: PRINT	15
6.7	Selecting Data-Specific Functions: LOCAL-MENU	16
6.8	Looking at Other Items: NEXT and PREVIOUS	16
6.9	Editing and Creating Alternate Files: USE	16
6.10	Saving Files Without Exiting the Editor: SAVE	17
7.	INTERACTIVE TEXT EDITING FUNCTIONS	18
7.1	Inserting Characters: INSERT-MODE	18
7.2	Inserting Lines and Areas, Splitting Lines: INSERT	18
7.3	Moving Lines and Areas, Joining Lines: PICK-UP	19
7.4	Backspace: BACKSPACE	20
7.5	Deleting Characters: DELETE-CHARACTER	20
7.6	Deleting the End of a Line: ENTER DELETE-CHARACTER	20
7.7	Duplicating Lines or Areas: PICK-COPY	20
7.8	Placing Text in a File: PUT-COPY and PUT-DOWN	21
7.9	Searching for a Text String: +SEARCH and -SEARCH	21
7.10	Canceling a Search: BREAK	22
7.11	Searching and Replacing: REPLACE	22
7.12	Moving to a Specified Line: GO-TO	22
7.13	Creating More Than One Window: WINDOW	23
7.14	Changing Windows: NEXT-WINDOW	23
7.15	Setting and Clearing Tabs: SET-TAB	24
7.16	Editing Wide Lines: RIGHT and LEFT	24
7.17	Inserting Control Characters: QUOTE	24
7.18	Redrawing the Display: REFRESH	25
7.19	Deleting and Restoring: DELETE and RESTORE, # File	25
7.20	Using FUNCTIONS	25
8.	ARGUMENTS TO FUNCTIONS	27
8.1	Typed Arguments	27
8.2	Cursor-Defined Arguments	28

8.2.1	Designating Lines	28
8.2.2	Designating Blocks	28
8.2.3	Designating Running Text Regions	29
9.	TEXT PROCESSING FUNCTIONS	31
9.1	Using Arguments: ENTER argument DO	31
9.2	Replacing Strings: rpl	32
9.3	Formatting Text: FORMAT	32
9.4	Setting Margins: MARGIN	32
9.5	Centering Text: CENTER	33
9.6	Selecting Fonts: FONT	33
9.7	Canceling a Filter Operation: BREAK	33
9.8	More DO Capabilities: tee, cat, sort, and date	33
10.	RECOVERY	35
10.1	Old Versions of ASCII Files	35
10.2	Old Versions of Structured Files	35
10.3	Automatic Recovery	35
11.	MISCELLANEA	36
11.1	Text File Format	36
11.2	TEN/PLUS Limitations	36
11.3	The Editor Profile	36
Appendix A:	SUMMARY OF TEN/PLUS FUNCTIONS	37
Appendix B:	UTILITY PROGRAMS FOR STRUCTURED FILES	43
Appendix C:	BOX CHARACTER SET	45
Appendix D:	ERROR MESSAGES	47
1.	GLOSSARY	47
2.	ERROR MESSAGES	48



TEN/PLUS* Reference Manual

1. INTRODUCTION

This is a reference manual for the TEN/PLUS system and the INed* editor. It is intended for those who have already used the TEN/PLUS system to create, edit, and manage text files. It is not designed to teach these procedures to users who have never used them. Readers of this manual should have already read the "TEN/PLUS Tutorial," which contains an introduction to the features of both the TEN/PLUS system and the INed editor. Refer to your user's manual for more detailed information about available UNIX* System commands.

1.1 Accessing TEN/PLUS Functions

Accessing TEN/PLUS functions requires different keystroke sequences on different keyboards. Refer to the appropriate section of "TEN/PLUS Keyboard Information" for an alphabetic listing of the TEN/PLUS functions and the keystroke sequences required for your keyboard.

2. TEN/PLUS SYSTEM OVERVIEW

2.1 Components and Capabilities

The TEN/PLUS User Interface consists of five components: the INed editor, the File Manager, the Profile Helper, the Print Helper, and the History Display. These are used to create, edit, and manage text files. Together, they provide a bridge to the UNIX System, allowing users to perform basic UNIX System functions without having to learn a lengthy set of system commands.

All functions that can be performed from the UNIX System shell can be performed using one of the TEN/PLUS User Interface components. Since all five components use a system of menus and functions that allow the user to select a desired operation, it is rarely necessary to remember a specific UNIX System command.

2.1.1 *The INed Editor*

INed is a screen-oriented text editor that allows users to display and edit text files. The INed editor is used to modify files by typing over existing text displayed in a two-dimensional window on the screen. The INed editor is easy to use because it allows users to enter and modify text files directly.

INed users enter and edit text by typing on the display as they would using a typewriter. Placing the cursor at the appropriate position on the display and then typing new characters enters them into the file.

The display contains a window that can be moved to the left or right to see other parts of the file. This window can be divided into several smaller windows for convenient editing or examination of files. This feature can be used in conjunction with other functions to “pick up” text from one file and “put” it into another file or another portion of the same file.

The INed command language employs a variety of functions for inserting, deleting, and moving characters, lines, blocks of text, and running text on the screen. INed functions are invoked by a series of keystrokes. Additional INed features include text processing, paragraph fill, right margin justification, and global replacement. The INed editor can be expanded to include additional user- or system-provided programs that are invoked interactively from within the editor.

The INEd editor provides facilities for recreating previous versions of files. In addition, if an editing session terminates prematurely, the INEd editor provides recovery mechanisms.

2.1.2 The File Manager

The TEN/PLUS File Manager allows users to create, access, move, copy, and delete files by positioning the cursor and using a specific function. It is easy to use the File Manager to perform these operations because there are no UNIX System commands or syntax conventions to remember.

The File Manager uses the same functions as the INEd editor. Files are picked up and moved, or copied and moved, using the same INEd functions that perform these operations on lines of text.

2.1.3 The Profile Helper

The TEN/PLUS Profile Helper allows users to customize the editing environment to suit individual needs. It can be used to help the editor locate forms, helpers, messages, and forms language scripts, as well as to add, change, or delete the options on certain menus.

The Profile Helper uses forms to build custom menus that simplify operations in the TEN/PLUS environment. Custom menus allow the user to perform certain routine tasks by selecting options from menus. Refer to “TEN/PLUS Profiles” for additional information about the Profile Helper.

2.1.4 The Print Helper

The TEN/PLUS Print Helper allows users to print a file by selecting an option from the Print Menu. The options on the Print Menu can be customized by editing the print profile.

Like the Profile Helper, the Print Helper uses forms to build the custom Print Menu. The custom Print Menu simplifies printing operations in the TEN/PLUS environment by giving the user a choice of menu options for printing a file. Refer to “TEN/PLUS Profiles” for additional information about the Print Helper.

2.1.5 The History Display

The TEN/PLUS History Display provides a way to keep track of changes made to structured files. It displays a form detailing information about previous changes to a file, and allows the user to recall and see any previous version. It also enables the user to copy a previous version of a file into the current file for editing.

The History Display uses menus and forms to implement its functions. History Display functions are accessed by positioning the cursor at the desired option on the menu or form and using a function. Like the File Manager, it does not require knowledge of any specific commands or syntax.

2.2 File Types

The INEd editor can edit two types of files: ASCII and structured. ASCII files are standard text files coded in the “American Standard Code for Information Interchange.” Most programs, such as compilers, expect data in this form. Structured files, while not directly usable with most UNIX System commands, can be converted easily to ASCII files (refer to Appendix B).

Structured files contain information about the structure of the data in the file. Structured files also contain information about how the file was created so that a user can recover previous versions. For ASCII files, only one previous version is kept in a .bak file. History is kept more compactly in a structured file, and changes to large structured files can usually be stored more quickly after editing than changes in large ASCII files. However, unless history is periodically removed, a structured file will become much larger than the corresponding ASCII file over a lifetime of editing.

A number of programs are useful for managing structured files. These programs are described in Appendix B.

2.3 Initializing the TEN/PLUS System

A successful login initializes the TEN/PLUS system when the system is configured to enter directly into the TEN/PLUS environment. If the File Manager screen does not appear after login, type `e $HOME` and press the **ENTER** key to initialize the TEN/PLUS system. This also initializes the editor.

The editor starts up in insert mode with the cursor in the top left-hand corner of the window. The procedures for using the INEd editor to modify files are described in detail in §6, §7, §8, and §9.

2.4 Exiting the TEN/PLUS Environment

To exit the TEN/PLUS environment, use **EXIT**.¹ If the system is configured to enter directly into the TEN/PLUS environment, **EXIT** will log you out. If the system is not configured to enter directly into the TEN/PLUS environment, this brings the system to the command level (indicated by the system prompt). To reenter the TEN/PLUS environment and return to the editing session, type **e** and press the **ENTER** key.

When you **EXIT**, all files edited during the session are saved. If the file is an ASCII file, the editor renames the previous version of the file by truncating the file name to 10 characters (if necessary) and then appending **.bak**. If an old **.bak** file exists, it is overwritten. If the file is a structured file, the previous version is retained as part of the history.

Unmodified files are not rewritten. Files are considered modified if any printing characters are typed while editing the file, even if they do not result in an alteration of the contents. For example, using the space bar (a printing character) instead of **→** causes a file to be saved, even if the intent is merely to examine it.

Use **ENTER** **q** **EXIT** to exit an ASCII file without saving the changes. Note that changes to all ASCII files edited since the last save will be canceled. To recover a previous version of a file, use the **.bak** file for ASCII files or the History Display for structured files.

-
1. If for any reason **EXIT** fails to bring the system out of the TEN/PLUS environment, use **QUIT**. **QUIT** terminates the editing session unconditionally, attempts to save all files, and produces the **core** file. (The **core** file can be used by a system programmer to determine the reason for the malfunction.)

3. THE FILE MANAGER

This section explains how to use the File Manager to create, access, and manipulate files and directories.

3.1 Creating Files

Files are created from the File Manager screen by typing a name and optional description, then using **ZOOM-IN**. A menu appears on the display, requesting that the user select the type of file or directory desired, or re-enter the file name. If a file type is selected, an empty window appears on the display. If a directory is selected, an empty File Manager display appears. All text is typed within the resulting window or File Manager display.

3.2 Exiting Files

Files are exited by using **ZOOM-OUT**. After a few seconds, the File Manager screen appears on the display.

3.3 Accessing Existing Files

Existing files are accessed from the File Manager screen by positioning the cursor on the line on which the file is listed and using **ZOOM-IN**. The text from the file appears on the display. Existing files may also be accessed by using **ENTER** *filename* **USE**.

3.4 Copying Files

Files are copied from the File Manager screen into the current directory or another directory by positioning the cursor in any field on the line on which the file is listed and using **PICK-COPY**. The cursor will move down one line. Position the cursor where the copy is to be placed and use **PUT-DOWN**. An instruction box with the message File name "*filename*" already exists. Press **CANCEL** to abort file restore, or enter a new name: appears on the display. Type the new file name at the cursor position and use **EXECUTE**. The file is copied into the current directory and listed on the File Manager screen at the cursor position. You can also copy multiple files using **BOX-MARK** or **ENTER** *vertical-motion* with **PICK-COPY** and **PUT-DOWN**.

To copy a file into another directory, access that directory using the procedure described in §3.9 before using **PUT-DOWN**. In

this case, the file name usually will not be duplicated so the instruction box will not display.

3.5 Moving Files

Files are moved by positioning the cursor in any field on the line on which the file is listed and using **PICK-UP**. The file is removed from the directory listing and the files listed below it move up one line. Change to the directory where the file is to be located using the procedures described in §3.9, and use **PUT-DOWN**. The file is moved into the current directory and listed on the File Manager screen at the cursor position. You can also move multiple files using **BOX-MARK** or **ENTER** *vertical-motion* with **PICK-COPY** and **PUT-DOWN**.

3.6 Renaming Files

Files are renamed from the File Manager screen by positioning the cursor at the appropriate character in the file name and using any of the editing functions for modifying text. For example, change into overwrite mode by using **INSERT-MODE**, type over the old file name, and delete any extra characters by using **DELETE-CHARACTER**.

3.7 Deleting Files

Files are deleted from the File Manager screen by positioning the cursor in any field on the line on which the file is listed and using **DELETE**. You can also delete multiple files using **BOX-MARK** or **ENTER** *vertical-motion*. The file (or files) is removed from the directory and stored in `$HOME/.putdir`. To restore a file from `$HOME/.putdir`, use **PICK-COPY**, access the desired directory, then use **PUT-DOWN**.

3.8 Creating Directories

Directories are created from the File Manager screen by typing a name and optional description and using **ZOOM-IN**. A menu appears on the display requesting that the user select the type of file or directory desired. Position the cursor on the line on which Create a directory is listed and use **EXECUTE**. A blank File Manager screen for the new directory appears on the display.

A subdirectory is created from the File Manager screen of the directory in which the new subdirectory is to be located. The

appropriate directory must be accessed before creating the new sub-directory. To access a different directory, use the procedure described in §3.9.

3.9 Accessing Directories

The procedure for accessing a directory depends on the current directory location and the location of the directory to be accessed:

1. Directories listed on the File Manager screen of the home directory are accessed by positioning the cursor in any field on the line on which the directory is listed and using **ZOOM-IN**.
2. Directories that are located in a directory branching off of the home directory are accessed by using **ZOOM-IN** at each successive directory level until the desired directory is accessed. For example, to access the directory `$HOME/client/reports` from the File Manager screen of the home directory, **ZOOM-IN** on `client`, then **ZOOM-IN** on `reports`.
3. Directories located in other directories that do not branch off of the home directory are accessed by using **ZOOM-OUT** until the parent directory for the desired directory is accessed, and then by using **ZOOM-IN** at each successive level until the desired directory is accessed. Alternatively, directories are accessed by using **ENTER** *pathname* **USE**.
4. The home directory is accessed from any other directory by using **MENU** and selecting Show home directory.

3.10 Copying Directories

Directories are copied from the File Manager screen into the current directory or another directory by positioning the cursor in any field on the line on which the directory is listed and using **PICK-COPY**. The cursor will move down one line. Position the cursor where the copy is to be placed and use **PUT-DOWN**. An instruction box with the message `Filename "pathname"` already exists. Press **CANCEL** to abort file restore, or enter a new name: appears on the display. Type the new directory name at the cursor position and use **EXECUTE**. The directory is copied into the current directory and listed on the File Manager screen at the cursor position. You can also copy multiple directories using **BOX-MARK** or **ENTER** *vertical-motion* with **PICK-COPY** and **PUT-DOWN**.

To copy a directory into another directory, access that directory using the procedure described in §3.9 before using **PUT-DOWN**. In this case, the path name usually will not be duplicated so the instruction box will not display.

3.11 Moving Directories

Directories are moved by positioning the cursor in any field on the line on which the directory is listed and using **PICK-UP**. The directory is removed from the directory listing and the files listed below it move up one line. Move the cursor to the location where the directory is to be moved and use **PUT-DOWN**. You can also copy multiple directories using **BOX-MARK** or **ENTER** *vertical-motion* with **PICK-COPY** and **PUT-DOWN**. If the directory is to be moved to another directory, change to that directory using the procedures described in §3.9 before using **PUT-DOWN**. The directory is moved into the current directory and listed on the File Manager screen at the cursor position.

3.12 Renaming Directories

Directories are renamed from the File Manager screen by positioning the cursor at the appropriate character in the directory name and using any of the editing functions for modifying text. For example, change into overwrite mode by using **INSERT-MODE**, type over the old directory name, and **DELETE-CHARACTER** any extra characters.

3.13 Removing Directories

Directories are removed from the File Manager screen by positioning the cursor in any field on the line on which the directory is listed and using **DELETE**. The directory, and the entire structure (files and directories) below it, is removed from the File Manager screen and stored in `$HOME/.putdir`. You can also remove multiple directories using **BOX-MARK** or **ENTER** *vertical-motion* with **DELETE**. To restore a directory from `$HOME/.putdir`, use **PICK-COPY**, access the desired directory, then use **PUT-DOWN**.

3.14 Using LOCAL-MENU With the File Manager

LOCAL-MENU is used with the File Manager to display “hidden” files, such as `$HOME/.putdir`. It is also used to return to the normal display (without “hidden” files), and to display and edit additional file attributes, such as file permissions and ownerships.

4. THE HISTORY DISPLAY

This section explains how to use the History Display to recreate a previous version of a structured file.

4.1 Accessing the History Display

The History Display is accessed while editing a file by using **MENU** and selecting Show history of current file. This causes the History Display form to appear on the display. The History Display form shows the name of the user who edited the file; the date and time the editing started; and the number of lines or records that were inserted, deleted, or changed.

4.2 Accessing the Versions of a File

The different versions of a file are accessed from the History Display form by positioning the cursor in any field on the line on which the version is listed and using **ZOOM-IN**. The selected version of the file, as it existed after the changes indicated on the form, is displayed. **ZOOM-OUT** to return to the History Display form.

4.3 Using LOCAL-MENU with the History Display

LOCAL-MENU is used while viewing a previous version of a file to print the time of the version in a popup box, show the next version, show the previous version, redisplay the history form, or save the version.

4.4 Saving a Version of a File

A version of a file is saved by accessing that version from the History Display form and selecting **LOCAL-MENU** option (5) save current version of file. This causes an instruction box with the message Enter file name (*filename*): to appear on the display.

To save the version under a different name, enter the name and **EXECUTE**. To save the version under the original name (and move the current version to *filename.bak*), **EXECUTE** without entering a new file name. **ZOOM-OUT** twice to return to the original file.

5. CURSOR- AND WINDOW-POSITIONING FUNCTIONS

5.1 Cursor-Positioning Functions

There are eleven cursor-positioning functions used with the INEd editor:



These functions move the cursor one space at a time in the indicated direction. When any of these functions are used and the cursor reaches the screen boundary, the cursor wraps around to the opposite border. If the cursor is positioned on top of a border character, any typing causes the terminal to beep.

HOME

This function moves the cursor to the upper left-hand corner of the window. If multiple windows are displayed, it moves the cursor to the upper left-hand corner of the window in which the cursor is currently positioned.

LINE-FEED

This function positions the cursor at the beginning of the next line of a form.

TAB **-TAB**

These functions move the cursor to the next tab stop to the right (**TAB**) or left (**-TAB**) of the current cursor position. When the cursor is to the right of the rightmost tab stop in the field, **TAB** moves the cursor to the left to the first column of the window. When the cursor is in the leftmost column of the window, **-TAB** moves the cursor to the last column of the window. If the next tab stop is beyond the window boundary, the window scrolls appropriately to the left or right. If a field contains no tab stops, **TAB** moves the cursor to the next field or window, and **-TAB** moves the cursor to the previous field or window. Default tab stops are located eight columns apart, beginning with the first column of text. Tab stops are cleared and set with **SET-TAB** (§7.15).

ENTER

This function moves the cursor to the first nonblank character of the next line. If the next line is blank, the cursor is placed immediately below the first nonblank character in the

previous line. If the cursor is on the last line of the window, the window scrolls one line before the cursor moves to the next line. This allows text to be entered continuously at the end of a file without having to reposition the window to bring in new blank lines.

BEGIN-LINE **END-LINE**

These functions move the cursor to the beginning (**BEGIN-LINE**) or end (**END-LINE**) of the current line. **BEGIN-LINE** moves the cursor to the first character on the line and **END-LINE** moves the cursor one space to the right of the last character on the line.

5.2 Window-Positioning Functions

There are seven window-positioning functions used with the INEd editor:

+LINE **-LINE**

These functions move text one-third of a window forward (**+LINE**) or backward (**-LINE**). The cursor remains on the same line unless the line it is on scrolls outside of the window. If **+LINE** is used and the cursor line scrolls outside of the window, the cursor moves to the top of the window. If **-LINE** is used and the cursor line scrolls outside of the window, the cursor moves to the bottom of the window. **ENTER** **+LINE** moves the window so the line containing the cursor is the top line in the window. **ENTER** **-LINE** moves the window so the line containing the cursor is the bottom line in the window.

ENTER *n* **+LINE** moves the window *n* lines forward, and **ENTER** *n* **-LINE** moves the window *n* lines backward.

+PAGE **-PAGE**

These functions move the text one window forward (**+PAGE**) or backward (**-PAGE**). The cursor remains in the same column on the same line (relative to the top and bottom borders) in the window. **ENTER** *n* **+PAGE** moves the window *n* pages forward, and **ENTER** *n* **-PAGE** moves the window *n* pages backward.

LEFT **RIGHT**

These functions move the window in the indicated direction, displaying text previously to the left (**LEFT**) or right (**RIGHT**) of the window. The cursor remains in the same column unless the column it is in scrolls outside of the window. If **LEFT** is used and the cursor column scrolls outside of the window, the cursor moves to the right-hand margin. If **RIGHT** is used and the cursor column scrolls outside of the window, the cursor moves to the left-hand margin. **ENTER** *n* **RIGHT** moves the window *n* columns to the right, and **ENTER** *n* **LEFT** moves the window *n* columns to the left.

GO-TO

This function moves the window to the first page of the file and the cursor to the first line. If the cursor is at the first line in the file, the window moves to the last page of the file and the cursor moves to the last line. **ENTER** **GO-TO** moves the cursor to the last line of the file. **ENTER** *n* **GO-TO** moves the cursor to line *n*.

6. TEN/PLUS SYSTEM FUNCTIONS

This section describes functions that assist in controlling TEN/PLUS functions and simplify system operations performed within the TEN/PLUS environment. It includes procedures for interrupting search or filter operations; displaying information about error messages, functions, and operations; using menus to simplify system operations and select data-specific functions; controlling detail while viewing structured files; printing documents; moving sequentially through the items in a structured file; creating and editing alternate files; and saving files without exiting the editor.

6.1 Canceling Operations: **BREAK**

BREAK interrupts a search or filter operation (§7.9, §9), leaving the window position and the file unchanged. **BREAK** works only if the canceled operation has not completed. **BREAK** will not undo the operation if the operation completes prior to using **BREAK**.

6.2 Error Message and Other Explanations: **HELP**

HELP provides information about error messages, functions, and operations. **HELP** can be used from anywhere within the TEN/PLUS environment. If **HELP** is used and there are no error messages or popup boxes in the window, the Help Menu displays. The Help Menu is used to provide additional information about editor functions, keyboard layouts, TEN/PLUS operations, customizing the editing system, and customizing **MENU**. If **HELP** is used while a popup box is in the window, additional information about the information in the box appears on the display.

After you have selected one of the options from the menu, **USE** to continue editing.

6.3 Accessing the Operating System and Other Files: **MENU**

MENU is used to simplify certain operations and access the operating system and other files without having to exit the TEN/PLUS environment. It allows the user to select an operation from a list of options, which can be customized by altering the editor profile. The procedure for customizing the New Task Menu is described in “TEN/PLUS Profiles.”

Although the New Task Menu is installation-dependent, it usually has options for displaying the home directory, executing UNIX shell commands, running a shell command in a box, showing the profiles directory, editing the editor profile, and displaying the history of the

current file. Options can be added for accessing other files or directories, running a specific program and displaying the output in a popup box, and clearing the screen to run specific programs. Most of the menu operations require a series of steps if performed without the use of **MENU**.

The cursor-positioning functions are used to position the cursor at the desired option. Once the desired item is selected, use **EXECUTE** to select the option and perform the indicated operation; **CANCEL** to remove the menu without executing the selected operation; or **HELP** to provide additional information about using **MENU**. You can also use any of the functions **(1)** through **(8)**, depending on the number that corresponds to the line on which your choice is listed. For example, to select `Show your profiles directory` from the default New Task Menu, use **(4)**, since this is the fourth option on the menu.

If **MENU** is accidentally used to run an interactive command (one requiring additional input from the user), **BREAK** can sometimes be used to interrupt the process.

6.4 Removing Boxes: **CANCEL**

Four types of boxes can appear in the window during an editing session: menus, error messages, instructions, and informative messages. The last type disappears after a few seconds, but the first three must be removed explicitly with **CANCEL**.

Occasionally, the system needs additional information from the user before a command can be executed. In these situations, **EXECUTE** is used to “send” the requested information to the system, and **CANCEL** is used to cancel the operation and return to the editing session.

6.5 Controlling Detail: **ZOOM-IN** and **ZOOM-OUT**

There are certain situations in which it is desirable to “focus in” on a specific item displayed on the screen. For example, while viewing a structured file (§2.2), such as the editor profile file, it is often necessary to see additional data related to a specific item in the window. Additional detail is displayed by moving the cursor to the item and using **ZOOM-IN**. **ZOOM-OUT** reverses the process.

6.6 Printing a Document: **PRINT**

PRINT displays a menu containing options for printing a file. The standard Print Menu includes options for printing on a default

printer, printing with options, overwriting the output to a file, and appending the output to a file. Additional options can be included by modifying `$HOME/profiles/printprf`, the print profile file. The procedure for modifying the print profile file is described in detail in “TEN/PLUS Profiles.”

6.7 Selecting Data-Specific Functions: LOCAL-MENU

While editing certain types of structured data, a menu of additional capabilities specific to the particular data is provided. `LOCAL-MENU` displays this menu. `LOCAL-MENU` can be used in exactly the same way as the Help Menu, the New Task Menu, and the Print Menu (§6.2, §6.3, §6.6).

6.8 Looking at Other Items: NEXT and PREVIOUS

Structured files often contain items that can be viewed sequentially by using `ZOOM-IN`, viewing the item, then using `ZOOM-OUT`. It is possible to access the next item in the sequence without using `ZOOM-OUT` by using `NEXT`. It is also possible to access the previous item in the sequence without using `ZOOM-OUT` by using `PREVIOUS`. Both `NEXT` and `PREVIOUS` accept numeric arguments indicating which item in the sequence to view (counting from 0). An example of a file containing items that can be viewed sequentially is the editor profile, which contains `MENU Options`, `HELP Options`, and so on.

6.9 Editing and Creating Alternate Files: USE

`ENTER filename USE` reads in *filename* as the current file and saves the previous file as the alternate file. If a file with the specified name does not exist, a menu displays, requesting that the user select the type of file to be created. Possible file types are described in §2.2. `CANCEL` returns the editor to its previous state; a new file is not created, and the window and alternate file status are not changed. `USE` without an argument switches the current and alternate files. The cursor position of the alternate file is restored to its previous position and the current file becomes the new alternate file. `ENTER USE` is similar to `ENTER filename USE`, but takes the file name argument from the text in the window, starting at the current cursor position and ending with the first blank.

The current file is both the current and alternate file if that file name is used as the argument to `ENTER filename USE`. `USE` can then be used to switch back and forth between two different positions in the same file without losing the cursor positions. This is

useful when two different sections of the same file are edited, or when you wish to move lines from one part of a file to another.

6.10 Saving Files Without Exiting the Editor: SAVE

For ASCII files, **SAVE** writes an edited file onto disk. For structured files, **SAVE** writes the current record to disk. The current state of the edited file or record is not changed, and the editing session can continue. The edited file or record is saved again upon exiting the TEN/PLUS environment if it is modified after the **SAVE**. When editing ASCII files or structured nontext files with large records, it is advisable to **SAVE** periodically.

ENTER filename SAVE saves the current image of the file being edited and then copies the file into *filename*. The editor renames the overwritten file to include a `.bak` suffix.

7. INTERACTIVE TEXT EDITING FUNCTIONS

The INed command language consists of cursor movements, functions, and arguments. The first element of the INed command language is the cursor position. The cursor is a pointer to where the next typed character will appear. Most editing functions are performed at the position or line indicated by the cursor. Cursor-positioning functions are described in §5.1.

Functions perform designated operations and are the second element of the INed command language. For example, **DELETE** deletes the line of text at the cursor position. Editing functions are described in this section and text processing functions are described in §9.

Arguments, described in §8, are the third element of the INed command language. Arguments modify function defaults.

This section explains the various INed editing functions. The effect of each function is summarized in Appendix A. **DO**, **FORMAT**, **MARGIN**, **CENTER**, and **FONT**, which are used for interactive text processing, are covered separately in §9.

7.1 Inserting Characters: INSERT-MODE

The INed editor has two modes: *overwrite* and *insert*. The mode affects all printing characters (including the space character) as well as the action of **BACKSPACE** (§7.4).

In overwrite mode, a typed character is placed at the cursor position, overwriting any previously existing character. In insert mode, a typed character is inserted at the cursor position, and the characters at and to the right of that position move one position to the right; no characters are deleted. If the line extends beyond the right-hand margin, the last word on the line wraps to the next line (§9.4).

In overwrite mode, **INSERT-MODE** changes the mode to insert; in insert mode, **INSERT-MODE** changes the mode to overwrite.

7.2 Inserting Lines and Areas, Splitting Lines: INSERT

INSERT inserts a blank line at the cursor line, and moves the cursor line and the lines below it down one line. For all variations of **INSERT**, the cursor always returns to the position it occupied before **INSERT** was used.

ENTER *n* **INSERT**, where *n* is a positive numeric argument, inserts *n* blank lines at the cursor line and moves the cursor line and the lines below it down *n* lines.

ENTER **INSERT** splits the current line by moving the characters at and to the right of the cursor position to a new next line, and all subsequent lines down one line.

INSERT can also be used in conjunction with a cursor-defined argument (§8.2) to open either a specific number of lines or a block. If lines are indicated, **ENTER** *motion* **INSERT** opens the number of lines defined, where *motion* indicates a cursor-positioning function. If an area is indicated, blank spaces are inserted and the remainder of each line on which the blanks are inserted is moved to the right. No characters are deleted; however, this sometimes causes the remainder of the line to move beyond the right-hand window border.

7.3 Moving Lines and Areas, Joining Lines: PICK-UP

PICK-UP picks up the current line and adds it to the *pick* buffer. For all variations of **PICK-UP**, the cursor always returns to the position it occupied before **PICK-UP** was used and all subsequent lines in the file move up one line.

The *pick* buffer contains lines or areas of text from **PICK-UP** or **PICK-COPY** (§7.7). The text in the *pick* buffer can be inserted anywhere in the text by using **PUT-COPY** or **PUT-DOWN** (§7.8).

ENTER **PICK-UP** picks up the current line from the current cursor position to the end of the line and replaces it with the text on the next line. In effect, it joins the text on the current line to the left of the cursor with the next line. The effect of **ENTER** **INSERT** can be canceled by issuing **ENTER** **PICK-UP** immediately thereafter.

ENTER *n* **PICK-UP**, where *n* is a positive numeric argument, removes *n* lines, beginning with the current line. The lines are placed in the *pick* buffer and all subsequent lines are moved up *n* lines. If the specified number of lines extends beyond the end of the file, a sufficient number of blank lines is supplied to provide a total of *n* lines in the *pick* buffer.

PICK-UP can also be used in conjunction with a cursor-defined argument (§8.2) to remove either a specific number of lines or a block. If lines of text are designated, the effect is the same as **ENTER** *n* **PICK-UP**. If an area is defined, the designated portion

of each line is picked up and the characters to the right of the area move to the left.

7.4 Backspace: BACKSPACE

BACKSPACE moves the cursor to the left. In overwrite mode, it moves the cursor to the left by one position, replacing any existing character with a blank character. In insert mode, it moves the cursor to the left by one position, deleting the character at that position and moving all characters to the right of the deleted character one position to the left. Characters deleted with **BACKSPACE** are not saved and must be retyped to be recovered.

7.5 Deleting Characters: DELETE-CHARACTER

DELETE-CHARACTER deletes the character at the current cursor position and moves all characters to the right of the deleted character one position to the left. The cursor does not move. Characters deleted with **DELETE-CHARACTER** are not saved and must be retyped to be recovered.

7.6 Deleting the End of a Line: ENTER DELETE-CHARACTER

ENTER **DELETE-CHARACTER** deletes the current line from the cursor position to the end of the line. The deleted text is added to the wastebasket buffer (§7.19) and can be retrieved with **RESTORE**.

7.7 Duplicating Lines or Areas: PICK-COPY

PICK-COPY places a copy of the current line in the pick buffer and moves the cursor down one line. **PICK-COPY** is similar in function to **PICK-UP**, except that the lines or regions are copied to the pick buffer and are not removed from the file. **PICK-COPY** does not alter the file being edited.

ENTER n **PICK-COPY**, where n is a positive numeric argument, places n lines, beginning with the current line, in the pick buffer and moves the cursor down n lines. If the specified number of lines extends beyond the end of the file, a sufficient number of blank lines is supplied to provide a total of n lines to the buffer.

PICK-COPY can also be used in conjunction with a cursor-defined argument (§8.2) to copy either a specific number of lines or a block into the pick buffer. If lines are indicated, the effect is the same as **ENTER** n **PICK-COPY**, except that the cursor does not move. If an area is defined, it is placed in the buffer as a group of partial

lines and can be inserted within lines by using **PUT-COPY** or **PUT-DOWN** (§7.8). Any text to the right of the insertion is moved to the right.

7.8 Placing Text in a File: PUT-COPY and PUT-DOWN

PUT-COPY places a copy of the newest contents of the pick buffer at the cursor position, leaving the buffer unaffected. **PUT-DOWN** moves the newest contents of the pick buffer to the current cursor position, removing it from the buffer. If a numeric argument to either function is specified with **ENTER**, the specified number of copies of the newest data in the buffer is inserted at the cursor location.

Text is inserted into the file in one of two ways, depending on whether a set of lines or an area of text is inserted. Lines of text inserted with **PUT-COPY** or **PUT-DOWN** are inserted at the current line and all subsequent lines are moved down to make room for the inserted lines. Areas of text inserted with **PUT-COPY** or **PUT-DOWN** are inserted with the upper left-hand corner of the area at the current cursor position. The portions of the affected lines to the right of and including the cursor column are moved to the right to make room for the inserted text. This sometimes causes the affected lines to move beyond the right-hand window border.

PUT-COPY and **PUT-DOWN** can be used with **PICK-COPY** or **PICK-UP** to move text from one file to another. The contents of the buffer are unchanged when windows or current files are switched. Refer to **USE** and **NEXT-WINDOW** (§6.10, §7.14) for additional information about moving between files or windows.

7.9 Searching for a Text String: +SEARCH and -SEARCH

ENTER *string* **+SEARCH**, where *string* is any sequence of printing characters, searches forward in the current file for the next occurrence of *string*. The search begins at the character immediately after the cursor. If the string is found, the cursor is moved to the first character in that occurrence of the string. If the string is in the window in which the search is initiated, the cursor moves directly to the string. If the string is not in the window where the search is initiated, the window advances in the file to bring the line containing the search string to a position several lines down from the top of the window. If the string is not found between the cursor position and the end of the file, the message `search failed on string "string"` appears in a box, and the window and cursor

positions remain unchanged. When a search string is located in a column past the right-hand boundary of the editing window, the window moves to the right to display the string.

+SEARCH without an argument searches for the previous search string. An error message displays if a search has not yet been initiated during the editing session. **ENTER** **+SEARCH** uses the text at the cursor position up to the first blank space or the end of the line as the search string.

-SEARCH functions as **+SEARCH** does, but **-SEARCH** searches backward in the file, starting at the character immediately preceding the cursor.

7.10 Canceling a Search: BREAK

A search can be canceled with **BREAK** (§6.1). This causes the message **Stopped by BREAK** to appear in a box on the display. The cursor and window positions are not changed when **BREAK** is used.

7.11 Searching and Replacing: REPLACE

ENTER *string* **REPLACE**, where *string* is any sequence of printing characters, places *string* in the *replace* buffer and compares the string at the cursor position to the search string. If they are identical, the string at the cursor position is replaced by *string*. If the string at the cursor position does not match the search string, no action is taken. **ENTER** **REPLACE** deletes the string at the cursor position if it matches the current search string. This puts a null string in the replace buffer.

Once a string has been entered in the replace buffer, **REPLACE** is used with **+SEARCH** and **-SEARCH** to replace selected instances of the search string. This is accomplished by first using **+SEARCH** or **-SEARCH** until a string to be replaced is located, and then using **REPLACE**. Refer to §9.2 for additional information about replacing search strings.

7.12 Moving to a Specified Line: GO-TO

ENTER *n* **GO-TO**, where *n* is a positive numeric argument, scrolls the window so that it displays line number *n* several lines down from the top of the window. If line *n* is already positioned in the current window, the cursor is positioned on the designated line and the window position remains unchanged.

GO-TO without an argument moves the window to the beginning of the file. It is equivalent to **ENTER** 1 **GO-TO**. If the cursor is positioned at the beginning of the file, **GO-TO** has the same effect as **ENTER** **GO-TO**, which scrolls the window to the end of the file and displays the last line several lines down from the top of the window.

7.13 Creating More Than One Window: **WINDOW**

When a file is opened, the screen contains one large editing window. This window can be divided into several smaller windows, each containing its own file, and each with the window and cursor at the desired positions in the file.

A new window is created by dividing an existing window into two smaller windows. If the cursor is on the first line in a window, that window is divided vertically at the cursor position. Otherwise, the window is divided horizontally at the cursor position.

To divide a window, position the cursor in the window to be divided and use **ENTER** *filename* **WINDOW**, where *filename* is the name of the file to be contained in the window. A horizontal division creates a new window below the new boundary. A vertical division creates a new window to the right of the new boundary.

When **WINDOW** is used, the new window becomes the current window and displays the file specified as the argument. **WINDOW** without an argument creates a new window containing the beginning portion of the file in the current window. The remainder of the old window continues to display its current file. It is reactivated by using **NEXT-WINDOW** (§7.14).

The active window during an editing session is called the “current window.” The cursor is positioned in the current window. **NEXT-WINDOW** activates another window as the current window.

Windows are deleted by using **ENTER** **WINDOW**. The window containing the cursor expands to occupy the entire screen.

7.14 Changing Windows: **NEXT-WINDOW**

NEXT-WINDOW is used when the screen contains multiple editing windows. It activates the next window, in order of creation, as the current editing window. **ENTER** **NEXT-WINDOW** activates the next window in the inverse order of creation.

An alternate way of changing windows is to move the cursor “through” the window boundary into an adjacent window. With this approach, however, the window change does not take effect until an editing function is used.

7.15 Setting and Clearing Tabs: SET-TAB

Tabs are initially set to the system default, which is every eighth column in the file. Changes to tab settings are valid only for the current editing session. Each file used in an editing session has its own tab stops.

SET-TAB sets a tab stop at the cursor column. **ENTER SET-TAB** clears any tab stops at the cursor column.

7.16 Editing Wide Lines: RIGHT and LEFT

RIGHT and **LEFT** are used when editing lines that are wider than the window. **RIGHT** moves the window to the right one-third of the window or to the right-hand margin of the file, whichever is closer. **LEFT** moves the window to the left one-third of the window, or to the left-hand margin of the file, whichever is closer. The cursor remains on the line it occupied prior to the move. If the original cursor column is still on the display, the cursor remains there. If the column moves off the display, the cursor moves to the right-hand border if **LEFT** is used, or to the left-hand border if **RIGHT** is used.

RIGHT and **LEFT** can also be used in conjunction with **ENTER** to indicate a specific number of columns for the window to move. **ENTER n LEFT** moves the window n columns to the left, and **ENTER n RIGHT** moves the cursor n columns to the right.

7.17 Inserting Control Characters: QUOTE

Control characters are nonprinting characters, some of which perform formatting functions in text files. **QUOTE** allows the insertion of control characters into text. To insert control characters into a file, use **QUOTE**, then type the printing character that corresponds to the desired control character. For example, to insert a `Ctrl-a` at the cursor position, use **QUOTE** a. Some control characters are displayed as special graphics characters. To enter several control characters, use the graphics font (§8.11).

7.18 Redrawing the Display: REFRESH

REFRESH erases, then redraws the display. This is useful when a system message appears on the screen during an editing session.

7.19 Deleting and Restoring: DELETE and RESTORE, # File

Organizing lines in a file typically involves moving, copying, and deleting operations. The first two are managed with **PICK-UP**, **PUT-DOWN**, **PICK-COPY**, and **PUT-COPY**. Deletion is handled with **DELETE**.

Lines deleted with **DELETE** are added to the *wastebasket* buffer. Deleted lines in the wastebasket buffer can be restored with **RESTORE**. **RESTORE** removes the last deleted line or area from the wastebasket buffer and places it in the file at the current cursor position. A numeric argument can be used with **RESTORE** to indicate the number of copies of the last deleted line or area to be inserted at the cursor location.

In situations where text deleted early in a session must be recovered, use **RESTORE** until the missing text is moved from the buffer to the display. Or, if the file is a structured file, the previous version can be retrieved using the History Display (§2.2, §4.4).

All text deleted during an editing session is saved in a file named #. Although this file cannot be altered, it can be used to retrieve deleted text by using **ENTER** # **USE**, then using **PICK-COPY** to recover the deleted text.

7.20 Using FUNCTIONS

The ten basic functions included with the TEN/PLUS system are **MENU**, **LOCAL-MENU**, **INSERT**, **PICK-COPY**, **PUT-COPY**, **PICK-UP**, **PUT-DOWN**, **FORMAT**, **ZOOM-IN**, and **ZOOM-OUT**. Each of these functions can only be used in certain situations and, consequently, only a subset of these functions may be active at any given time. **FUNCTIONS** displays the set of functions that are currently active. (On some systems, more than ten functions may be displayed.) For example, when editing a File Manager display, **FORMAT** is not displayed by **FUNCTIONS** because you cannot use **FORMAT** while viewing a File Manager display.

The complete Functions Menu typically looks like this:

Functions Menu

Move the cursor to the desired function and touch EXECUTE. Touch CANCEL to do nothing, HELP for help.

MENU
LOCAL-MENU
FORMAT
INSERT
PICK-UP
PUT-DOWN
PICK-COPY
PUT-COPY
ZOOM-IN
ZOOM-OUT

FUNCTIONS options are selected in the same way any menu options are selected. To remove the menu, use **CANCEL**.

8. ARGUMENTS TO FUNCTIONS

Functions, when used alone, pass simple requests to the editor. Modifying a function with an argument permits a greater range of function capabilities. The process of entering an argument can be canceled by using **CANCEL** or another **ENTER**.

8.1 Typed Arguments

Most functions accept arguments. The argument can be a number, a string, or a cursor-positioning sequence. If the argument contains the name of an exported shell variable, the variable is expanded before the operation is performed.

To use a function with an argument, use **ENTER**, the argument, and then the function. After **ENTER** is issued, a popup box into which the argument should be typed displays on the screen. If the popup box is too small, it automatically scrolls to fit the entire argument. Issuing the appropriate function initiates the operation.

Some functions, like **+PAGE** and **LEFT**, accept only numeric arguments. Others, like **+SEARCH** and **WINDOW**, accept any string that does not contain cursor-positioning functions. If a cursor-positioning function is issued as an argument to **ENTER**, the argument is interpreted as a cursor-defined argument (§8.2). If an invalid argument is given, the editor writes an error message into a popup box on the screen. The table in Appendix A indicates the type of argument accepted by each function, and gives the interpretation for each.

You can use any of these functions to edit the text typed into a popup box:

←, →
BACKSPACE
BEGIN-LINE, **END-LINE**
DELETE-CHARACTER
INSERT-MODE

To cancel the argument entirely, use **ENTER** or **CANCEL**.

LAST-ARG displays the argument last typed in the **ENTER** popup box. Type additional characters or use any of the functions listed above to modify the argument.

8.2 Cursor-Defined Arguments

Cursor-positioning functions are used as arguments to **ENTER** to define a set of lines or characters, or a block on the display, to be manipulated. Cursor-defined arguments are used in conjunction with **INSERT**, **DELETE**, **PICK-UP**, **PICK-COPY**, **MARGIN**, and **CENTER** (§7.2, §7.19, §7.3, §7.7, §9.4, §9.5) to designate groups of lines or specific portions of lines for editing. Cursor-defined arguments can also be initiated by using **BOX-MARK** or **TEXT-MARK** instead of **ENTER**.

8.2.1 Designating Lines

To use a cursor-defined argument to delete lines, position the cursor anywhere on the first line to be deleted, then use **ENTER** to begin defining the argument. Position the cursor on the last line to be deleted, then use **DELETE** to delete the lines from the starting cursor position to the ending cursor position. To create one or more lines of blank space, move lines, or copy lines, use this same procedure, substituting **INSERT**, **PICK-UP**, or **PICK-COPY**, respectively, for **DELETE**.

The message ******BOX/LINE****** appears at the bottom of the screen when cursor-defining is in progress. The text defined by a vertical cursor motion includes the line from the initial cursor position up to and including the last line on which the cursor is positioned. On some displays, the defined lines are highlighted.

Use **+PAGE**, **-PAGE**, **+LINE**, and **-LINE** (§5.2) with the cursor-positioning functions after using **ENTER** to define a set of lines spanning more than one screenful of text. When indicating a cursor-defined argument, **+LINE** and **-LINE** cannot immediately follow **ENTER** because this sequence has another interpretation (§5.2). **+LINE** and **-LINE** can be used after **+PAGE**, **-PAGE**, or a cursor-positioning function has been issued.

8.2.2 Designating Blocks

A cursor-defined block is a portion of the window defined on the screen using the cursor-positioning functions as the argument to **BOX-MARK**. To designate a block of text, use **INSERT**, **DELETE**, **PICK-UP**, or **PICK-COPY** (§7.2, §7.19, §7.3, §7.7) along with the cursor-positioning functions. The cursor can be moved vertically and horizontally to define a block. Moving the cursor horizontally across columns defines an area of text that can be opened, deleted, moved, or picked up. If the area is opened, or if

a picked-up area is put down, the lines on which the text is inserted open horizontally to make room for the new material; the remainder of each line is moved to the right.

The portion of the screen defined by vertical and horizontal cursor motion includes all of the lines between the initial and the final cursor positions, including the initial and final lines. The columns from and including the leftmost position, up to but *not* including the rightmost position, are also included.

Use **+PAGE**, **-PAGE**, **+LINE**, and **-LINE**, to indicate arguments spanning more than one window of text. In addition, **LEFT** and **RIGHT** move the window to define arguments extending beyond the horizontal window of text. **BEGIN-LINE** or **END-LINE** can be used after **ENTER** to move the cursor to the beginning or end of the current line, respectively. **ENTER +PAGE** and **ENTER -PAGE** move the window forward or backward one page, respectively, and cursor-define the lines of text from the beginning cursor position to the ending cursor position. This sequence is commonly used to begin defining a block (for moving, copying, inserting, or deleting) that extends beyond a page boundary.

An area can be defined in any direction. The only requirements are that a group of lines be defined by indicating the first and last line in either direction, and that an area of text be defined by indicating two opposite corners in any order.

8.2.3 Designating Running Text Regions

TEXT-MARK, used to define a running sequence of text, is most often used to delete, move, or copy sentences. For example, to delete a sentence, (1) position the cursor on the first character of the sentence to be deleted; (2) use **TEXT-MARK** to begin defining the region; (3) position the cursor directly on the first character *not* to be deleted (for example, on the first character of the next sentence); and (4) use **DELETE**.

INSERT, **PICK-UP**, or **PICK-COPY** can also be used with **TEXT-MARK** by substituting **INSERT**, **PICK-UP**, or **PICK-COPY**, respectively, for **DELETE**.

When **TEXT-MARK** is used, the message *******TEXT******* appears at the bottom of the screen. On some displays, moving the cursor highlights the selected region. The sequence can be terminated with **ENTER** or **CANCEL**.

Use **+PAGE**, **-PAGE**, **+LINE**, and **-LINE** to indicate arguments spanning more than one window. The cursor-positioning functions can be used to define partial lines, and **BEGIN-LINE** and **END-LINE** can be used after **TEXT-MARK** to move the cursor to the beginning or end of the current line, respectively.

9. TEXT PROCESSING FUNCTIONS

Programs that perform text processing operations, such as string replacement, paragraph fill and justification, indenting, and multiple spacing, can be run from within the editor by using a *filter*. A filter is a program that performs a predefined operation on text. The requested operation is performed on the indicated paragraphs or lines and the results are usually indicated on the screen.

Some filters are invoked using the sequence **ENTER** *argument* **DO**. This sequence passes the requested lines to the specified program and replaces those lines with the result of the program's execution. The old lines are saved and can be accessed with **RESTORE**. Other filters are invoked by simply using a function, such as **FORMAT** or **CENTER** (§9.3, §9.5).

This section describes the text processing filters that can be used with the editor. Possible filter arguments, and their effects, are explained in detail in §9.1. The most useful text processing filters are described in §9.2. Additional text processing capabilities are described in §9.3 through §9.6. Canceling a filter operation is described in §9.7. Examples of how to use text processing programs, or filters, with **DO** are provided in §9.8.

Interactive programs (those that request input from the user) should not be run using **ENTER** *argument* **DO**. Programs started this way can be stopped with **BREAK**.

9.1 Using Arguments: ENTER argument DO

ENTER *argument* **DO** is unique because it can take more than one argument. If more than one argument is specified, each argument is separated by one or more spaces. Any environment variables in the argument string are expanded before use (see *sh(1)* for an explanation of environment variables).

Possible arguments for **ENTER** *argument* **DO** are:

1. An optional first argument specifying the number of lines (an integer followed by the letter *l*) or paragraphs (a positive integer) to be processed. If this argument is omitted, the default is one paragraph. The first line processed is always the current cursor line. If lines beyond the end of the file are requested, only lines up to the end of the file are used. Thus, a large argument (for example, 9999) usually indicates the rest of the file, beginning with the current line.

2. The name of the program (filter) to be executed. This argument is required.
3. Arguments to the filter. These arguments vary according to the filter used. An argument containing one or more spaces must be enclosed in paired single or double quotes (' ' or '"').

Once an argument has been specified, **DO** can be used without **ENTER** or an argument to run the previous filter operation again.

9.2 Replacing Strings: **rpl**

The **rpl** program (see *rpl(1)* for details) requires two arguments. It replaces all occurrences of the first argument string with the second argument string. If no argument precedes **rpl**, the replacement starts at the cursor line and continues through to the end of the paragraph. Paired single or double quotes are required for arguments containing spaces.

The **rpl** program recognizes regular expressions in its first argument (see *ed(1)* for a full explanation of regular expressions). To delete a string using **rpl**, replace it with the null string (""). To delete all occurrences of the words **Computers** and **computers** in the next 50 lines of text, use the sequence:

```
ENTER 501 rpl "[Cc]omputers" "" DO
```

9.3 Formatting Text: **FORMAT**

FORMAT uses a fast internal algorithm for formatting the paragraph containing the cursor. Text is filled according to the margins and is not justified. The lines are stored in the wastebasket buffer and can be restored to their preformatted state with **RESTORE**.

9.4 Setting Margins: **MARGIN**

Margins are indicated on the screen by the characters **l** and **r** above the top border of the window. Margins are reset using **MARGIN**, **ENTER MARGIN**, or **ENTER horizontal motion MARGIN**. **MARGIN** sets the left-hand margin at the current cursor column. **ENTER MARGIN** sets the right-hand margin at the current cursor column. **ENTER horizontal motion MARGIN** sets the left-hand margin to the left edge of the cursor-defined area and the right-hand margin to the right edge.

Word wrap automatically moves words extending beyond the right-hand margin to the left-hand margin of the next line. The contents of the next line move to the right or down one line before the word

is moved. Word wrap is enabled unless the right-hand margin is removed.

Changes in margin settings are effective only during the current editing session. If the file is exited and reentered, the default margins are restored.

9.5 Centering Text: CENTER

Text is centered using **CENTER**, **ENTER** *n* **CENTER**, or **ENTER** *vertical motion* **CENTER**. **CENTER** centers the current line between the margins. **ENTER** *n* **CENTER** centers *n* lines of text, beginning with the cursor line. **ENTER** *vertical motion* **CENTER** centers all lines within the cursor-defined area between the margins.

If margins are subsequently changed, centered lines are not automatically recentered.

9.6 Selecting Fonts: FONT

Each file has a current and alternate font associated with it. The default current font is Roman with no underline; the default alternate font is Roman with continuous underline.

FONT switches the editor between the current and alternate fonts. **ENTER** **FONT** saves the current font as the alternate font and sets the current font to Roman with no underline. **ENTER** *letter* **FONT** saves the current font as the alternate font and sets the current font as specified by *letter*. The acceptable values for *letter* are r (Roman), c (continuous underline), w (word underline), and g (graphics characters). The graphics character set is described in Appendix C.

ENTER ? **FONT** tells you which fonts are available and which font is currently in use.

9.7 Canceling a Filter Operation: BREAK

A filter operation can be interrupted with **BREAK** (§6.1). **BREAK** leaves the file and window positions unchanged. It does not undo the operation if it completes prior to using **BREAK**.

9.8 More DO Capabilities: tee, cat, sort, and date

Your system includes a variety of text processing programs suitable for use with **DO**. Refer to your user's manual for additional information about specific programs.

Any program suitable for use as a filter can be used with **DO**, including user-written programs. For example, to pick five lines from the current file and place them in *file1*, use **ENTER** 51 tee *file1* **DO**. The tee filter writes the five lines into *file1* without changing the lines in the current file.

Use **ENTER** 0 cat *file2* **DO** to write *file2* into the current file at the current cursor line. Use **ENTER** sort **DO** to sort the lines up to the next blank line in alphabetical order. Use **ENTER** 0 date **DO** to insert the current date and time at the current line.

10. RECOVERY

This section describes the TEN/PLUS procedures that prevent accidental destruction of information due to system failures or inadvertent user actions.

10.1 Old Versions of ASCII Files

A modified ASCII file is saved by exiting the editor or using **SAVE** (§6.10). The previous version is not deleted, but is renamed by truncating the file name if necessary and appending `.bak`. The previous `.bak` file is overwritten.

10.2 Old Versions of Structured Files

Previous versions of structured files are automatically saved within the file itself, and can be recalled through the History Display (§4). Specific versions can be saved either by exiting TEN/PLUS or using **SAVE**.

It is recommended that history be removed periodically using either `rmhist(1)` or the `Housekeep` option on the `New Task Menu`. After the history is removed, versions prior to the time that the history was removed will no longer be available.

10.3 Automatic Recovery

In situations such as power failure or storage exhaustion, the editor saves the current editor session. Text characters entered after the last editing command, however, are lost. The recovery is automatic, and in most situations the user does not need to take action.

If possible, the editor restarts itself. If the editor cannot restart, type `e` and press the **ENTER** key. The state the file was in prior to the session in which the problem occurred is restored.

11. MISCELLANEA

11.1 Text File Format

If a line is not explicitly modified during an editing session, it remains unchanged. If it is modified, certain changes are made. When files are read by the editor, existing ASCII tab characters are maintained in the text (even though they appear as spaces in the display) until a line is modified. When a line is modified, all tabs are converted into spaces. Then, when a file is saved, leading spaces are replaced with ASCII tab characters wherever possible (that is, each string of eight spaces is replaced by a single tab character). Spaces after the first nonblank character are not replaced; trailing blanks are removed. If the last line of a file is not terminated by the new-line character (octal 012), a new-line character is supplied when that line is edited. All control characters other than tab (`Ctrl-i`) and linefeed (`Ctrl-j`) are preserved in files edited by INed. `Ctrl-j` will cause the loss of the end-of-line and should be avoided.

11.2 TEN/PLUS Limitations

If there are any TEN/PLUS limitations specific to your system, an addendum has been provided that describes them.

11.3 The Editor Profile

It is possible to give the editor instructions on how to act in certain circumstances. This information is normally stored in the editor profile. Refer to “TEN/PLUS Profiles” for details on how to create and modify the editor profile file.

Appendix A: SUMMARY OF TEN/PLUS FUNCTIONS

FUNCTION	Action of FUNCTION	Action of ENTER FUNCTION	Action of ENTER <i>X</i> FUNCTION (<i>X</i> is any argument)	Action of ENTER <i>motion</i> FUNCTION (<i>motion</i> is a combination of cursor-positioning functions.)
(1)-(8)	Use the <i>n</i> th special function.	Data-dependent.	Data-dependent.	Data-dependent.
↑ → ↓ ←	Move the cursor one position in the indicated direction.	Enter BOX/LINE mode; define a region to the next cursor position.	ERROR (Note 1)	Continue defining argument.
BACKSPACE	Delete the character to the left of the cursor.	No effect.	Delete the last character of <i>X</i> .	ERROR (Note 2)
BEGIN-LINE	Move the cursor to the first non-space character of the current line.	Enter BOX/LINE mode; define a region to the first non-space character of the current line.	ERROR	Continue defining argument.
BOX-MARK	Enter BOX/LINE mode.	ERROR	ERROR	ERROR
BREAK	Interrupt a search or a filter.	No effect.	No effect.	No effect.
CANCEL	Remove an error message or popup box from the screen.	Removes box.	Removes box.	Removes box.
CENTER	Center current line between margins.	ERROR	<i>X</i> is positive. Center <i>X</i> lines.	Center all lines within the box or line region.
DELETE	Delete CCL (Note 3). Deleted text is available with RESTORE.	Delete the right part of the CCL starting at the cursor position, replacing it with the line below CCL. Deleted text is available with RESTORE.	<i>X</i> is positive. Delete <i>X</i> lines starting with CCL. Deleted text is available with RESTORE.	Delete lines or block defined by cursor. Deleted text is available with RESTORE.

- ERROR** means that an error will be indicated by either the sound of the bell or by an explicit error message.
- When an error message results from using ENTER *motion* FUNCTION, you might need to use CANCEL twice; the first CANCEL removes the error message and the second CANCELS the ENTER *motion* sequence.
- CCL stands for "Current Cursor Line."

FUNCTION	Action of FUNCTION	Action of ENTER FUNCTION	Action of ENTER <i>X</i> FUNCTION (<i>X</i> is any argument)	Action of ENTER <i>motion</i> FUNCTION (<i>motion</i> is a combination of cursor-positioning functions.)
DELETE-CHARACTER	Delete the character under the cursor.	Delete from the cursor to the end of CCL.	ERROR	ERROR
DO	Run last DO function exactly as it was given.	Print last DO function.	<i>X</i> is a command in the format [<i>n</i> [1]] <i>prg</i> [<i>arg...</i>]. Replace <i>n</i> paragraphs (or <i>n</i> lines if 1 appears) by the result of running filter <i>prg</i> on that text with given args. Old paragraphs are available with RESTORE.	ERROR
END-LINE	Move the cursor to the last non-space character of the current line.	Enter BOX/LINE mode; define a region to the last non-space character of the current line.	ERROR	Continue defining argument.
ENTER	Begin accepting function arguments.	Removes box.	Removes box.	Removes box.
ENTER (Key)	Move the cursor to the beginning of the next line.	Enter BOX/LINE mode; move the cursor to the left end of the next line.	ERROR	Continue defining argument.
EXECUTE	Select a menu item; answer a question in the affirmative.	No effect.	No effect.	ERROR
EXIT	Exit editor. All altered files are written to disk.	ERROR	ENTER q EXIT saves all structured files and not ASCII files.	Same as EXIT.
FONT	Exchange current and alternate fonts. Initial current font is Roman, alternate is continuous underline.	Save current font as alternate font; set current font to Roman, with no underline.	<i>X</i> is a string. Save current font as alternate font; set current font as specified by <i>X</i> . Permissible values for <i>X</i> are c, w, or g (continuous- or word-underline, or graphics.)	ERROR

FUNCTION	Action of FUNCTION	Action of ENTER FUNCTION	Action of ENTER <i>X</i> FUNCTION (<i>X</i> is any argument)	Action of ENTER <i>motion</i> FUNCTION (<i>motion</i> is a combination of cursor-positioning functions.)
FORMAT	Format the current paragraph with the margins set on the screen. The old version of the text is available with RESTORE.	ERROR	ERROR	ERROR
FUNCTIONS	Display the currently active FUNCTIONS menu.	Display the functions active while using ENTER.	Display the functions active while using ENTER <i>X</i> FUNCTION.	Display the functions active while using ENTER <i>motion</i> FUNCTION.
GO-TO	Move window so line 1 of file is top line. If already at line 1, move to the last line in the file.	Move window so last line of file is in the middle of the window.	<i>X</i> is positive. Move window so line <i>X</i> of file is in the middle of the window.	ERROR
HELP	Explain the current situation.	Explain the ENTER function.	Explain the ENTER function.	Describe TEXT mode.
HOME	Move the cursor to the upper left-hand corner of the screen.	Enter BOX/LINE mode and draw box from original cursor position to HOME position.	ERROR	Enter BOX/LINE mode and draw box from original cursor position to HOME position.
INSERT	Insert a blank line at CCL.	Move the right part of CCL starting at the cursor position to a new next line.	<i>X</i> is positive. Insert <i>X</i> blank lines above CCL.	Insert blank lines or block in area defined by cursor.
INSERT-MODE	Alternate between insert and overstrike mode.	Alternate between insert and overstrike mode.	Alternate between insert and overstrike mode.	ERROR
LAST-ARG	Display last ENTER argument.	Display last ENTER argument.	Display last ENTER argument.	ERROR
LEFT	Move window left one-third of its width or to file boundary, whichever is less.	Move window so that the column the cursor is in is the rightmost column.	<i>X</i> is positive. Move window left <i>X</i> columns or to file boundary, whichever is less.	Allow cursor-defined region to span vertical screen boundary.

FUNCTION	Action of FUNCTION	Action of ENTER FUNCTION	Action of ENTER <i>X</i> FUNCTION (<i>X</i> is any argument)	Action of ENTER <i>motion</i> FUNCTION (<i>motion</i> is a combination of cursor-positioning functions.)
+LINE -LINE	Move window forward (+) or backward (-) part of a page.	Move window so CCL is the first line in window (+) or last line in window (-).	<i>X</i> is positive. Move window forward (+) or backward (-) <i>X</i> lines.	Allow cursor-defined region to span horizontal screen boundary.
LINE-FEED	Put cursor in the first column of the next line of a ganged field if not on the last line or if the field is scrollable. Otherwise, ERROR.	ERROR	ERROR	ERROR
LOCAL-MENU	Display the local, data-specific task menu.	Display the local, data-specific task menu.	Display the local, data-specific task menu.	ERROR
MARGIN	Set left margin at current cursor column.	Set right margin at current cursor column. Enables <i>word wrap</i> if right margin is to the right of the left margin.	ERROR	Set left margin to left edge of the cursor-defined area and right margin to right edge.
MENU	Display the New Task Menu.	Display the New Task Menu.	Execute command <i>X</i> ; display the results in a popup box, if possible.	ERROR
NEXT	Display the next item in a list.	ERROR	<i>X</i> is positive. Display the <i>X</i> th item in a list.	ERROR
NEXT-WINDOW	The next window becomes the current window.	The last window becomes the current window.	ERROR	ERROR
+PAGE -PAGE	Move window forward (+) or backward (-) one page.	Begin definition of cursor-defined argument.	<i>X</i> is positive. Move window forward or backward <i>X</i> pages.	Allow cursor-defined region to span horizontal screen boundary.
PICK-COPY	Add CCL to buffer; move cursor to next line.	ERROR	<i>X</i> is positive. Place copy of <i>X</i> lines starting with CCL in buffer.	Place lines or block defined by cursor in buffer.
PICK-UP	Pick up CCL and add it to buffer.	Pick up the part of CCL starting with cursor position, replace it with the line below CCL and place it in the buffer.	<i>X</i> is positive. Pick up <i>X</i> lines starting with CCL and put them in buffer.	Pick up lines or block defined by cursor and put it in buffer.

FUNCTION	Action of FUNCTION	Action of ENTER FUNCTION	Action of ENTER <i>X</i> FUNCTION (<i>X</i> is any argument)	Action of ENTER <i>motion</i> FUNCTION (<i>motion</i> is a combination of cursor-positioning functions.)
PREVIOUS	Display the previous item in a list.	ERROR	<i>X</i> is positive. Display the <i>X</i> th item in a list.	ERROR
PRINT	Display the menu of print options.	ERROR	ERROR	ERROR
PUT-COPY	Place newest contents of buffer at cursor position (Note 4).	ERROR	<i>X</i> is positive. Place <i>X</i> copies of buffer contents at cursor location.	ERROR
PUT-DOWN	Move newest contents from buffer to cursor position (Note 4).	ERROR	<i>X</i> is positive. Place <i>X</i> copies of buffer contents at cursor position.	ERROR
QUIT	Terminate editing session after attempting to save files.	Terminate editing session after attempting to save files.	Terminate editing session after attempting to save files.	Terminate editing session after attempting to save files.
QUOTE	Translate the next typed character into its CTRL key equivalent.	ERROR	ERROR	ERROR
REFRESH	Redraw display.	Redraw display.	Redraw display.	Redraw display.
REPLACE	Replace the search string with the replace string.	Set the replace string to null. If cursor is at the beginning of the search string, that string is deleted.	<i>X</i> is a replace string. If cursor is at the beginning of the search string, it is replaced with the replace string.	ERROR
RESTORE	Place last-deleted text at cursor position (Note 4).	ERROR	<i>X</i> is positive. Place <i>X</i> copies of last deleted text at cursor position.	ERROR
RIGHT	Move window right one-third of its width or to file boundary, whichever is less.	Move window so that the column the cursor is in is the leftmost column.	<i>X</i> is positive. Move window right <i>X</i> columns or to file boundary, whichever is less.	Allow cursor-defined region to span vertical screen boundary.

4. If buffer contains full lines, CCL and following lines are pushed down to make room for inserted lines. If buffer contains a rectangular area, its contents are inserted with the upper left-hand corner of the block at the cursor position; the text to the right of the inserted material is pushed to the right.

FUNCTION	Action of FUNCTION	Action of ENTER FUNCTION	Action of ENTER <i>X</i> FUNCTION (<i>X</i> is any argument)	Action of ENTER <i>motion</i> FUNCTION (<i>motion</i> is a combination of cursor-positioning functions.)
SAVE	Write current file out on disk if it has been modified since last SAVE.	Write current file out on disk if it has been modified since last SAVE.	Write current file out on disk and copy it into file named <i>X</i> .	ERROR
+SEARCH -SEARCH	Search forward (+) or backward (-) for the last string searched for (if any).	Search forward (+) or backward (-) for the string pointed to by the cursor up to the first blank.	Search forward (+) or backward (-) for the string <i>X</i> .	ERROR
SET-TAB	Set tab stop at cursor column.	Remove tab stop at cursor column.	ERROR	ERROR
TAB -TAB	Move the cursor to the next tab stop right (TAB) or left (-TAB).	Enter BOX/LINE mode; define a region to the next tab stop.	ERROR	Continue defining argument.
TEXT-MARK	Enter TEXT mode.	ERROR	ERROR	ERROR
USE	Switch window to alternate file; current file becomes alternate file.	Edit file taking name from cursor position up to next blank; old current file becomes alternate file.	Edit file named <i>X</i> as current file; old current file becomes alternate file.	ERROR
WINDOW	Make a new window containing current file with border extending from current cursor position.	Delete all windows except the one containing the cursor.	Make a new window and edit file <i>X</i> in it.	ERROR
ZOOM-IN	Display more detailed information.	ERROR	ERROR	ERROR
ZOOM-OUT	Display more summary information.	ERROR	ERROR	ERROR

Appendix B: UTILITY PROGRAMS FOR STRUCTURED FILES

Programs described in this appendix are useful for manipulating the structured files described in §2.2. To run these programs, use **MENU** and select the option `Execute UNIX shell commands`. This will cause the editor to move into a subshell. To exit the subshell, press the **ENTER** key, then press `Ctrl-d`.

`ghost oldname [newname [m/d/y [h:m:s]]]`

Creates the version of *oldname* that existed at the time indicated by the last two arguments. If the last two arguments are omitted, the most recent version is produced. The output is put into *newname*. If *newname* is omitted, *oldname* is backed up in a `.bak` file and the results are put into *oldname*.

`history filename`

Prints out a detailed description of the history of changes to the structured file *filename*.

`newfile asciifile [structuredfile]`

Converts *asciifile* from ASCII to structured format and places it into *structuredfile*. If no second argument is given, the first argument file is saved into a `.bak` file and converted to structured format.

`readfile filename`

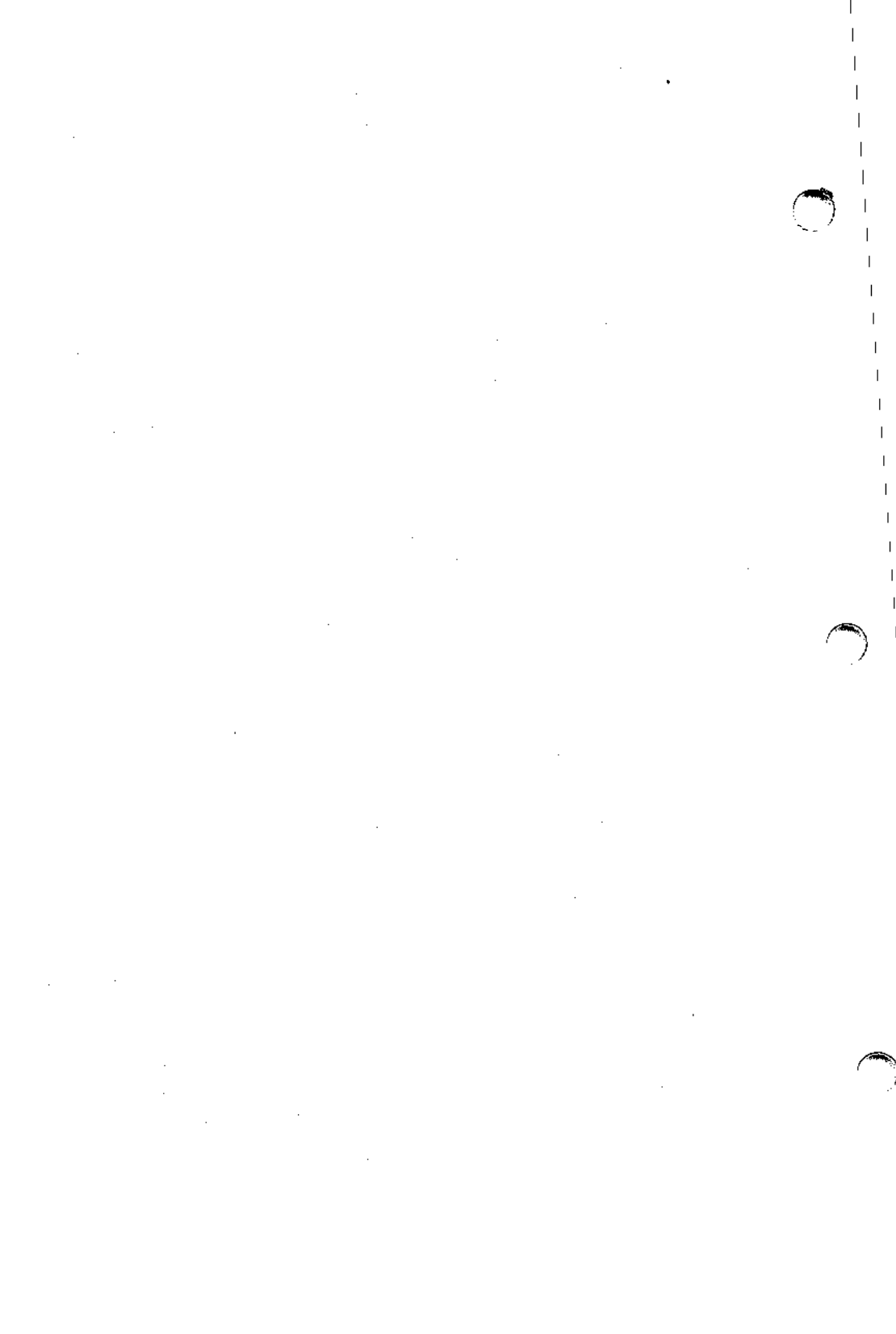
Writes the structured file *filename* onto the screen. Thus, `readfile filename > newfilename` can be used to convert the structured file *filename* into the ASCII file *newfilename*.

`rmhist file [file ...]`

Takes a list of files and removes the history information from each. The original version is saved in a `.bak` file. If the original file is not structured, a warning is given and the next file is processed.

`versions filename`

Uses `history` but only prints out the times when the file was changed. The result can be used to create an argument to `ghost` to recreate a file as it existed at a certain moment in its history.



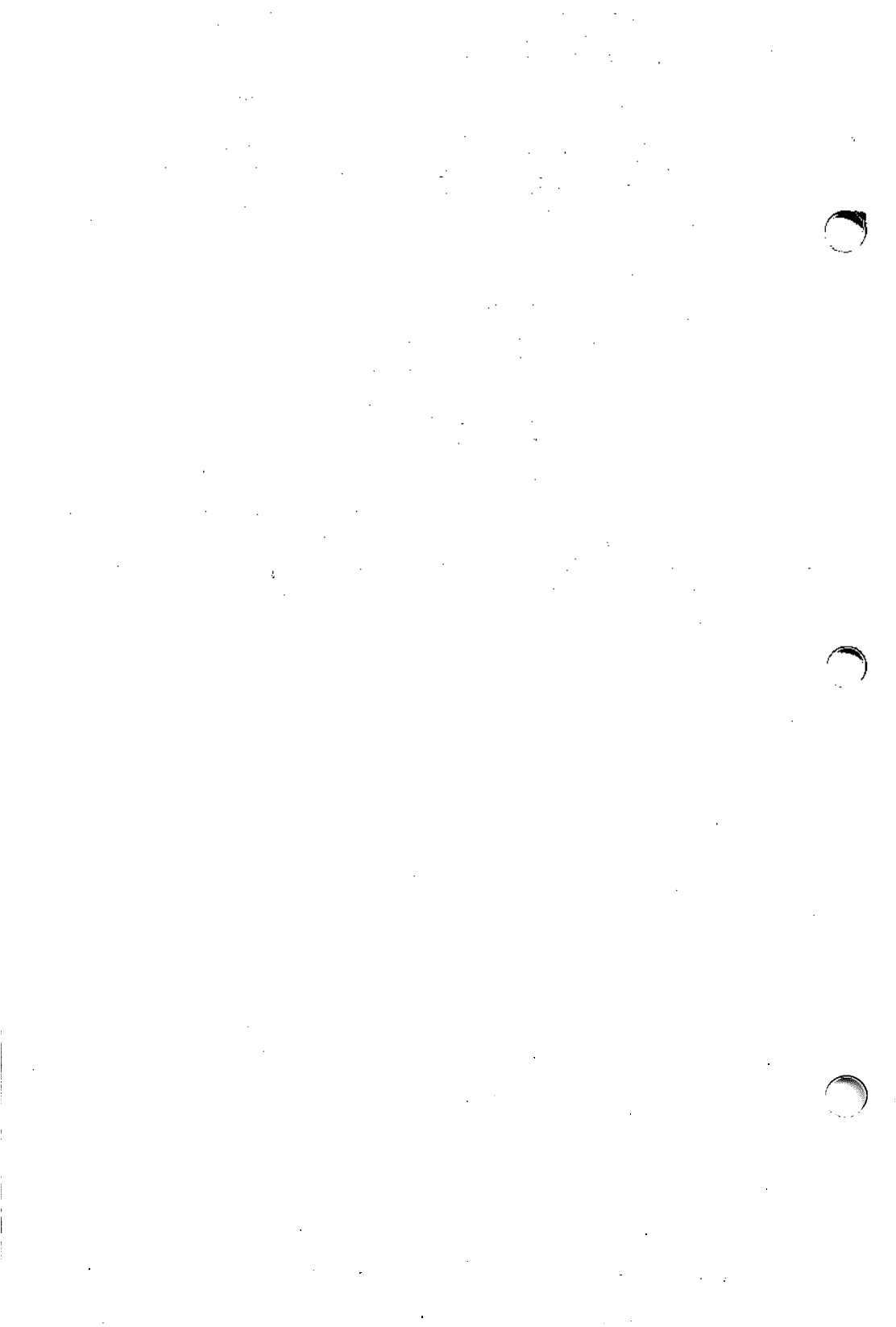
Appendix C: BOX CHARACTER SET

Box characters can be entered from the editor by using the graphics font or **QUOTE**. To create a box character, either enter the graphics font by using **ENTER** g **FONT**, or use **QUOTE** and type the letter that corresponds to the character you want:

<i>Graphics Font/QUOTE Character</i>	<i>Box Character</i>
s	upper left-hand corner
w	left-hand intersection of two corners
f	lower left-hand corner
q	horizontal line
t	top intersection of two corners
z	central intersection of four corners
r	bottom intersection of two corners
d	upper right-hand corner
e	right-hand intersection of two corners
a	vertical line
g	lower right-hand corner

Some terminals support a graphics character set to represent box characters. Others use standard keyboard characters to represent box characters. On displays that support graphics characters, the standard box characters in the graphics font (and their corresponding keys) are:

┌ ^s	- q	┐ ^t	└ ^d
├ ^w		┤ ^z	┘ ^e
			^a
└ ^f		┘ ^r	┐ ^g



Appendix D: ERROR MESSAGES

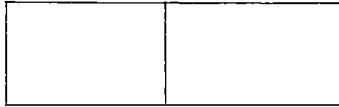
This appendix provides a glossary that defines terms used in error messages generated by the editor. It also explains each error message and gives suggestions for correcting each error situation.

The editor displays each error message in a popup box on the screen. Most frequently, error messages result when you use a function that is not valid in the current context, or when you attempt to perform an operation that makes no sense (such as deleting -3 lines).

Most of the error messages issued by the editor are listed below in alphabetical order. Error messages not listed probably indicate system or editor malfunctions and should be reported if they persist. It should be noted that in the event an operating system problem prevents a file from being saved, your editing session will not be lost. After the problem is corrected, type *e* without an argument at the command level to cause automatic recovery of the file.

1. GLOSSARY

- Field:** A field is an attribute of a structured data file and is defined in the form used to enter and display data. A field is assigned either a text data path or a tree data path. A field with a text data path can contain only text; a field with a tree data path, also referred to as an *indexed* or *list* field, contains more complex data.
- Ganged Fields:** Ganged fields are spatially adjacent fields that have tree data paths with at least one element common to all members of the gang.
- Indexed Field:** An indexed field is used in a structured data file and has a tree data path. Indexed fields are often adjacent to one another in a form (see also “Ganged Fields”). This example shows ganged, indexed fields:



- Item:** An item consists of the data in one line of an indexed field.
- List Field:** A list field is the same as an indexed field.
- Structured File:** A structured file contains information about the structure of the data in the file and about any changes that have been made to the data. A structured file contains hierarchical data that is displayed and edited using forms and may have multiple levels accessed via **ZOOM-IN** and **ZOOM-OUT**. The top level is the first level displayed when a structured file is edited.

2. ERROR MESSAGES

Alternate file "*filename*" does not exist.

Cause: You have used **USE** to switch to the alternate file, but the alternate file does not exist. It may have been removed or renamed, or you may never have defined an alternate file.

Action: Use **ENTER filename USE** to switch to an existing file.

Cannot backup file "*filename*".

Cause: You have used **ENTER filename SAVE**. This involves creating a backup copy of the file. Due to operating system problems, this is impossible. Usually, this means that you do not have write permission in the current directory. The file will not be saved.

Action: Change the access permissions or ask your system administrator to solve the operating system problem, then try **SAVE** again.

Cannot create directory "*directory*".

Cause: You have attempted to create *directory*; this operation has failed, possibly due to a lack of space.

Action: Contact your system administrator.

Cannot create file "*filename*".

Cause: You have attempted to create a file via **MENU**, but the editor cannot create the file; or you have tried to create a file when the disk is full.

Action: Contact your system administrator.

Cannot edit a ".*index*" file.

Cause: You have attempted to edit a file whose name is either *index* or ends with *.index*. You are not allowed to edit these files.

Action: Select another file to edit.

Cannot edit a special file.

Cause: You are attempting to edit a special UNIX System file. This might be a non-editable directory or device. The editor is not capable of editing these.

Action: Select another file to edit.

Cannot edit directory "*directory*" without read and execute permission.

Cause: You have attempted to edit a directory in which you do not have read or execute permission.

Action: Change the access permissions or ask your system administrator to change them so that you have read and execute permission, then try again.

Cannot edit file beginning with '*...*'.

Cause: You have attempted to edit a file that has the characters *...* at the beginning of the file name. These files cannot be edited.

Action: Select another file to edit.

Cannot edit file "*filename*"
without read permission.

Cause: You do not have permission to read *filename*.

Action: Use **LOCAL-MENU** to change the file's permission bits and re-edit it.

Cannot find help message file for this application.

Cause: You have used **HELP** while in a TEN/PLUS application, and the system is unable to locate the help message file for that application. Either the application was not installed properly or the search paths specified in the Editor Search Paths section of your editor profile are incorrect.

Action: Make sure that the Editor Search Paths section of your editor profile includes a separate line specifying \$SYS/hmgs as a search path. If this line is specified correctly and this message displays, contact your system administrator.

Cannot format in a list field.

Cause: You have used **FORMAT** in a field with a tree data path (that is, an indexed field). **FORMAT** can be used only in fields having text data paths.

Action: No action required.

Cannot go to next item in a list of files.

Cause: You have used **NEXT** while at the top level of a file. **NEXT** works only when you have used **ZOOM-IN** in a structured file having tree data paths.

Action: **ZOOM-IN** to the file if possible, then try to use **NEXT**.

Cannot go to next item of this list.

Cause: You have used **NEXT** when there is no next item in the list.

Action: **ZOOM-IN** or **ZOOM-OUT** if possible, then try to use **NEXT**.

Cannot go to previous item in a list of files.

Cause: You have used **PREVIOUS** while at the top level of a file. **PREVIOUS** only works when you have used **ZOOM-IN** in a structured file having tree data paths.

Action: **ZOOM-IN** to the file if possible, then try to use **PREVIOUS**.

Cannot go to previous item of this list.

Cause: You have used **PREVIOUS** when there is no previous item in the list.

Action: **ZOOM-IN** or **ZOOM-OUT** if possible, then try to use **PREVIOUS**.

Cannot join in indexed field.

Cause: You have used either **ENTER PICK-UP** or **ENTER DELETE** to combine two lines while in a field with a tree data path (that is, an indexed field). **ENTER PICK-UP** and **ENTER DELETE** can be used only in fields with text data paths.

Action: To join the lines, use pick and put operations in conjunction with **BOX-MARK** and the horizontal and/or vertical cursor-positioning functions.

Cannot open form "form".

Cause: Either you have used **ZOOM-IN** in a field for which the editor can find no zoom form, or you have used the Change form menu option to specify a form that the editor cannot find. If the editor cannot find a zoom form, the form may be installed in the wrong place. If this message displayed when you used **MENU**, you may have specified an invalid form name.

Action: If this error displayed while you were trying to **ZOOM-IN** to a field in a form, check the Editor Search Paths section of your editor profile; if necessary, contact your system administrator. If this error displayed while using the Change form menu option, provide a valid form name to the option.

Cannot open output file "*filename*".

Cause: You have tried to edit *filename*, but the editor is unable to open it. Either you do not have permission to create the file or directory, you have specified an illegal name, or the disk is full.

Action: Change the permission bits so that you can create the file or directory. To edit the file or directory, specify its legal name. If the disk is full, contact your system administrator.

Cannot put a window there.

Cause: You have attempted to create a window that is too small to contain data.

Action: Move the cursor and try again.

Cannot run program to "*program*".

Cause: You have tried to run a program that cannot execute.

Action: Contact your system administrator.

Cannot save editor state in file "*filename*".

Cause: The editor attempts to remember what you were doing just before you use **EXIT** so that you can continue the next time you run the editor. It does this by saving its state in a file. The attempt to open this file has failed. The next time you start the editor without an argument, you will be looking at a default file.

Action: No action required.

Cannot split in indexed field.

Cause: You have used **ENTER** **INSERT** to split a line while in a field with a tree data path (that is, an indexed field). **ENTER** **INSERT** can be used only in fields having text data paths.

Action: To split the line, use pick and put operations in conjunction with **BOX-MARK** and the horizontal and/or vertical cursor-positioning functions.

Cannot use TEXTREGION in a ganged field.

Cause: You have used **TEXT-MARK** to specify a region of text in a ganged field. **TEXT-MARK** can only be used in fields with text data paths; ganged fields have tree data paths.

Action: To manipulate data in a ganged field, use pick and put operations in conjunction with **BOX-MARK** and the horizontal and/or vertical cursor-positioning functions.

Cannot write to /tmp.

Cause: The operation you are trying to perform requires the editor to write to the /tmp directory. The editor is unable to complete the operation either because you do not have permission to write in the directory or there is not enough room in the directory.

Action: Contact your system administrator.

Cannot zoom in any further.

Cause: You have used **ZOOM-IN** at the lowest level of this file; you cannot **ZOOM-IN** any further.

Action: No action required.

Cannot zoom out any further.

Cause: You have used **ZOOM-OUT** in the top directory. You cannot **ZOOM-OUT** any further.

Action: No action required.

Character to be quoted must be a letter.

Cause: You have attempted to use **QUOTE** to insert a character into the file. The correct sequence is **QUOTE** followed by an alphabetic character. The control equivalent is then inserted into the file. You have used a nonalphabetic character that has no control equivalent.

Action: Determine the correct text character to type and try again.

Command does not take a numeric argument.

Cause: You have used **ENTER** followed by a number, followed by a function. The function, however, does not take a numeric argument.

Action: Refer to §8 to determine the correct way to accomplish what you want to do and try again.

Command does not take a region argument.

Cause: You have specified a region argument (using the cursor positioning functions with either **BOX-MARK** or **TEXT-MARK**) to a function that does not accept region arguments.

Action: Refer to §8 to determine the correct way to accomplish what you want to do and try again.

Command does not take a string argument.

Cause: You have used **ENTER** followed by a string, followed by a function. The function, however, does not take a string argument.

Action: Refer to §8 to determine the correct way to accomplish what you want to do and try again.

Command does not take an empty argument string.

Cause: You have used **ENTER** followed by a function. The function, however, does not take an empty argument string.

Action: Refer to §8 to determine the correct way to accomplish what you want to do and try again.

Command stopped by **BREAK**.

Cause: You have used **BREAK** while the editor was executing a command in a popup box.

Action: No action required.

Directory "*directory*" does not exist.

Cause: You have attempted to edit a file in a nonexistent directory.

Action: Specify a path to a file using existing directory names.

Error in LOCAL-MENU program.

Cause: There is a problem in the application you are using. The editor will try to restart the application so that you can continue editing.

Action: If the application does not restart, contact your system administrator.

File "*filename*" has not been modified.

Cause: You are attempting to save file *filename*; but it has not been modified, so there is no need to save it.

Action: No action required.

File "*filename*" is not a text or structured file.

Cause: You are attempting to edit a binary file. The editor is not capable of doing this.

Action: Specify an ASCII or structured file and edit it.

File "*/path/tofile*" is not a directory.

Cause: You have tried to edit a file; however, the path to the file included a component that was not a directory. If this error occurred while using **MENU**, then some files have not been installed correctly.

Action: Specify a correct path. If this error occurred while using **MENU**, contact your system administrator.

Filter error on command "*command*".

Cause: The filter that you just ran has failed. It should have printed its own error message.

Action: Determine the cause of the error and try again.

Filter stopped by BREAK.

Cause: You have used **BREAK** after using **DO**. **BREAK** stopped the editor from completing the execution of **DO** and caused this message to display.

Action: No action required.

Function key "*function*" has no special meaning for this data.

Cause: You have used a function that is not defined for this data. (Some applications define functions **(1)** through **(8)** to perform operations specific to certain kinds of structured files.)

Action: Use **LOCAL-MENU** to determine whether functions have been defined for the data you are editing.

Function not implemented.

Cause: You have used a function that the editor does not recognize.

Action: Use this manual to figure out the correct function and try again.

Helper error during restart.

Cause: The application you are using did not restart correctly, and currently is not functioning properly.

Action: **EXIT**, then try using the application again. If the application does not work after exiting and reentering the file, contact your system administrator.

Margin doesn't take a TEXTLINE region argument.

Cause: You have tried to use **TEXT-MARK** **MARGIN** to specify both the left and right margins simultaneously. **MARGIN** does not work with **TEXT-MARK**.

Action: Use **BOX-MARK** **MARGIN** to specify both margins at once.

Multiple windows are not implemented for non-text files.

Cause: You have used **WINDOW** while editing a structured data file. **WINDOW** can only be used with text files.

Action: No action required.

Negative numeric argument not allowed.

Cause: You have tried to provide a negative number as an argument to a function that accepts numeric arguments. Negative numeric arguments are not allowed.

Note, however, that negative numbers can be used as string arguments; for example, you can use **ENTER** -5 **+SEARCH**.

Action: Specify positive numeric arguments to commands.

No alternate file.

Cause: You have used **USE** to switch to an alternate file, but there is no alternate file because the current file is the first one to be edited in this session.

Action: Use **ENTER** *filename* **USE** to switch to a specific file.

No file name on current line.

Cause: You have used **ZOOM-IN** on a blank line in the File Manager display.

Action: Move the cursor to a line containing a file name before using **ZOOM-IN**.

No filter string set.

Cause: You have used **DO** expecting to rerun the last filter, but no previous filter string has been set.

Action: Use **ENTER** *argument* **DO**.

No margins allowed in list fields.

Cause: You have used **MARGIN** while in a field with a tree data path (that is, a list field). Margins are not allowed in fields with tree data paths.

Action: No action required.

No margins allowed in one line fields.

Cause: You have tried to set margins in a one-line field. Margins are not allowed in one-line fields.

Action: No action required.

No output from "program".

Cause: You have tried to run a program or command using **ENTER** *program* **MENU**. The program produced no output.

Action: No action required.

No previous item in this list.

Cause: You have used **PREVIOUS** while looking at the first item in a list; there are no items preceding the one you are viewing.

Action: No action required.

No search string.

Cause: You have used **+SEARCH** or **-SEARCH** expecting to search for a previously specified string, but this is your first attempt to search.

Action: Use **ENTER** string **+SEARCH** or **ENTER** string **-SEARCH** to specify a search string.

No space after line count in "number".

Cause: You have used **ENTER** number **DO** without specifying a command or without inserting a space between number and the command (for example, **ENTER** 5sort **DO**).

Action: Specify a command using **ENTER** number command **DO** (for example, **ENTER** 5 sort **DO**).

No windows to delete.

Cause: You have used **ENTER** **WINDOW** to delete all but the current window. But the current window is the only window.

Action: No action required.

Output too large to fit here ("number" lines).

Cause: You have used **EXECUTE**. It has generated more data than will fit into memory.

Action: No action required.

Restarting the editor on file "filename"....

Cause: The editor has detected a situation from which it cannot recover. It will attempt to restart itself.

Action: No action required.

Search failed on string "*string*".

Cause: There are no more occurrences of the search string in the file in the direction in which you were searching.

Action: No action required.

Stopped by BREAK.

Cause: You have used **BREAK** to stop a search.

Action: No action required.

String argument not allowed to next/previous.

Cause: You have used **ENTER** *string* **NEXT** or **ENTER** *string* **PREVIOUS** in a situation in which **NEXT** and **PREVIOUS** do not accept string arguments.

Action: No action required.

Tabs not allowed in this field.

Cause: You have tried to use **SET-TAB** or **ENTER** **SET-TAB** in a field that does not have tabs defined. This is not allowed.

Action: No action required.

The available fonts are ``r`` for roman, ``w`` for word underlining, ``c`` for continuous underlining, and ``g`` for graphics characters.

Cause: You have specified an illegal argument to **FONT**.

Action: Select a font from the set [rwcg].

The current filter string is "*filter*".

Cause: You have used **ENTER** **DO**; this instructs the editor to display this message indicating the *filter* last used. This *filter* will be used if **DO** is used without any arguments.

Action: No action required.

The cursor is not on a file name or character string.

Cause: You have used the sequence **ENTER** **USE** to switch to another file; however, the cursor is not on a file name.

Action: To switch to another file using **USE**, either: (1) use **ENTER**, type a file name in the popup box that displays, then use **USE**; or (2) move the cursor to the name of a file displayed on the screen, then use **ENTER USE**.

There are non-printing characters in "file"
Press Execute to edit "file" or press Cancel or Help.

Cause: You have entered a file name that contains characters that do not display, for example, control characters.

Action: **EXECUTE** to edit this file, otherwise, **CANCEL**.

There is no item "number" in this list.

Cause: You have used **ENTER number NEXT** or **ENTER number PREVIOUS** to move to a specific item; however, **NEXT** and **PREVIOUS** do not work at this level of the file.

Action: Use **ZOOM-IN** or **ZOOM-OUT** if possible, then try using **NEXT** or **PREVIOUS**. If **NEXT** and **PREVIOUS** work, use **ENTER number NEXT** or **ENTER number PREVIOUS** to move to a specific item.

There is no LOCAL-MENU for this data.

Cause: You have used **LOCAL-MENU** when there is no local menu for the type of data being edited.

Action: No action required.

There is no place available to save files.

Cause: You have tried to move or copy files using **DELETE**, **PICK-UP**, or **PICK-COPY**. The files are usually saved before they are moved, but the File Manager is unable to perform the save, either because you do not have write permission in your \$HOME/.putdir directory, or because your user i.d. has changed.

Action: Make sure that your \$HOME/.putdir directory allows you write permission. Contact your system administrator if necessary.

Unable to close file "*file*" during SAVE.

Cause: You have used **SAVE** while editing the current file; but, due to a lack of space on the disk, the system cannot save the file.

Action: Contact your system administrator before exiting the system.

Unable to restart helper "*helper*".

Cause: You have returned to an application that you were using earlier, but the editor is unable to restart the helper.

Action: Contact your system administrator. To continue working with the file, **EXIT** and reenter the file.

Warning: cannot fix file modes of "*filename*".

Cause: The editor was attempting to use the *chmod* system call on *filename*. The attempt failed. The editor will continue to try to open and save the file.

Action: If the file modes for the file are not those you desire, use the *chmod* command to change them.

Warning: write error on file "*filename*".

Cause: You have used **ENTER** *filename* **SAVE**. In attempting to write data into *filename*, the operating system encountered a disk write failure. No further data will be written.

Action: Determine the reason for the write failure, correct it, and try again.

You cannot modify this field.

Cause: You are trying to modify a field that has read permission only; this is not allowed.

Action: No action required.

You cannot modify this file.

Cause: You are attempting to type into or otherwise modify a file for which you do not have write permission.

Action: Change the permissions (if you are allowed to do so) and re-edit the file.

You cannot put a text region in a list field.

Cause: You have used **TEXT-MARK** and the cursor-positioning functions to pick up a text region then tried to put the text down in a field that has a tree data path (a list field). Fields with tree data paths cannot accept text regions.

Action: To move data from a field with a text data path to a field with a tree data path, use the pick and put operations alone or in conjunction with **BOX-MARK** and the horizontal and/or vertical cursor-positioning functions.

You cannot put "type" data in this field.

Cause: You have used **PICK-UP**, **PICK-COPY**, or **DELETE** to pick data from one type of field (such as a field with a tree data path), then tried to put it into another type of field (such as a field with a text data path or another type of tree data path). The data that you have picked can only be put into a field having the same structure.

Action: To pick text from any field, use the pick and put operations with **BOX-MARK** and the horizontal and/or vertical cursor-positioning functions. Text picked up in this way can be put into any type of field.

You cannot set the left margin to the right of the right margin.

Cause: You have tried to set the left margin to the right of the right margin; this is not allowed.

Action: To set the left margin, move the cursor to a position to the left of the right margin and use **MARGIN**.

You cannot set the right margin to the left of the left margin.

Cause: You have tried to set the right margin to the left of the left margin. This is not allowed.

Action: To set the right margin, move the cursor to a position to the right of the left margin and use **ENTER MARGIN**.

You cannot use the DO key in a list field.

Cause: You have tried to use **DO** in a field with a tree data path (that is, a list field). **DO** can only be used in a field with a text data path.

Action: No action required.

You do not have execute permission in "/path/directory".

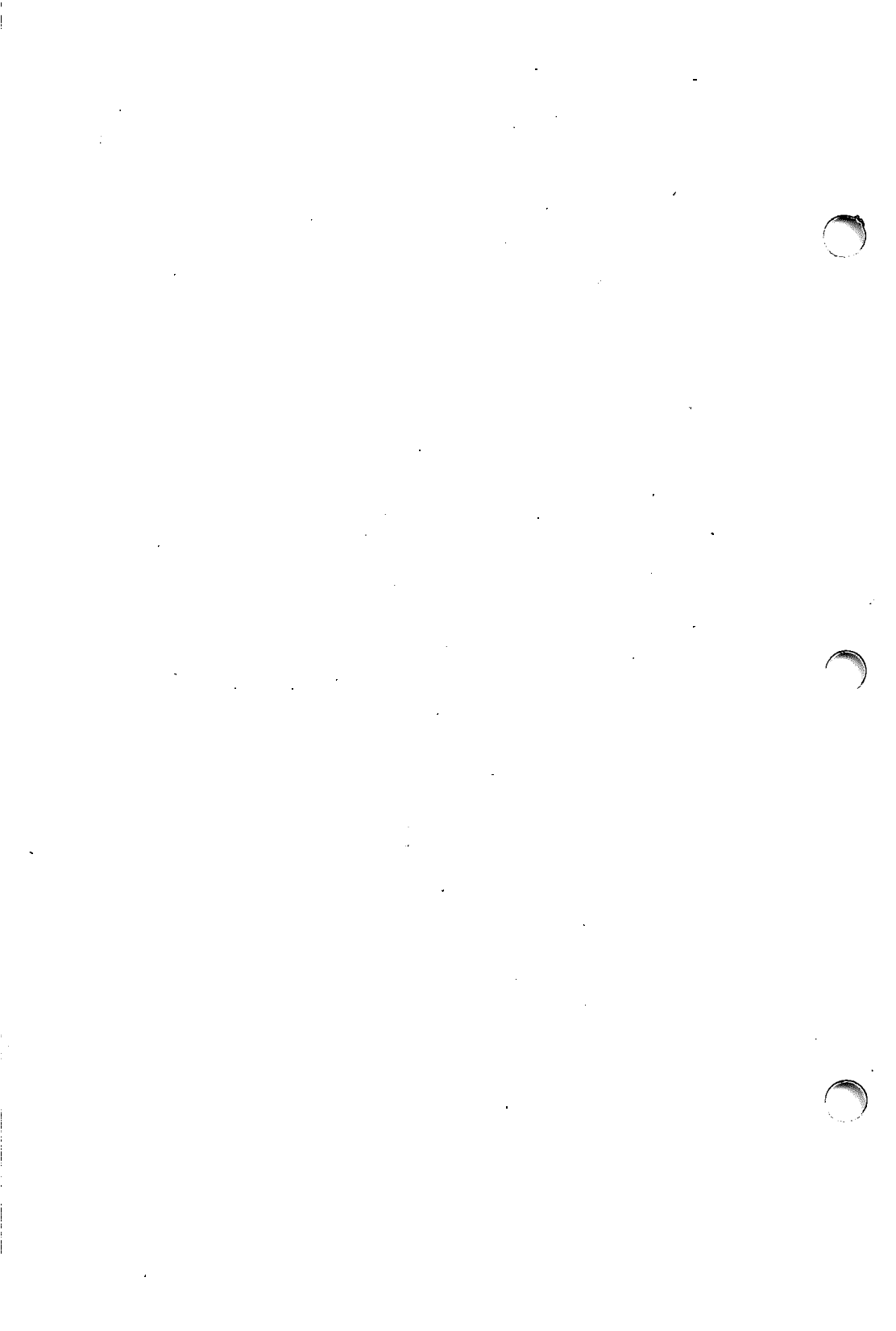
Cause: You are trying to edit a file that is located in a directory in which you do not have execute permission.

Action: Change the access permissions (if possible) or ask your system administrator to do so, then try again.

You do not have write permission in this directory.

Cause: You are trying to edit a file that is located in a directory in which you do not have write permission.

Action: Change the access permissions (if possible) or ask your system administrator to do so, then try again.



INDEX

- # file 25
- accessing a directory 8
- accessing a file 6
- active window, changing 23
- alternate file, creating 16
- alternate file, editing 16
- area defined by cursor motion 29
- area, definition of 28
- area, duplicating 20
- area, inserting 18
- area, moving 19
- argument, cancel 27
- argument, canceling 27
- argument, cursor-defined 28, 29
- argument, definition of 18
- argument, displaying last 27
- argument, errors when typing 27
- argument, invalid 27
- arguments, filter 31
- arguments to functions 27
- ASCII file 4
- ASCII file, saving 5, 17
- automatic recovery 35
- BACKSPACE 18, 20, 27
- backup procedures 3, 35
- .bak file 5, 17, 35
- BEGIN-LINE 12, 27, 29
- blocks, designating 28
- box characters 45
- boxes, removing 15
- BOX-MARK 28
- ****BOX/LINE**** 28
- BREAK 14, 22, 31, 33
- CANCEL 15, 27
- canceling a filter 31, 33
- canceling a search 22
- canceling an argument 27
- canceling operations 14
- cat filter 34
- CENTER 28, 33
- centering text 33
- character, deleting 20
- character, inserting 18
- components, User Interface 2
- control character, inserting 24
- control characters in text files 36
- copying a directory 8
- copying a file 6
- copying a line 28
- copying a sentence 29
- creating a directory 7
- creating a file 6
- creating a subdirectory 7
- current window 23
- cursor, definition of 18
- cursor, moving 12
- cursor-defined argument 28, 29
- cursor-defined argument, multiple page 29
- cursor-positioning functions 11
- cursor-positioning functions with ENTER 27
- date and time filter 34
- DELETE 25, 28
- deleted text, retrieving 20, 25
- DELETE-CHARACTER 20, 27
- deleting a character 20
- deleting a file 7
- deleting a line 28
- deleting a sentence 29
- deleting a string 32
- deleting remainder of line 20
- deleting windows 23
- designating a line 28
- detail, controlling 15
- directory, accessing 8
- directory, copying 8
- directory, creating 7
- directory, moving 9
- directory, removing 9
- directory, renaming 9
- directory, restoring 9
- display, redrawing 25
- dividing a window 23
- DO 31, 33
- DO with interactive programs 31
- DO without ENTER or an argument 32
- duplicating a line or area 20
- editing structured data 16
- editing text 2
- editing wide lines 24
- editing window 23
- editor profile 14, 36
- END-LINE 12, 27, 29
- end-of-file, moving to 23
- ENTER 11, 27, 28, 29, 31, 33
- ENTER # USE 25
- ENTER argument DO 31
- ENTER DELETE-CHARACTER 20
- ENTER filename SAVE 17
- ENTER filename USE 16
- ENTER filename WINDOW 23
- ENTER FONT 33
- ENTER GO-TO 13, 23
- ENTER horizontal motion MARGIN 32
- ENTER INSERT 19
- ENTER letter FONT 33
- ENTER +LINE 12
- ENTER MARGIN 32
- ENTER motion INSERT 19
- ENTER n CENTER 33
- ENTER n GO-TO 13, 22
- ENTER n INSERT 19
- ENTER n LEFT 13, 24
- ENTER n +/-LINE 12
- ENTER n +/-PAGE 12
- ENTER n PICK-COPY 20
- ENTER n PICK-UP 19
- ENTER n PUT-COPY 21
- ENTER n PUT-DOWN 21
- ENTER n RESTORE 25
- ENTER n RIGHT 13, 24
- ENTER NEXT-WINDOW 23
- ENTER PICK-UP 19

- ENTER REPLACE 22
- ENTER +SEARCH 22
- ENTER SET-TAB 24
- ENTER string REPLACE 22
- ENTER string +SEARCH 21
- ENTER USE 16
- ENTER vertical motion CENTER 33
- ENTER WINDOW 23
- ENTER with argument 27
- ENTER with cursor-defined argument 28
- environment variable 31
- error messages 14, 47
- errors when typing argument 27
- EXIT 5
- exiting a file 6
- exiting TEN/PLUS environment 5, 17
- file, # 25
- file, accessing 6
- file, ASCII 4
- file, copying 6
- file, creating 6
- file, deleting 7
- file, exiting 6
- file format 36
- file, history 4
- file, initializing 6
- File Manager, capabilities 3
- File Manager functions 3
- File Manager, using 3
- file, moving 7
- file, moving to beginning of 23
- file, renaming 7
- file, restoring 7
- file, saving 5, 17
- file, structured 4
- file types 4
- file versions, accessing 10
- file versions, saving 10
- files, editing several 16
- files, hidden 9
- filter 31
- filter arguments 31
- filter, canceling 14, 31, 33
- filter, cat 34
- filter, date 34
- filter, sort 34
- filter, tee 34
- FONT 33
- font, alternate 33
- font, current 33
- font, default alternate 33
- font, default current 33
- font selection 33
- font values 33
- fonts, switching 33
- FORMAT 25, 32
- format operations, undoing 32
- FORMAT, undoing 32
- formatting text 32
- function, definition of 18
- FUNCTIONS 25
- functions, cursor-positioning 11
- functions, File Manager 3
- functions, History Display 4
- functions, summary 37
- functions, text editing 18
- functions, text processing 31, 33
- functions, window-positioning 12
- ghost 43
- GO-TO 13, 22, 23
- graphics characters 45
- graphics font 24, 45
- HELP 14
- Help Menu 14
- hidden files 9
- history 35, 43
- History Display 35
- History Display, accessing 10
- History Display, capabilities 3
- History Display form 10
- History Display functions 4
- History Display, using 3
- history, removing 35
- history, structured file 4
- HOME 11
- HOME, multiple windows and 11
- Housekeep 35
- INed, features 2
- INed, initializing 4
- INed, running 4
- INed, using 2
- information boxes, using 15
- initializing a file 6
- initializing INed 4
- initializing TEN/PLUS system 4
- INSERT 18, 25, 28
- INSERT, cursor-defined 19
- insert mode 18, 20
- INSERT-MODE 18, 27
- inserting a character 18
- inserting a control character 24
- inserting a line 18
- inserting a n area 18
- interrupting MENU commands 15
- interrupting search/filter 14
- joining lines 19
- LAST-ARG 27
- leading spaces in text files 36
- LEFT 13, 24, 29
- limitations 36
- +LINE 12
- +/-LINE 28, 29
- line, copying 28
- line, deleting 28
- line, deleting remainder of 20
- line, duplicating 20
- line, inserting 18
- line, moving 19, 28
- line, moving to specified 22
- line, splitting 18, 19
- LINE-FEED 11
- lines, editing wide 24
- lines, joining 19
- LOCAL-MENU 9, 16
- LOCAL-MENU, using 16
- LOCAL-MENU 10, 25

- MARGIN 28, 32
- margin changes 33
- margins, resetting 32
- margins, setting 32
- MENU 14, 25
- MENU commands, interrupting 15
- menu, local 16
- MENU options 14
- MENU, using 15
- modes, changing 18
- moving a directory 9
- moving a file 7
- moving a line 19, 28
- moving a sentence 29
- moving an area 19
- moving text between files 21
- moving to beginning of file 23
- moving to end-of-file 23
- moving to specified line 22
- newfile 43
- new-line characters in text files 36
- NEXT 16
- NEXT-WINDOW 23
- numeric arguments with ENTER 47
- opening a sentence 29
- opening lines 28
- opening lines and areas 18
- operating system, accessing 14
- overwrite mode 18, 20
- PAGE 12
- +/-PAGE 28, 29
- pick buffer 19
- pick buffer, moving copies from 21
- pick buffer, removing text from 21
- PICK-COPY 20, 25, 28
- PICK-UP 19, 25, 28
- PICK-UP, cursor-defined 19
- picking lines 34
- PREVIOUS 16
- PRINT 15
- Print Helper, capabilities 3
- Print Helper, using 3
- Print Menu 15
- print profile 16
- printing 15
- Profile Helper, capabilities 3
- Profile Helper, using 3
- .putdir directory 7, 9
- .putdir, restoring a directory from 9
- .putdir, restoring a file from 7
- PUT-COPY 19, 21, 25
- PUT-DOWN 19, 21, 25
- QUIT 5
- QUOTE 24
- readfile 43
- recovery 35
- REFRESH 25
- removing a directory 9
- removing boxes 15
- renaming a directory 9
- renaming a file 7
- REPLACE 22
- replace function 32
- replacing a string 32
- RESTORE 20, 25, 31, 32
- restoring a directory 9
- restoring a file 7
- RIGHT 13, 24, 29
- rmhist 43
- rpl 32
- running text region 29
- SAVE 17, 35
- saving a file 5
- saving a structured file 5
- saving an ASCII file 5
- saving file versions 10
- +SEARCH 21
- SEARCH 22
- search and replace 22
- search, canceling 22
- search, interrupting 14
- searching for a string 21
- sentence, copying 29
- sentence, deleting 29
- sentence, moving 29
- sentence, opening 29
- SET-TAB 11, 24
- sort filter 34
- sorting lines 34
- spaces, leading 36
- spaces, trailing 36
- splitting a line 18, 19
- string, deleting 32
- string, replacing 32
- structured data, editing 16
- structured file 4
- structured file, restoring text from 25
- structured file, saving 5, 17
- structured file, viewing 15, 16
- structured files, utility programs 43
- subdirectory, creating 7
- subshell, entering 43
- subshell, exiting 43
- switching current/alternate files 16
- TAB 11
- tab stops 11
- tab stops, default 24
- tabs, clearing 11, 24
- tabs in text files 36
- tabs, setting 11, 24
- tee filter 34
- TEN/PLUS environment, exiting 5, 17
- TEN/PLUS system, initializing 4
- *****TEXT***** 29
- text, centering 33
- text editing 2
- text editing functions 18
- text file format 36
- text formatting 32
- text, moving between files 21
- text processing functions 31, 33
- text, retrieving deleted 25
- TEXT-MARK 28, 29
- TEXT-MARK, terminating 29
- time and date filter 34
- trailing spaces in text files 36

↑ ↓ → ← 11
undoing FORMAT operations 32
USE 16
User Interface components 2
utility programs, structured files 43
versions 43
wastebasket buffer 25
wastebasket buffer, adding lines to 25
wastebasket buffer, restoring lines from 25
WINDOW 23
window, current 23
window, dividing 23
window, editing 23
window, moving horizontally 24
window-positioning functions 12
windows 2
windows, changing 23
windows, creating several 23
windows, deleting 23
word wrap 18, 32
ZOOM-IN 15, 25
ZOOM-OUT 15, 25

TEN/PLUS Keyboard Information

CONTENTS

1. INTRODUCTION	1
1.1 Overview of This Document	1
2. Enhanced AT	2
2.1 The Enhanced AT Keyboard	2
2.2 TEN/PLUS Functions on the Enhanced AT Keyboard	4
3. AT	8
3.1 The AT Keyboard	8
3.2 TEN/PLUS Functions on the AT Keyboard	10
4. VT100	14
4.1 The VT100 Keyboard	14
4.2 TEN/PLUS Functions on the VT100 Keyboard	16
5. IBM 3101-1X AND IBM 3101-2X	20
5.1 The IBM 3101 Keyboards	20
5.2 TEN/PLUS Functions on the IBM 3101 Keyboards	22
6. ESPRIT ESP 6310	26
6.1 The Esprit ESP 6310 Keyboard	26
6.2 TEN/PLUS Functions on the Esprit ESP 6310 Keyboard	28
7. AT&T PERSONAL COMPUTER 6300	32
7.1 The AT&T Personal Computer 6300 Keyboard	32
7.2 TEN/PLUS Functions on the AT&T Personal Computer 6300 Keyboard	34
8. VT220 and VT240	38
8.1 The VT220 and VT240 Keyboards	38
8.2 TEN/PLUS Functions on VT220 and VT240 Keyboards	40
9. AT&T 5425 and AT&T 4425	44
9.1 The AT&T 5425 and AT&T 4425 Keyboards	44
9.2 TEN/PLUS Functions on the AT&T 5425 and AT&T 4425 Keyboards	46

Appendix: TERMINAL DESCRIPTION FILES 51



TEN/PLUS* Keyboard Information

1. INTRODUCTION

1.1 Overview of This Document

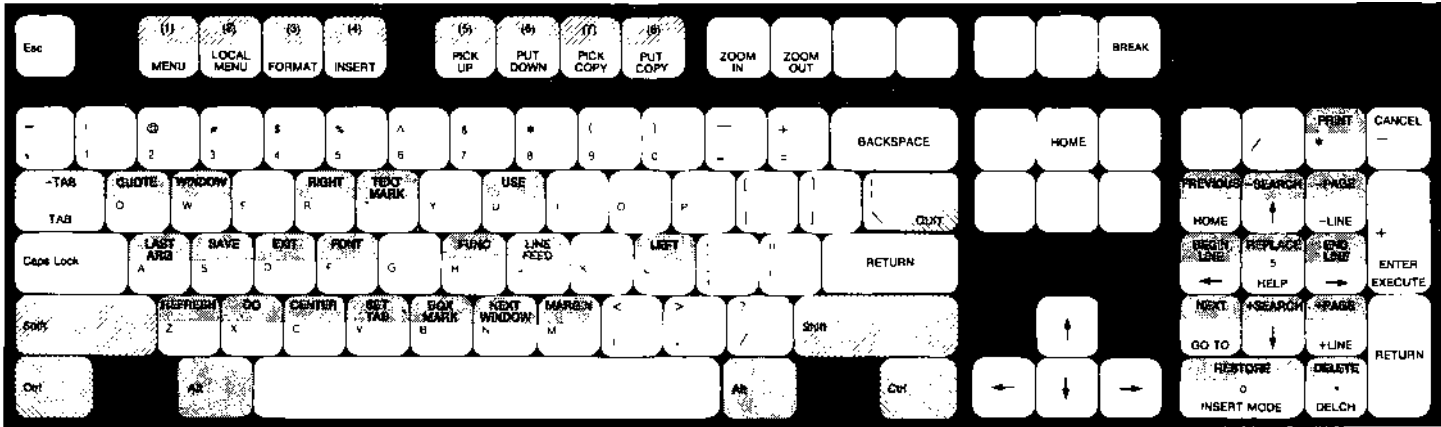
Functions in the TEN/PLUS environment are invoked through function keys or special key sequences. Different key sequences are required to invoke TEN/PLUS functions on different keyboards. This document shows the keyboard layout and function locations for various keyboards supported by the TEN/PLUS system and explains how to invoke TEN/PLUS functions on each. It is only necessary to read the section that describes your particular keyboard.


This document is intended as a supplement to other TEN/PLUS documents. It is essential that you keep it handy while reading the "TEN/PLUS Tutorial" and the "TEN/PLUS Reference Manual" in this guide.


2. Enhanced AT

2.1 The Enhanced AT Keyboard

Figure 1 shows the Enhanced AT* keyboard layout and function locations for the TEN/PLUS system. The same keyboard layout can be used when the TEN/PLUS system is invoked using the INTERACTIVE `xpcterm` utility.



To invoke a function indicated by , hold down the **Alt** key and touch the designated key at the same time.

To invoke a function indicated by , hold down the **Shift** key and touch the designated key at the same time.


To invoke a function indicated by , hold down the **Ctrl** key and touch the designated key at the same time.

Figure 1. Enhanced AT Keyboard Layout for the TEN/PLUS System

2.2 TEN/PLUS Functions on the Enhanced AT Keyboard

Functions are accessed in one of five ways on the Enhanced AT keyboard:

1. Touch the defined key. For example, **FORMAT** is invoked by touching F3 on the top row of keys.
2. Hold down the **Alt** key, then touch the designated key. For example, **BOX-MARK** is invoked by holding down **Alt**, then touching the **b** key.
3. Hold down the **Shift** key, then touch the designated key. For example, **-TAB** is invoked by holding down **Shift**, then touching the **TAB** key.
4. Hold down the **Ctrl** key, then touch the designated key. For example, **LINE-FEED** is invoked by holding down **Ctrl**, then touching **j**.

To log out from the system level, hold down the **Ctrl** key while simultaneously touching **d**.

<i>Function</i>	<i>Keystrokes</i>
(1)	Shift F1 on the top row of keys
(2)	Shift F2 on the top row of keys
(3)	Shift F3 on the top row of keys
(4)	Shift F4 on the top row of keys
(5)	Shift F5 on the top row of keys
(6)	Shift F6 on the top row of keys
(7)	Shift F7 on the top row of keys
(8)	Shift F8 on the top row of keys
↑	↑ to the left of the numeric keypad or on the numeric keypad
↓	↓ to the left of the numeric keypad or on the numeric keypad
←	← to the left of the numeric keypad or on the numeric keypad

→	→ to the left of the numeric keypad or on the numeric keypad
BACKSPACE	← Backspace on the main keyboard
BEGIN-LINE	Alt 4 on the numeric keypad
BOX-MARK	Alt b on the main keyboard
BREAK	Break to the right on the top row of keys
CANCEL	- in the upper right corner of the numeric keypad
CENTER	Alt c on the main keyboard
DELETE	Alt Del on the numeric keypad ¹
DELETE-CHARACTER	Del on the numeric keypad
DO	Alt x on the main keyboard
END-LINE	Alt 6 on the numeric keypad
ENTER	+ to the right of the numeric keypad ²
EXECUTE	+ to the right of the numeric keypad
EXIT	Alt d on the main keyboard
FONT	Alt f on the main keyboard
FORMAT	F3 on the top row of keys
FUNCTIONS	Alt h on the main keyboard
GO-TO	1 on the numeric keypad
HELP	5 on the numeric keypad
HOME	Home to the left of the numeric keypad or 7 on the numeric keypad

-
1. Do not confuse this key with Delete to the left of the numeric keypad.
 2. Do not confuse this key with Enter to the lower right of the numeric keypad or with Enter on the main keyboard.

INSERT	F4 on the top row of keys ³
INSERT-MODE	Ins on the numeric keypad
LAST-ARG	Alt a on the main keyboard
LEFT	Alt 1 on the main keyboard
+LINE	3 on the numeric keypad
--LINE	9 on the numeric keypad
LINE-FEED	Ctrl j on the main keyboard
LOCAL-MENU	F2 on the top row of keys
MARGIN	Alt m on the main keyboard
MENU	F1 on the top row of keys
NEXT	Alt 1 on the numeric keypad
NEXT-WINDOW	Alt n on the main keyboard
+PAGE	Alt 3 on the numeric keypad ⁴
--PAGE	Alt 9 on the numeric keypad ⁵
PICK-COPY	F7 on the top row of keys
PICK-UP	F5 on the top row of keys
PREVIOUS	Alt 7 on the numeric keypad
PRINT	Alt * at the top of the numeric keypad
PUT-COPY	F8 on the top row of keys
PUT-DOWN	F6 on the top row of keys
QUIT	Ctrl \ on the main keyboard
QUOTE	Alt q on the main keyboard

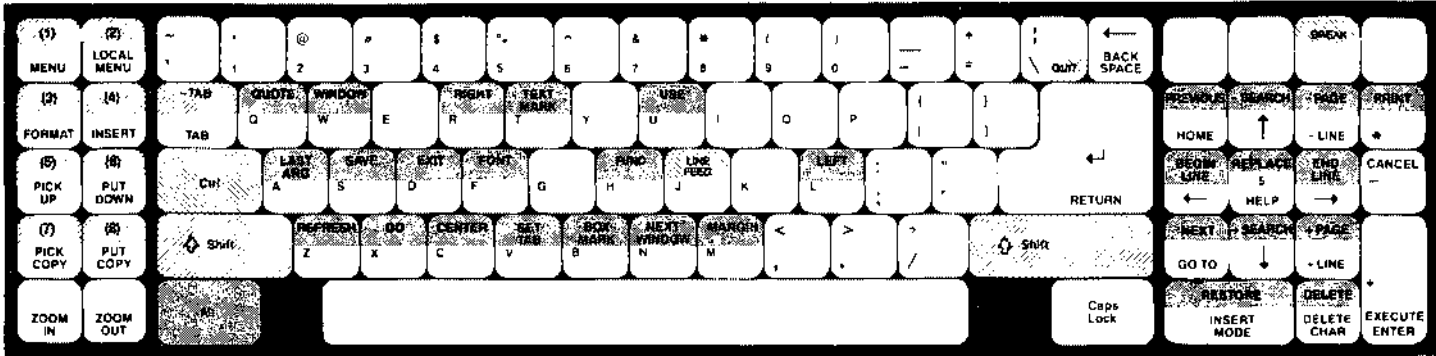
-
3. Do not confuse this key with **Insert** to the left of the numeric keypad or with **Ins** on the numeric keypad.
 4. Do not confuse this key with **Page Down** to the left of the numeric keypad.
 5. Do not confuse this key with **Page Up** to the left of the numeric keypad.


REFRESH	Alt z on the main keyboard
REPLACE	Alt 5 on the numeric keypad
RESTORE	Alt Ins on the numeric keypad
RETURN	Enter on the right of the main keyboard or to the right of the numeric keypad
RIGHT	Alt r on the main keyboard
SAVE	Alt s on the main keyboard
+SEARCH	Alt 2 on the numeric keypad
-SEARCH	Alt 8 on the numeric keypad
SET-TAB	Alt v on the main keyboard
TAB	Tab → on the left of the main keyboard
-TAB	Shift Tab ← on the left of the main keyboard
TEXT-MARK	Alt t on the main keyboard
USE	Alt u on the main keyboard
WINDOW	Alt w on the main keyboard
ZOOM-IN	F9 on the top row of keys
ZOOM-OUT	F10 on the top row of keys


3. AT

3.1 The AT Keyboard

Figure 2 shows the AT keyboard layout and function locations for the TEN/PLUS system.



To invoke a function indicated by , hold down the `Alt` key and touch the designated key at the same time.

To invoke a function indicated by , hold down the `Shift` key and touch the designated key at the same time.


To invoke a function indicated by , hold down the `Ctrl` key and touch the designated key at the same time.

Figure 2. AT Keyboard Layout for the TEN/PLUS System

3.2 TEN/PLUS Functions on the AT Keyboard

Functions are accessed in one of five ways on the AT terminal:

1. Touch the defined key. For example, **FORMAT** is invoked by touching **F3** on the function keypad.
2. Hold down the **Alt** key, then touch the designated key. For example, **BOX-MARK** is invoked by holding down **Alt**, then touching the **b** key.
3. Hold down the **Shift** key, then touch the designated key. For example, **-TAB** is invoked by holding down **Shift**, then touching the **TAB** key.
4. Hold down the **Ctrl** key, then touch the designated key. For example, **LINE-FEED** is invoked by holding down **Ctrl**, then touching **j**.

To log out from the system level, hold down the **Ctrl** key while simultaneously touching **d**.

<i>Function</i>	<i>Keystrokes</i>
(1)	Shift F1 on the function keypad
(2)	Shift F2 on the function keypad
(3)	Shift F3 on the function keypad
(4)	Shift F4 on the function keypad
(5)	Shift F5 on the function keypad
(6)	Shift F6 on the function keypad
(7)	Shift F7 on the function keypad
(8)	Shift F8 on the function keypad
↑	↑ on the numeric keypad
↓	↓ on the numeric keypad
←	← on the numeric keypad
→	→ on the numeric keypad
BACKSPACE	← to the top and right on the main keyboard

BEGIN-LINE	Alt 4 on the numeric keypad
BOX-MARK	Alt b on the keyboard
BREAK	Ctrl Scroll Lock at the top of the numeric keypad
CANCEL	- to the right of the numeric keypad
CENTER	Alt c on the keyboard
DELETE	Alt Del on the numeric keypad
DELETE-CHARACTER	Del on the numeric keypad
DO	Alt x on the keyboard
END-LINE	Alt 6 on the numeric keypad
ENTER	+ to the right of the numeric keypad ⁶
EXECUTE	+ to the right of the numeric keypad
EXIT	Alt d on the keyboard
FONT	Alt f on the keyboard
FORMAT	F3 on the function keypad
FUNCTIONS	Alt h on the keyboard
GO-TO	1 on the numeric keypad
HELP	5 on the numeric keypad
HOME	7 on the numeric keypad
INSERT	F4 on the function keypad
INSERT-MODE	Ins on the numeric keypad
LAST-ARG	Alt a on the keyboard
LEFT	Alt l on the keyboard
+LINE	3 on the numeric keypad

6. Do not confuse this key with Enter on the right of the keyboard.

-LINE	9 on the numeric keypad
LINE-FEED	Ctrl j on the keyboard
LOCAL-MENU	F2 on the function keypad
MARGIN	Alt m on the keyboard
MENU	F1 on the function keypad
NEXT	Alt 1 on the numeric keypad
NEXT-WINDOW	Alt n on the keyboard
+PAGE	Alt 3 on the numeric keypad
-PAGE	Alt 9 on the numeric keypad
PICK-COPY	F7 on the function keypad
PICK-UP	F5 on the function keypad
PREVIOUS	Alt 7 on the numeric keypad
PRINT	Alt * to the right of the numeric keypad
PUT-COPY	F8 on the function keypad
PUT-DOWN	F6 on the function keypad
QUIT	Ctrl \ on the keyboard
QUOTE	Alt q on the keyboard
REFRESH	Alt z on the keyboard
REPLACE	Alt 5 on the numeric keypad
RESTORE	Alt Ins on the numeric keypad
RETURN	Enter on the right of the keyboard
RIGHT	Alt r on the keyboard
SAVE	Alt s on the keyboard
+SEARCH	Alt 2 on the numeric keypad
-SEARCH	Alt 8 on the numeric keypad
SET-TAB	Alt v on the keyboard
TAB	← → on the left of the keyboard

-TAB

Shift ← → on the left of the keyboard

TEXT-MARK

Alt t on the keyboard

USE

Alt u on the keyboard

WINDOW

Alt w on the keyboard

ZOOM-IN

F9 on the function keypad

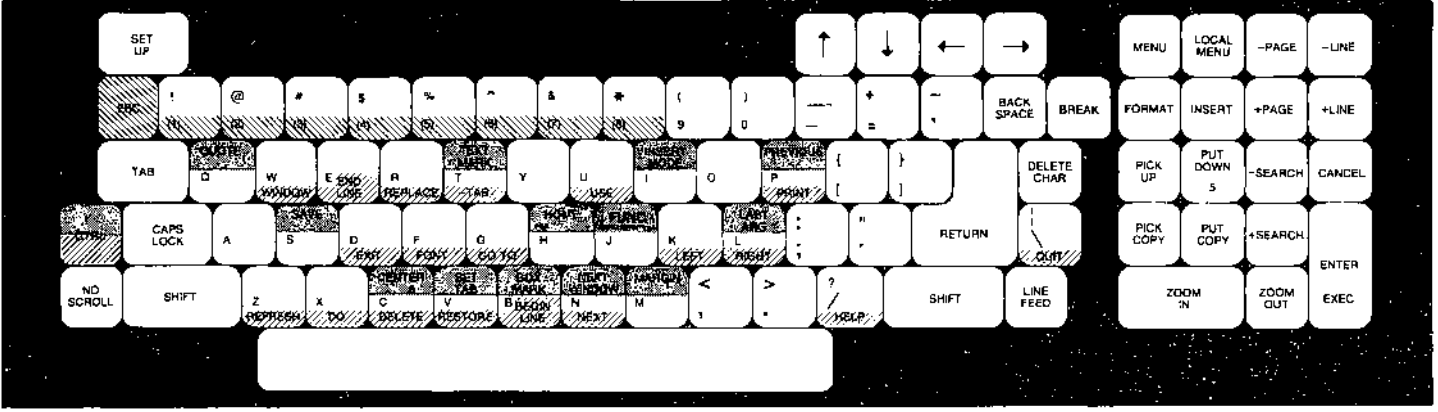
ZOOM-OUT


F10 on the function keypad


4. VT100

4.1 The VT100 Keyboard

Figure 3 shows the VT100* keyboard layout and function locations for the TEN/PLUS system. If you are using TEN/PLUS keycap covers, it is not necessary to refer to §4.2.



To invoke a function indicated by , hold down the CTRL key and touch the designated key at the same time.

To invoke a function indicated by , hold down the CTRL and a keys *simultaneously*, then release both and touch the designated key.

To invoke a function indicated by , touch the ESC key, then release it and touch the designated key.

Figure 3. VT100 Keyboard Layout for the TEN/PLUS System

4.2 TEN/PLUS Functions on the VT100 Keyboard

Functions are accessed in one of four ways on a VT100:

1. Touch the defined key. For example, **FORMAT** is invoked by touching 7 on the numeric keypad.
2. Hold down the **CTRL** key, then touch the defined key. For example, **HELP** is invoked by holding down **CTRL**, then touching the ? key.
3. Hold down the **CTRL** and a keys simultaneously, release both, then touch the designated key. For example, **SAVE** is invoked by touching **CTRL** and a together, then releasing both keys, and touching s.
4. Touch **ESC**, then touch the appropriate number on the main keyboard above the alpha characters. For example, **(1)** is invoked by touching **ESC**, then touching 1 on the main keyboard.

To log out from the system level, hold down the **CTRL** key while simultaneously touching d.

<i>Function</i>	<i>Keystrokes</i>
(1)	ESC then 1 on the keyboard
(2)	ESC then 2 on the keyboard
(3)	ESC then 3 on the keyboard
(4)	ESC then 4 on the keyboard
(5)	ESC then 5 on the keyboard
(6)	ESC then 6 on the keyboard
(7)	ESC then 7 on the keyboard
(8)	ESC then 8 on the keyboard
↑	↑ on the upper row of the keyboard
↓	↓ on the upper row of the keyboard
←	← on the upper row of the keyboard
→	→ on the upper row of the keyboard
BACKSPACE	BACKSPACE to the left of the break key

BEGIN-LINE	CTRL b on the keyboard
BOX-MARK	CTRL a then b
BREAK	BREAK in the upper right-hand corner of the keyboard
CANCEL	, to the right of the numeric keypad
CENTER	CTRL a then c
DELETE	CTRL c on the keyboard
DELETE-CHARACTER	DELETE key
DO	CTRL x on the keyboard
END-LINE	CTRL e on the keyboard
ENTER	ENTER to the right of the numeric keypad
EXECUTE	ENTER to the right of the numeric keypad
EXIT	CTRL d on the keyboard
FONT	CTRL f on the keyboard
FORMAT	7 on the numeric keypad
FUNCTIONS	CTRL a then j on the keyboard
GO-TO	CTRL g on the keyboard
HELP	CTRL ? on the keyboard
HOME	CTRL a then h on the keyboard
INSERT	8 on the numeric keypad
INSERT-MODE	CTRL a then i on the keyboard
LAST-ARG	CTRL a then l on the keyboard
LEFT	CTRL k on the keyboard
+LINE	- on the numeric keypad
-LINE	PF4 above the numeric keypad
LINE-FEED	LINE FEED on the right of the keyboard
LOCAL-MENU	PF2 above the numeric keypad

MARGIN	CTRL a then m on the keyboard
MENU	PF 1 above the numeric keypad
NEXT	CTRL n on the keyboard
NEXT-WINDOW	CTRL a then n on the keyboard
+PAGE	9 on the numeric keypad
-PAGE	PF 3 above the numeric keypad
PICK-COPY	1 on the numeric keypad
PICK-UP	4 on the numeric keypad
PREVIOUS	CTRL a then p on the keyboard
PRINT	CTRL p on the keyboard
PUT-COPY	2 on the numeric keypad
PUT-DOWN	5 on the numeric keypad
QUIT	CTRL \ on the keyboard
QUOTE	CTRL a then q on the keyboard
REFRESH	CTRL z on the keyboard
REPLACE	CTRL r on the keyboard
RESTORE	CTRL v on the keyboard
RETURN	RETURN on the right of the keyboard
RIGHT	CTRL l on the keyboard
SAVE	CTRL a then s on the keyboard
+SEARCH	3 on the numeric keypad
-SEARCH	6 on the numeric keypad
SET-TAB	CTRL a then v on the keyboard
TAB	TAB on the left of the keyboard
-TAB	CTRL t on the keyboard
TEXT-MARK	CTRL a then t on the keyboard
USE	CTRL u on the keyboard

WINDOW

CTRL w on the keyboard

ZOOM-IN

0 on the numeric keypad

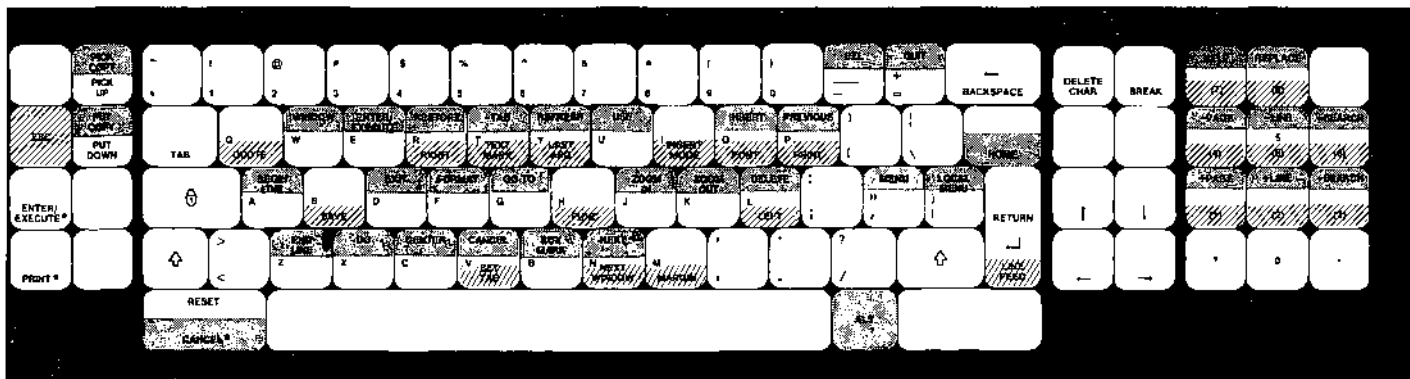
ZOOM-OUT


. on the numeric keypad


5. IBM 3101-1X AND IBM 3101-2X

5.1 The IBM 3101 Keyboards

Figure 4 shows the IBM* 3101-1x keyboard layout and function locations for the TEN/PLUS system and notes the differences for the IBM 3101-2x keyboard.



To invoke a function indicated by , hold down the ALT key and touch the designated key at the same time.

To invoke a function indicated by , touch and release the ESC key, then touch the designated key.

An asterisk (*) is used to indicate the location of a function on an IBM 3101-2x keyboard, if different from the location on an IBM 3101-1x keyboard.

Figure 4. IBM 3101-1x Keyboard Layout for the TEN/PLUS System

5.2 TEN/PLUS Functions on the IBM 3101 Keyboards

Functions are accessed in one of three ways on an IBM 3101:

1. Touch the defined key. For example, **PICK-UP** is invoked by touching ERASE EOL/EOF.
2. Hold down the ALT key, then touch the defined key. For example, **BEGIN-LINE** is invoked by holding down ALT while touching a.
3. Touch the ESC key, release, then touch the defined key. For example, **QUOTE** is invoked by touching ESC, releasing it, then touching q.

To log out from the system level, hold down the ALT key while simultaneously touching d.

<i>Function</i>	<i>Keystrokes</i>
(1)	ESC then 1 on the numeric keypad
(2)	ESC then 2 on the numeric keypad
(3)	ESC then 3 on the numeric keypad
(4)	ESC then 4 on the numeric keypad
(5)	ESC then 5 on the numeric keypad
(6)	ESC then 6 on the numeric keypad
(7)	ESC then 7 on the numeric keypad
(8)	ESC then 8 on the numeric keypad
↑	↑ between the numeric keypad and the main keyboard
→	→ between the numeric keypad and the main keyboard
↓	↓ between the numeric keypad and the main keyboard
←	← between the numeric keypad and the main keyboard
BACKSPACE	← on the main keyboard
BEGIN-LINE	ALT a on the main keyboard

BOX-MARK	ALT b on the main keyboard
BREAK	BREAK between the numeric keypad and the main keyboard
CANCEL	ALT v on the main keyboard
CENTER	ALT c on the main keyboard
DELETE	ALT 1 on the main keyboard
DELETE-CHARACTER	DEL between the numeric keypad and the main keyboard ⁷
DO	ALT x on the main keyboard
END-LINE	ALT z on the main keyboard
ENTER	ALT e on the main keyboard
EXECUTE	ALT e on the main keyboard
EXIT	ALT d on the main keyboard
FONT	ESC then o on the main keyboard
FORMAT	ALT f on the main keyboard
FUNCTIONS	ESC then h on the main keyboard
GO-TO	ALT g on the main keyboard
HELP	ALT 7 on the numeric keypad
HOME	ALT ← on the main keyboard
INSERT	ALT o on the main keyboard
INSERT-MODE	ESC then i on the main keyboard
LAST-ARG	ESC then y on the main keyboard
LEFT	ESC then 1 on the main keyboard
+LINE	ALT 2 on the numeric keypad

7. Do not confuse this key with DEL CHAR.

--LINE	ALT 5 on the numeric keypad
LINE-FEED	ESC then ↵
LOCAL-MENU	ALT { on the main keyboard
MARGIN	ESC then m on the main keyboard
MENU	ALT ' on the main keyboard
NEXT	ALT n on the main keyboard
NEXT-WINDOW	ESC then n on the main keyboard
+PAGE	ALT 1 on the numeric keypad
-PAGE	ALT 4 on the numeric keypad
PICK-COPY	ALT ERASE EOL/EOF to the left of the main keyboard
PICK-UP	ERASE EOL/EOF to the left of the main keyboard
PREVIOUS	ALT p on the main keyboard
PRINT	ESC then p on the main keyboard
PUT-COPY	ALT ERASE EOS to the left of the main keyboard
PUT-DOWN	ERASE EOS to the left of the main keyboard
QUIT	ALT =
QUOTE	ESC then q on the main keyboard
REFRESH	ALT y on the main keyboard
REPLACE	ALT 8 on the numeric keypad
RESTORE	ALT r on the main keyboard
RETURN	↵
RIGHT	ESC then r on the main keyboard
SAVE	ESC then s on the main keyboard
+SEARCH	ALT 3 on the numeric keypad
-SEARCH	ALT 6 on the numeric keypad

SET-TAB	ESC then v on the main keyboard
TAB	→ on the main keyboard
-TAB	ALT t on the main keyboard
TEXT-MARK	ESC then t on the main keyboard
USE	ALT u on the main keyboard
WINDOW	ALT w on the main keyboard
ZOOM-IN	ALT j on the main keyboard
ZOOM-OUT	ALT k on the main keyboard

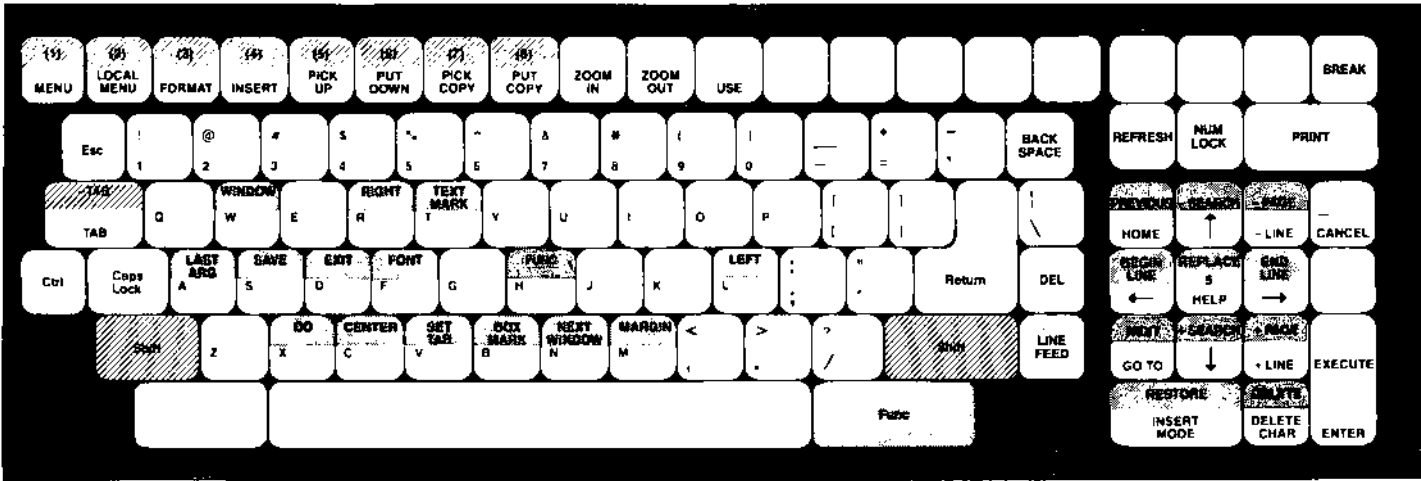
If you are using an IBM 3101-2X terminal, the following key sequences are also available:


<i>Function</i>	<i>Keystrokes</i>
ENTER	PRINT MSG to the left of the main keyboard
EXECUTE	PRINT MSG to the left of the main keyboard
PRINT	PRINT to the left of the main keyboard
CANCEL	ALT RESET to the left of the main keyboard

6. ESPRIT ESP 6310

6.1 The Esprit ESP 6310 Keyboard

Figure 5 shows the Esprit ESP 6310* keyboard layout and function locations for the TEN/PLUS system. The keyboard layout assumes that the terminal is in the Hazeltine emulation mode.



To invoke a function indicated by , hold down the **Func** key and touch the designated key at the same time.


To invoke a function indicated by , hold down the **Shift** key and touch the designated key at the same time.

Figure 5. Esprit ESP 6310 Keyboard Layout for the TEN/PLUS System

6.2 TEN/PLUS Functions on the Esprit ESP 6310 Keyboard

Functions are accessed in one of three ways on the Esprit ESP 6310:

1. Touch the defined key. For example, **HELP** is invoked by touching 5 on the numeric keypad.
2. Hold down the **Func** key, then touch the defined key. For example, **DO** is invoked by holding down the **Func** key and touching **x**.
3. Hold down the **Shift** key, then touch the defined key. For example, **(1)** is invoked by holding down the **Shift** key and touching **F 1**.

The only exception is **QUIT**, which is invoked by holding down the **Ctrl** key and touching **q**.

To log out from the system level, hold down the **Ctrl** key while simultaneously touching **d**.

<i>Function</i>	<i>Keystrokes</i>
(1)	Shift F 1 on the top row of keys
(2)	Shift F 2 on the top row of keys
(3)	Shift F 3 on the top row of keys
(4)	Shift F 4 on the top row of keys
(5)	Shift F 5 on the top row of keys
(6)	Shift F 6 on the top row of keys
(7)	Shift F 7 on the top row of keys
(8)	Shift F 8 on the top row of keys
↑	8 on the numeric keypad ⁸
→	6 on the numeric keypad ⁸
↓	2 on the numeric keypad ⁸

8. The arrow keys on the top of the Esprit keyboard are not guaranteed to perform these functions.

←	4 on the numeric keypad ⁸
BACKSPACE	Bs on the top right of the keyboard or ← on the top row of keys
BEGIN-LINE	Func 4 on the numeric keypad
BOX-MARK	Func b on the keyboard
BREAK	Break above the numeric keypad
CANCEL	- on the right of the numeric keypad
CENTER	Func c on the keyboard
DEL	Del on the right of the keyboard
DELETE	Func . on the numeric keypad
DELETE-CHARACTER	. on the numeric keypad
DO	Func x on the keyboard
END-LINE	Func 6 on the numeric keypad
ENTER	Enter on the right of the numeric keypad
EXECUTE	Enter on the right of the numeric keypad
EXIT	Func d on the keyboard
FONT	Func f on the keyboard
FORMAT	F3 on the top row of keys
FUNCTIONS	Func h on the keyboard
GO-TO	1 on the numeric keypad
HELP	5 on the numeric keypad
HOME	7 on the numeric keypad
INSERT	F4 on the top row of keys
INSERT-MODE	0 on the numeric keypad
LAST-ARG	Func a on the keyboard
LEFT	Func 1 on the keyboard
+LINE	3 on the numeric keypad

--LINE	9 on the numeric keypad
LINE-FEED	LINE FEED on the keyboard
LOCAL-MENU	F2 on the top row of keys
MARGIN	Func m on the keyboard
MENU	F1 on the top row of keys
NEXT	Func 1 on the numeric keypad
NEXT-WINDOW	Func n on the keyboard
+PAGE	Func 3 on the numeric keypad
-PAGE	Func 9 on the numeric keypad
PICK-COPY	F7 on the top row of keys
PICK-UP	F5 on the top row of keys
PREVIOUS	Func 7 on the numeric keypad
PRINT	Send above the numeric keypad
PUT-COPY	F8 on the top row of keys
PUT-DOWN	F6 on the top row of keys
QUIT	Ctrl q
REFRESH	Clear All above the numeric keypad
REFRESH	Func z on the keyboard ⁹
REPLACE	Func 5 on the numeric keypad
RESTORE	Func 0 on the numeric keypad
RETURN	Return on the right of the keyboard
RIGHT	Func r on the keyboard
SAVE	Func s on the keyboard
+SEARCH	Func 2 on the numeric keypad

9. This is an alternate key sequence.

-SEARCH	Func 8 on the numeric keypad
SET-TAB	Func v on the keyboard
TAB	Tab on the left of the keyboard
-TAB	Shift Tab on the left of the keyboard
TEXT-MARK	Func t on the keyboard
USE	F 11 on the top row of keys
USE	Func u on the keyboard ¹⁰
WINDOW	Func w on the keyboard
ZOOM-IN	F 9 on the top row of keys
ZOOM-OUT	F 10 on the top row of keys

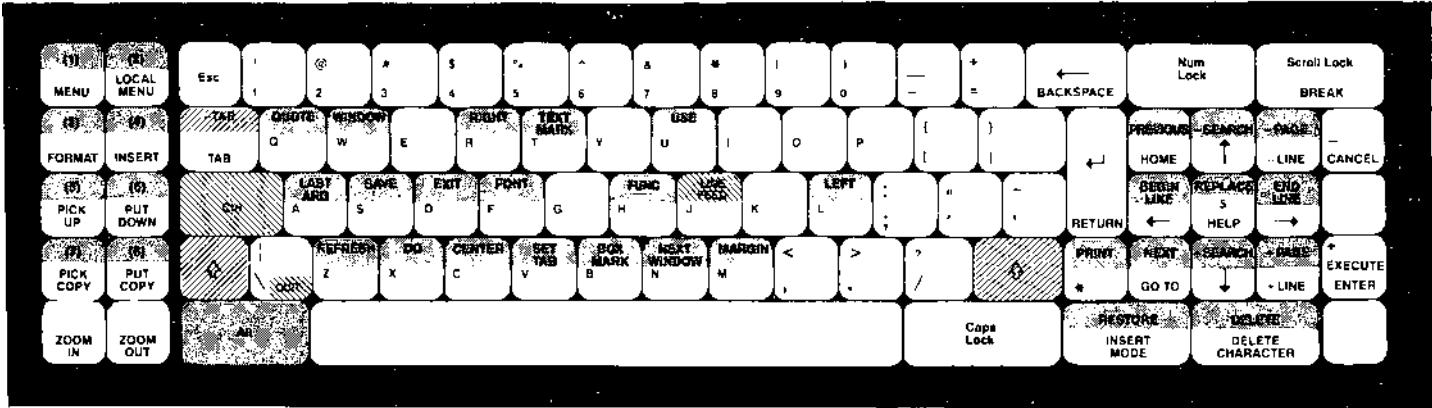
10. This is an alternate key sequence.


7. AT&T PERSONAL COMPUTER 6300

7.1 The AT&T Personal Computer 6300 Keyboard

Figure 6 shows the AT&T Personal Computer 6300* keyboard layout and function locations for the TEN/PLUS system.





To invoke a function indicated by , hold down the **Alt** key and touch the designated key at the same time.

To invoke a function indicated by , hold down the **Shift** key and touch the designated key at the same time.

To invoke a function indicated by , hold down the **Ctrl** key and touch the designated key at the same time.

Figure 6. AT&T Personal Computer 6300 Keyboard Layout for the TEN/PLUS System

7.2 TEN/PLUS Functions on the AT&T Personal Computer 6300 Keyboard

Functions are accessed in one of four ways on an AT&T Personal Computer 6300:

1. Touch the defined key. For example, **HELP** is invoked by touching 5 on the numeric keypad. (The Num Lock key must *not* be on.)
2. Hold down the **Alt** key, then touch the designated key. For example, **DO** is invoked by holding down the **Alt** key, then touching **x**.
3. Hold down the **Shift** key, then touch the designated key. For example, **-TAB** is invoked by holding down the **Shift** key, then touching **←**.
4. Hold down the **Ctrl** key, then touch the designated key. For example, **LINE-FEED** is invoked by holding down the **Ctrl** key, then touching **j**.

To log out from the system level, hold down the **Ctrl** key while simultaneously touching **d**.

<i>Function</i>	<i>Keystrokes</i>
(1)	Alt F1 on the function keypad
(2)	Alt F2 on the function keypad
(3)	Alt F3 on the function keypad
(4)	Alt F4 on the function keypad
(5)	Alt F5 on the function keypad
(6)	Alt F6 on the function keypad
(7)	Alt F7 on the function keypad
(8)	Alt F8 on the function keypad
↑	↑ on the numeric keypad
↓	↓ on the numeric keypad
←	← on the numeric keypad
→	→ on the numeric keypad

BACKSPACE	← to the right of + and = and at the top of the keyboard
BEGIN-LINE	Alt ← on the numeric keypad
BOX-MARK	Alt b on the keyboard
BREAK	Scroll Lock in the upper right-hand corner of the keyboard
CANCEL	- to the right of the numeric keypad
CENTER	Alt c on the keyboard
DELETE	Alt Delete below the numeric keypad
DELETE-CHARACTER	Delete below the numeric keypad
DO	Alt x on the keyboard
END-LINE	Alt → on the numeric keypad
ENTER	+ to the right of the numeric keypad
EXECUTE	+ to the right of the numeric keypad
EXIT	Alt d on the keyboard
FONT	Alt f on the keyboard
FORMAT	F3 on the function keypad
FUNCTIONS	Alt h on the keyboard
GO-TO	End on the numeric keypad
HELP	5 on the numeric keypad
HOME	Home on the numeric keypad
INSERT	F4 on the function keypad
INSERT-MODE	Insert below the numeric keypad
LAST-ARG	Alt a on the keyboard
LEFT	Alt l on the keyboard
+LINE	Pg Dn on the numeric keypad
-LINE	Pg Up on the numeric keypad

LINE-FEED	Ctrl j on the keyboard
LOCAL-MENU	F2 on the function keypad
MARGIN	Alt m on the keyboard
MENU	F1 on the function keypad
NEXT	Alt End on the numeric keypad
NEXT-WINDOW	Alt n on the keyboard
+PAGE	Alt Pg Dn on the numeric keypad
-PAGE	Alt Pg Up on the numeric keypad
PICK-COPY	F7 on the function keypad
PICK-UP	F5 on the function keypad
PREVIOUS	Alt Home on the numeric keypad
PRINT	Alt Prt Sc to the right of the keyboard
PUT-COPY	F8 on the function keypad
PUT-DOWN	F6 on the function keypad
QUIT	Ctrl \ on the keyboard
QUOTE	Alt q on the keyboard
REFRESH	Alt z on the keyboard
REPLACE	Alt 5 on the numeric keypad
RESTORE	Alt Insert below the numeric keypad
RETURN	↵ to the right of the keyboard
RIGHT	Alt r on the keyboard
SAVE	Alt s on the keyboard
+SEARCH	Alt ↓ on the numeric keypad
-SEARCH	Alt ↑ on the numeric keypad
SET-TAB	Alt v on the keyboard
TAB	→ to the left of the keyboard
-TAB	Shift ← to the left of the keyboard

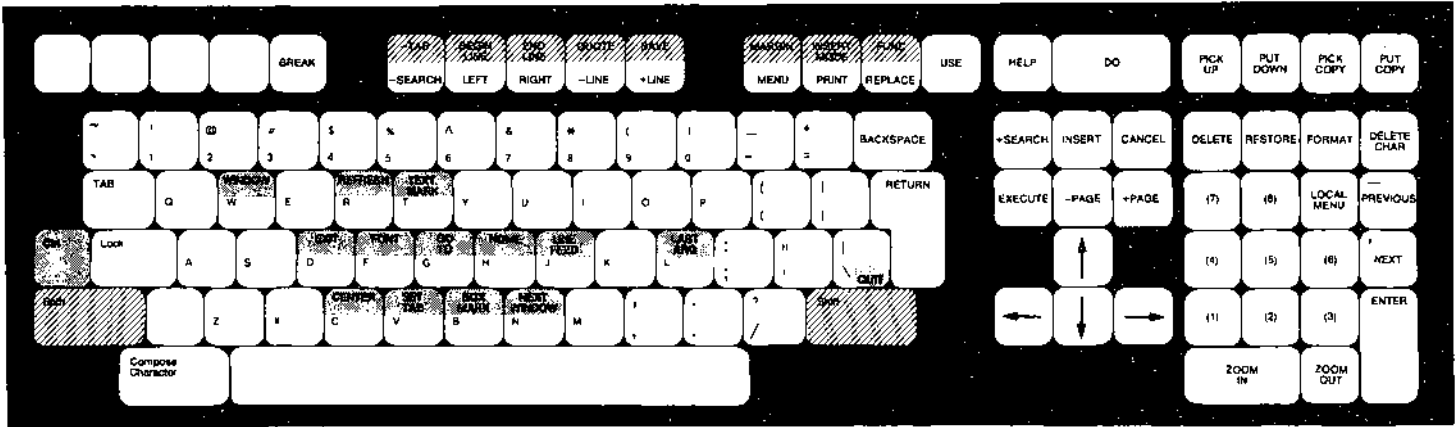
TEXT-MARK	Alt t on the keyboard
USE	Alt u on the keyboard
WINDOW	Alt w on the keyboard
ZOOM-IN	F9 on the function keypad
ZOOM-OUT	F10 on the function keypad


8. VT220 and VT240

8.1 The VT220 and VT240 Keyboards

Figure 7 shows the VT220* and VT240* keyboard layout and function locations for the TEN/PLUS system.





To invoke a function indicated by , hold down the CTRL key and touch the designated key at the same time.

To invoke a function indicated by , hold down the SHIFT key and touch the designated key at the same time.

Figure 7. VT220 and VT240 Keyboard Layout for the TEN/PLUS System

8.2 TEN/PLUS Functions on VT220 and VT240 Keyboards

Functions are accessed in one of three ways on VT240 and VT240 terminals:

1. Touch the defined key. For example, **FORMAT** is invoked by touching PF3 above the numeric keypad.
2. Hold down the **Ctrl** key, then touch the designated key. For example, **CENTER** is invoked by holding down **Ctrl**, then touching the **c** key.
3. Hold down the **Shift** key, then touch the designated key. For example, **SAVE** is invoked by holding down **Shift**, then touching the **F 10** key.

To log out from the system level, hold down the **Ctrl** key while simultaneously touching **d**.

<i>Function</i>	<i>Keystrokes</i>
(1)	1 on the numeric keypad
(2)	2 on the numeric keypad
(3)	3 on the numeric keypad
(4)	4 on the numeric keypad
(5)	5 on the numeric keypad
(6)	6 on the numeric keypad
(7)	7 on the numeric keypad
(8)	8 on the numeric keypad
↑	↑ to the left of the numeric keypad
↓	↓ to the left of the numeric keypad
←	← to the left of the numeric keypad
→	→ to the left of the numeric keypad
BACKSPACE	Word Char in the upper right-hand corner of the keyboard
BEGIN-LINE	Shift F7 above the keyboard
BOX-MARK	Ctrl b on the keyboard

BREAK	Break above the keyboard
CANCEL	Remove to the left of the numeric keypad
CENTER	Ctrl c on the keyboard
DELETE	PF 1 above the numeric keypad
DELETE-CHARACTER	PF 4 above the numeric keypad
DO	Do above the keyboard
END-LINE	Shift F 8 above the keyboard
ENTER	Enter to the right of the numeric keypad
EXECUTE	Select to the left of the numeric keypad
EXIT	Ctrl d on the keyboard
FONT	Ctrl f on the keyboard
FORMAT	PF 3 above the numeric keypad
FUNCTIONS	Shift F 13 above the keyboard
GO-TO	Ctrl g on the keyboard
HELP	Help above the keyboard
HOME	Ctrl h on the keyboard
INSERT	Insert Here to the left of the numeric keypad
INSERT-MODE	Shift F 12 above the keyboard
LAST-ARG	Ctrl l on the keyboard
LEFT	F 7 above the keyboard
+LINE	F 10 above the keyboard
-LINE	F 9 above the keyboard
LINE-FEED	Ctrl j on the keyboard
LOCAL-MENU	9 on the numeric keypad
MARGIN	Shift F 11 above the keyboard
MENU	F 11 above the keyboard

NEXT	, on the numeric keypad
NEXT-WINDOW	Ctrl n on the keyboard
+PAGE	Next Screen to the left of the numeric keypad
-PAGE	Prev Screen to the left of the numeric keypad
PICK-COPY	F 19 above the keyboard
PICK-UP	F 17 above the keyboard
PREVIOUS	- on the numeric keypad
PRINT	F 12 above the keyboard
PUT-COPY	F 20 above the keyboard
PUT-DOWN	F 18 above the keyboard
QUIT	Ctrl \ on the keyboard
QUOTE	Shift F9 above the keyboard
REFRESH	Ctrl r on the keyboard
REPLACE	F 13 above the keyboard
RESTORE	PF2 above the numeric keypad
RETURN	Return on the right of the keyboard
RIGHT	F8 above the keyboard
SAVE	Shift F10 above the keyboard
+SEARCH	Find to the left of the numeric keypad
-SEARCH	F6 above the keyboard
SET-TAB	Ctrl v on the keyboard
TAB	Tab on the left of the keyboard
-TAB	Shift F6 above the keyboard
TEXT-MARK	Ctrl t on the keyboard
USE	F 14 above the keyboard
WINDOW	Ctrl w on the keyboard

ZOOM-IN

0 on the numeric keypad

ZOOM-OUT

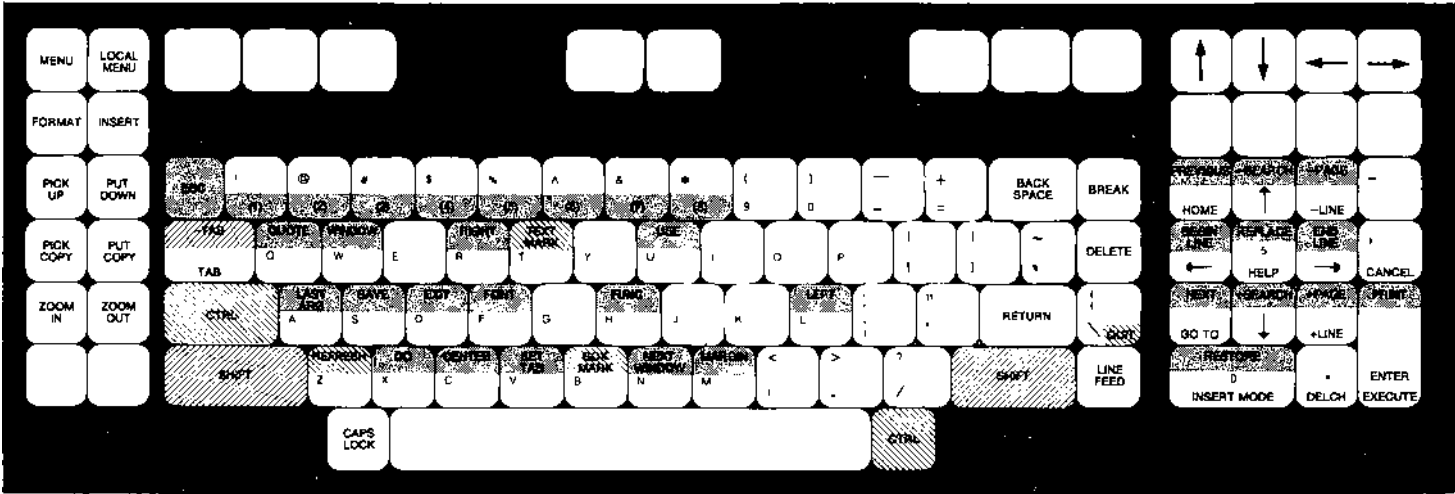
. on the numeric keypad

9. AT&T 5425 and AT&T 4425

9.1 The AT&T 5425 and AT&T 4425 Keyboards


Figure 8 shows the AT&T 5425 and AT&T 4425 keyboard layout and function locations for the TEN/PLUS system.





To invoke TEN/PLUS functions on the left keypad, the left keypad must be in Alternate Keypad mode.

To invoke a function indicated by , hold down the CTRL key and touch the designated key at the same time.

To invoke a function indicated by , touch the ESC key, release it, then touch the designated key.

To invoke a function indicated by , hold down the SHIFT key and touch the designated key at the same time.

Figure 8. AT&T 5425 and AT&T 4425 Keyboard Layout for the TEN/PLUS System

9.2 TEN/PLUS Functions on the AT&T 5425 and AT&T 4425 Keyboards

Functions are accessed in one of four ways on AT&T 5424 and AT&T 4425 terminals:

1. Touch the defined key. For example, **FORMAT** is invoked by touching **F3** on the function keypad.
2. Hold down the **CTRL** key, then touch the designated key. For example, **BOX-MARK** is invoked by holding down **CTRL**, then touching the **b** key.
3. Touch the **ESC** key, release it, then touch the designated key. For example, **SAVE** is invoked by touching **ESC**, releasing it, then touching the **s** key.
4. Hold down the **SHIFT** key, then touch the designated key. For example, **-TAB** is invoked by holding down **SHIFT**, then touching the **TAB** key.

Note that to invoke TEN/PLUS functions on the left keypad, the left keypad must be in Alternate Keypad mode.

To log out from the system level, hold down the **CTRL** key while simultaneously touching **d**.

<i>Function</i>	<i>Keystrokes</i>
(1)	ESC then 1 on the main keyboard
(2)	ESC then 2 on the main keyboard
(3)	ESC then 3 on the main keyboard
(4)	ESC then 4 on the main keyboard
(5)	ESC then 5 on the main keyboard
(6)	ESC then 6 on the main keyboard
(7)	ESC then 7 on the main keyboard
(8)	ESC then 8 on the main keyboard
↑	↑ above the right keypad or 8 on the right keypad
↓	↓ above the right keypad or 2 on the right keypad

←	← above the right keypad or 4 on the right keypad
→	→ above the right keypad or 6 on the right keypad
BACKSPACE	BACKSPACE on the keyboard
BEGIN-LINE	ESC then 4 on the right keypad
BOX-MARK	CTRL b on the keyboard
BREAK	BREAK on the keyboard
CANCEL	, on the right keypad
CENTER	ESC then c on the keyboard
DELETE	DELETE on the main keyboard
DELETE-CHARACTER	. on the right keypad ¹¹
DO	ESC then x on the keyboard
END-LINE	ESC then 6 on the right keypad
ENTER	ENTER on the right keypad
EXECUTE	ENTER on the right keypad
EXIT	ESC then d on the keyboard
FONT	ESC then f on the keyboard
FORMAT	F3 on the function keypad
FUNCTIONS	ESC then h on the keyboard
GO-TO	1 on the right keypad
HELP	5 on the right keypad
HOME	7 on the right keypad ¹²

11. Do not confuse this key with DEL CHAR on the left keypad.

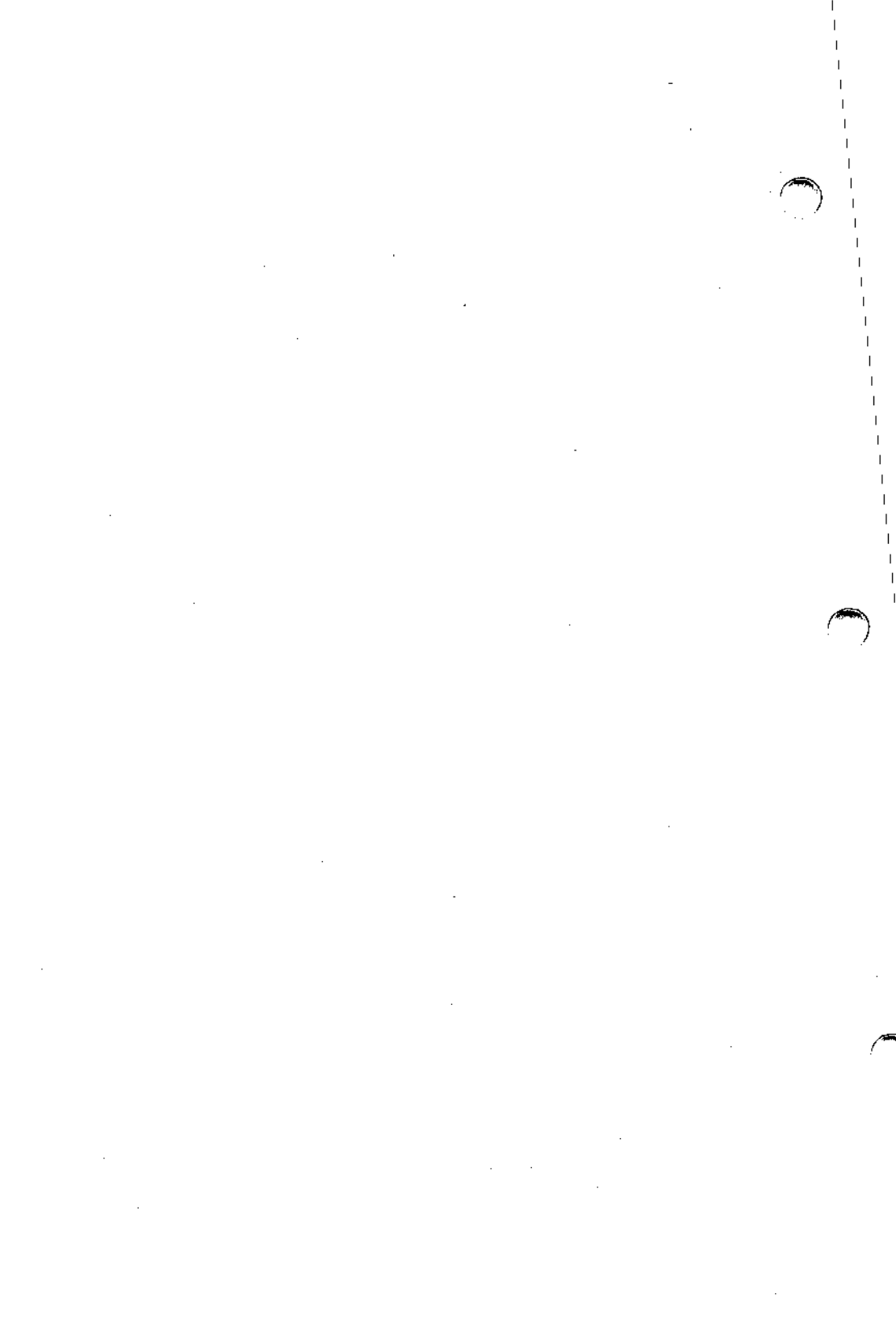
12. Do not confuse this key with HOME on the left keypad.

INSERT	F4 on the function keypad
INSERT-MODE	0 on the right keypad
LAST-ARG	ESC then a on the keyboard
LEFT	ESC then 1 on the keyboard
+LINE	3 on the right keypad
-LINE	9 on the right keypad
LINE-FEED	LINE FEED on the keyboard
LOCAL-MENU	F2 on the function keypad
MARGIN	ESC then m on the keyboard
MENU	F1 on the function keypad ¹³
NEXT	ESC then 1 on the right keypad
NEXT-WINDOW	ESC then n on the keyboard
+PAGE	ESC then 3 on the right keypad
-PAGE	ESC then 9 on the right keypad
PICK-COPY	F7 on the function keypad
PICK-UP	F5 on the function keypad
PREVIOUS	ESC then 7 on the right keypad
PRINT	ESC then PRINT on the right keypad ¹⁴
PUT-COPY	F8 on the function keypad
PUT-DOWN	F6 on the function keypad
QUIT	CTRL \ on the keyboard
QUOTE	ESC then q on the keyboard
REFRESH	CTRL z on the keyboard

13. Do not confuse this key with MENU above the keyboard.

14. Do not confuse this key with PRINT above the keyboard.

REPLACE	ESC then 5 on the right keypad
RESTORE	ESC then 0 on the right keypad
RETURN	RETURN on the right of the keyboard
RIGHT	ESC then r on the keyboard
SAVE	ESC then s on the keyboard
+SEARCH	ESC then 2 on the right keypad
-SEARCH	ESC then 8 on the right keypad
SET-TAB	ESC then v on the keyboard
TAB	TAB on the left of the keyboard
-TAB	SHIFT TAB on the left of the keyboard
TEXT-MARK	CTRL t on the keyboard
USE	ESC then u on the keyboard
WINDOW	ESC then w on the keyboard
ZOOM-IN	F9 on the function keypad
ZOOM-OUT	F10 on the function keypad



Appendix: TERMINAL DESCRIPTION FILES

This appendix contains technical information about the terminal description files located in the “termcaps” database. It is included as a reference for system programmers; it is not necessary for you to read this appendix if you are not a system programmer.

The INed* editor is designed to operate on most ASCII video (VDT or CRT) terminals and on a number of personal computers. The standard version of the editor uses data in the “termcaps” terminal description database to run the terminal. Many terminals manufactured have been described in this database. (Note that this appendix applies only to those terminals that are used with the standard version of the editor. Skip this appendix if your computer or terminal does not use this version of the editor.)

The termcap database is located in the file `/etc/termcap`. The editor does not use this file directly, but instead uses the file `terms.bin`. This file contains all the information in the termcap database, as well as information about where the TEN/PLUS function keys are located on the keyboard. All of this information is stored in a binary form for use by the editor.

The file `trm` is an editable version of `terms.bin`. You can compose it directly or use the program `tconvert(1)` to make `trm` from `/etc/termcap`. If you produce `trm` from `/etc/termcap`, the keyboard definition is taken from record 0 of `$(SYS)/termcap/def.trm`. Figure 9 shows the relationship between these files. All files other than `/etc/termcap` are located in the directory `$(SYS)/termcap`.

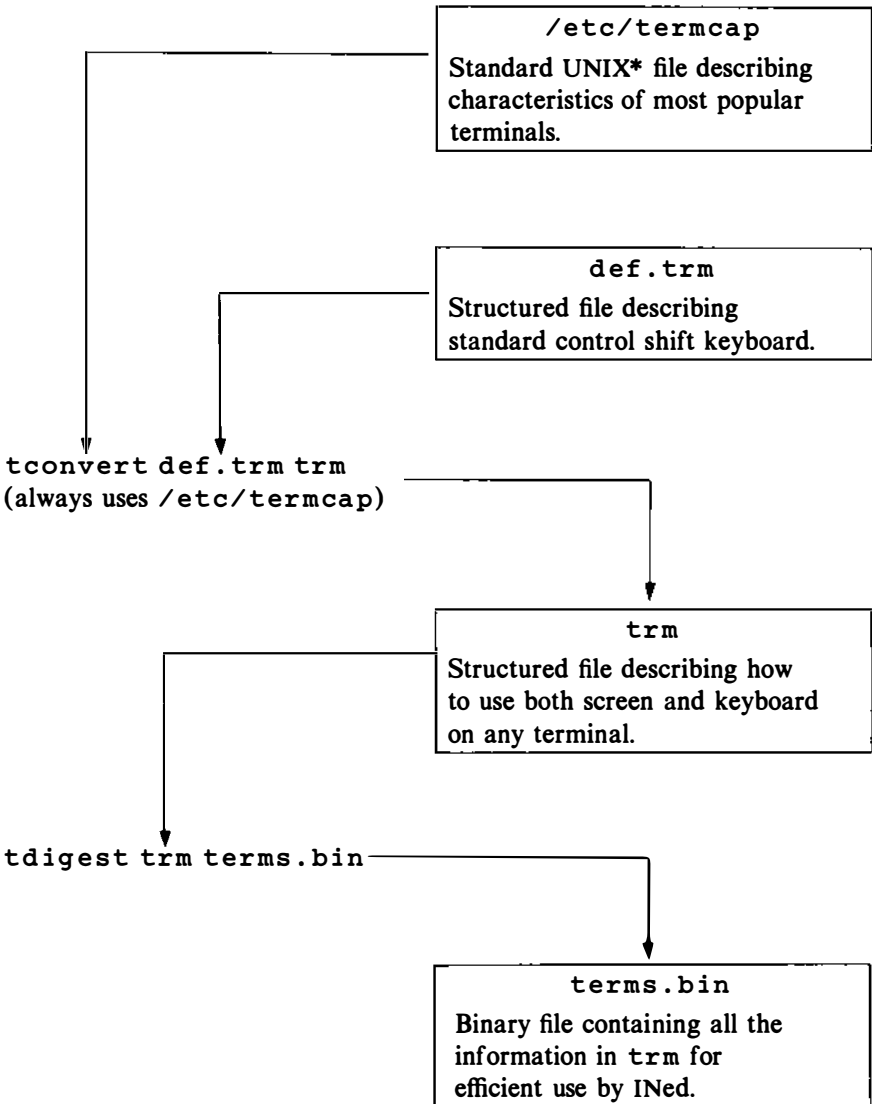


Figure 9. Relationship of Terminal Description Files

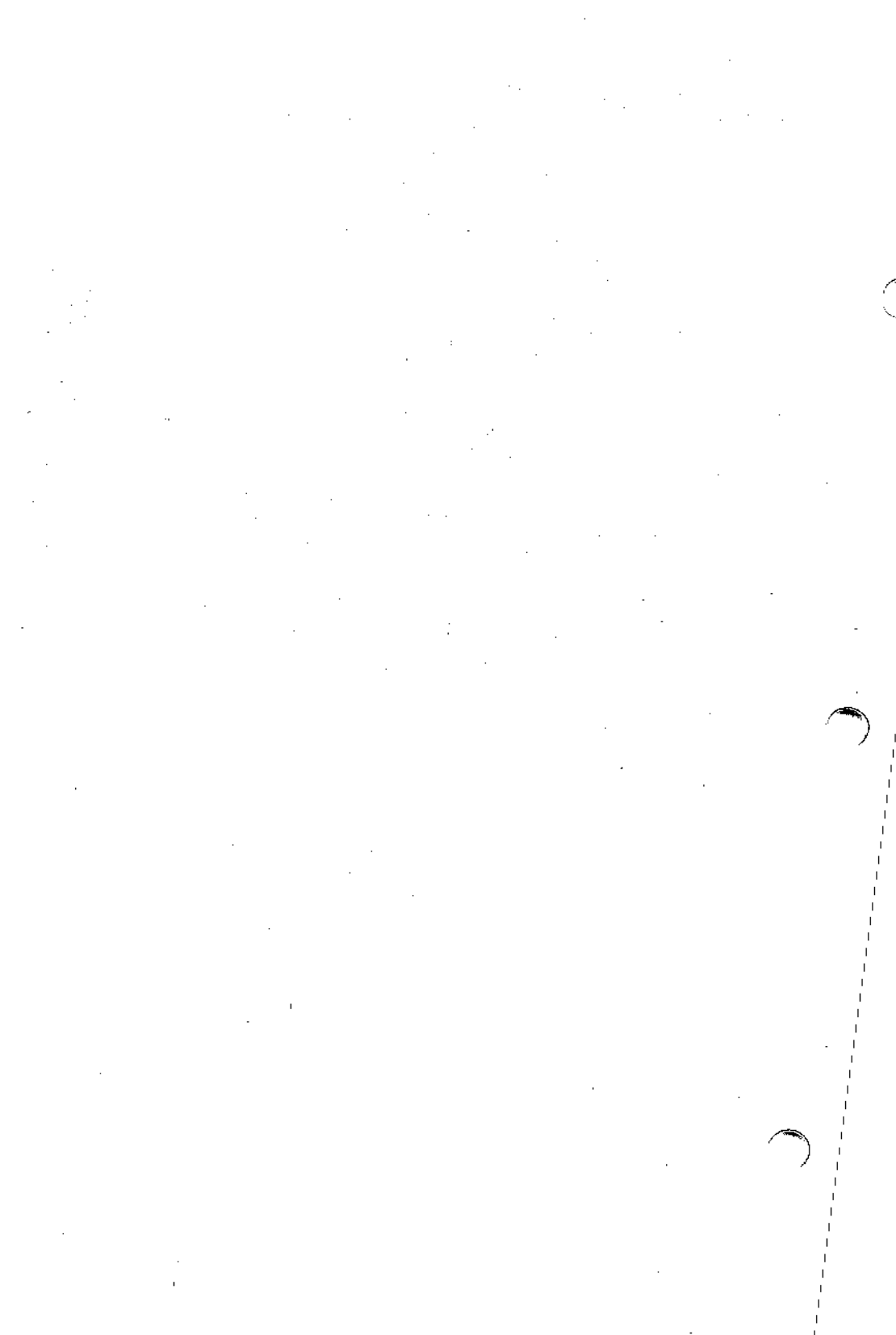
Typical usage is to make `trm` from `/etc/termcap`, then modify the keyboard mapping for some terminals to make use of special terminal keyboard features, such as extra keys.

The file `def.trm` is a structured file that describes all supported terminals. To see how a terminal is described, access `def.trm`, move the cursor to the appropriate line in the `Output Sequences` column, and **ZOOM-IN**.

The left-hand column lists the names of output sequences. The middle column gives a brief description of what the output sequence does. The right-hand column gives the codes implementing the sequence for the terminal. The sequence names and the representation for the codes are taken directly from the notation used in the `termcap` database.

The input sequences are shown when you **ZOOM-IN** to a terminal description with the cursor in the `Input Sequences` column. The notation for the input and output code sequences is similar to that used in `/etc/termcap`.

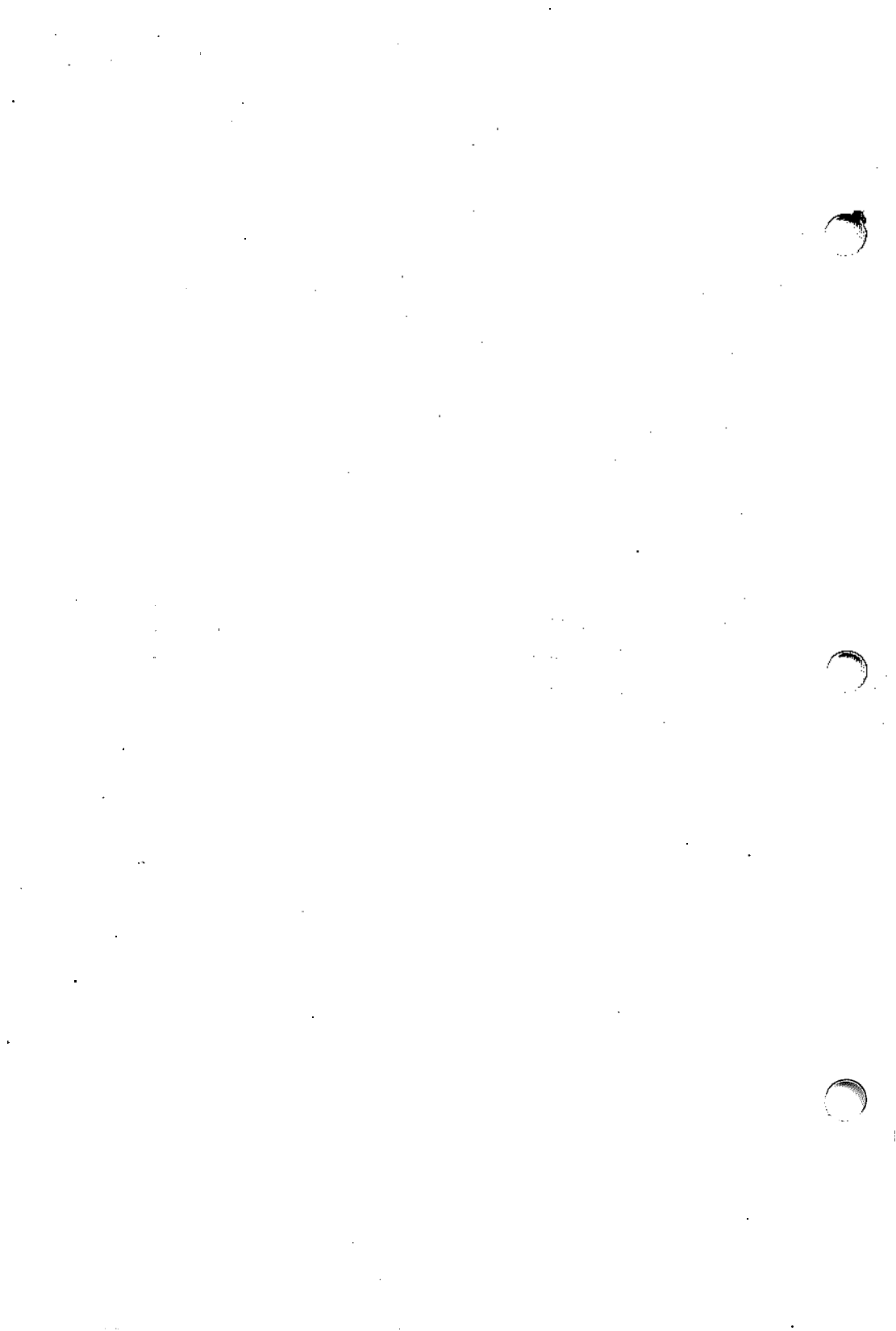
Before entering the editor, set the environment variable `TERM` to the type of the appropriate terminal. To avoid setting the `TERM` variable at each login, set it in an individual profile file in each user's login directory (e.g., `$HOME/.profile`).



TEN/PLUS Profiles

CONTENTS

1. INTRODUCTION	1
1.1 Accessing TEN/PLUS Functions	1
2. THE EDITOR PROFILE	2
2.1 Creating Your Editor Profile	2
2.2 The Top Level of the Editor Profile	2
2.3 MENU Options	3
2.4 HELP Options	6
2.5 Telling the Editor to Watch Specific Files	7
2.6 Editor Search Paths	8
3. THE PRINT PROFILE	10
3.1 Creating Your Print Profile	10
3.2 The Top Level of the Print Profile	11
3.3 Print Option Details	12
3.4 Changing Your Print Profile	13
4. THE FILE MANAGER PROFILE	14
4.1 Creating Your File Manager Profile	14
4.2 The Top Level of the File Manager Profile	14



TEN/PLUS* Profiles

1. INTRODUCTION

This document is intended as a training guide for those who wish to learn how to use TEN/PLUS profiles to customize their TEN/PLUS environment. It is intended for those who have already used the TEN/PLUS system to create, edit, and manage text files. Readers of this document should refer to the "TEN/PLUS Primer," the "TEN/PLUS Tutorial," and the "TEN/PLUS Reference Manual" in this guide, as appropriate to their needs.

1.1 Accessing TEN/PLUS Functions

Accessing TEN/PLUS functions requires different keystroke sequences on different keyboards. Refer to the appropriate section of "TEN/PLUS Keyboard Information" for an alphabetic listing of the TEN/PLUS functions and the keystroke sequences required for your keyboard.

2. THE EDITOR PROFILE

The editor profile file, `editorprf`, is used to customize the editor to suit the needs of individual users. It is used to override the default specifications in the standard editor profile intended for beginning users. The editor profile is used to specify:

- What the menu for `MENU` will look like.
- What the menu for `HELP` will look like.
- Which files the editor should watch.
- Which directories the editor should search to locate forms, helpers, messages, and forms language scripts.

On versions of the TEN/PLUS system that support color displays, an option is available that allows users to specify the colors used for text and background.

2.1 Creating Your Editor Profile

To create your editor profile file, use `MENU` and select the option `Edit your editor profile`. Popup boxes will appear, indicating that the system is creating a `profiles` directory (if one does not already exist) and a standard editor profile file, `editorprf`, in that directory. After a brief pause, the top level of the `editorprf` file will appear on the display.

2.2 The Top Level of the Editor Profile

The top level of the editor profile looks like this:

```

Editor Profile File

This file allows you to modify the behavior of the editor to suit your
preferences. Put the cursor on the line of interest and ZOOM-IN to specify
your choice of options.

MENU Options
HELP Options
Files the Editor Should Watch
Editor Search Paths

```

Additional profile options may be present, depending upon the configuration of your system. You can explore the profile by using `ZOOM-IN`, `ZOOM-OUT`, and the cursor-positioning functions.

2.3 MENU Options

If you **ZOOM-IN** to the first option, MENU Options, you will see a screen that looks like this:

MENU Options		
Description shown in menu	Type	Name of file or program
Show home directory	file	\$HOME
Execute UNIX shell commands	screen	echo "Touch CNTL-D to continue ed
Run a shell command in a box	popbox	"Shell_command:"
Show your profiles directory	file	\$HOME/profiles
Edit your editor profile	file	\$HOME/profiles/editorprf
Housekeep	popbox	housekeep
Display history of current file	helper	history

(Note that you can use **RIGHT** to see additional text in the Name of file or program field.)

This screen is used to determine what the New Task Menu, which is displayed by **MENU**, will look like. Each line corresponds to one line of the New Task Menu. Sample text menu entries can be found by using **HELP** and selecting the Suggestions for Your MENU option. If you do not know how to set up a menu entry, you should look at the suggestions to get some ideas on how it is usually done. The best way to get started is to use **PICK-COPY** to pick up some of the lines, and then use **PUT-DOWN** to put them in your editor profile. If that is all you want to do for now, you can proceed to §2.4.

The first field on the MENU Options screen, Description shown in menu, contains the text that will appear in the New Task Menu. The next field, Type, determines what kind of task is to be performed when this option is selected. Possible Type entries are:

- form:
change to a new form
- file:
change to a new file
- helper:
change to a new helper

screen:

clear the screen and run a program

popbox:

run a program and put its output in a popup box

The last field on the **MENU Options** screen, **Name of file or program**, contains the name of the file, helper, or program. This field can contain shell variables and user prompts. A shell variable is denoted by typing a dollar sign (\$) followed by the variable name. The editor also understands these special variable names:

\$FORM:

the name of the current form

\$FILE:

the name of the file you are editing

\$ALTFILE:

the name of the alternate file

\$HELPER:

the name of the current helper

\$SYS:

the directory where editor helpers, forms, and help files are installed

\$LANG:

the directory in which a foreign language version of forms, help messages, and scripts can be found

If the **Type** is **screen** or **popbox**, the last field is passed to the shell for processing. This means that the command can actually be a pipeline—several programs separated by the pipe character (|). It can also specify redirection of standard input and/or standard output (>, >>, <). Note, however, that if the **Type** is **popbox**, the last field cannot contain an interactive command (one requiring additional input from the user). If an interactive command is accidentally provided, try using **BREAK** or **QUIT** to interrupt the process when the option is selected.

User prompts are strings of underlined characters. When the editor processes a menu item, it puts each prompt into a popup box. Before executing the requested action, the editor replaces the prompt with whatever the user types into the box. If the user uses **CANCEL**, the editor returns to normal editing. For example,

assume that your system has a program called `phone` that takes one argument, the name to look up in a phone directory. The menu line to run this program might look like this:

```
| Look up a phone number | popbox | phone Enter name:
```

When you select this item, the editor displays this popup box:

```
Enter name:
```

If you type in `Bob` and `EXECUTE`, the editor runs the program `phone Bob`. Note that the prompt contains an underlined space character. This is used to display a multiword prompt. You can also display several prompts by separating them with nonunderlined space characters; for example, `command arg1 arg2 arg3 last_arg`.

If you `ZOOM-IN` to the first line on the MENU Options screen, the editor displays the next level of detail:

Details of MENU Option		
Description shown in menu	Type	Name of file or program
Show home directory	file	\$HOME
Flags: any non-space character means true		
<input type="checkbox"/> Sync and reopen file	<input type="checkbox"/>	Save all files

The `Sync and reopen file` box is used when a menu item alters a file in any way. An entry in this box causes the system to save the file, run the program, and then reopen the file. For example, to remove the history from the current file, you could use the `rmhist(1)` command on the current file and type any character in the `Sync and reopen file` box:

Details of MENU Option		
Description shown in menu	Type	Name of file or program
Remove history from current file	popbox	rnhist \$FILE
Flags: any non-space character means true		
<input checked="" type="checkbox"/> Sync and reopen file	<input checked="" type="checkbox"/>	Save all files

The **Save all files** box is used for saving all ASCII text files. The editor maintains special files that track the changes made to ASCII text files and, on exiting, converts these special files back to regular ASCII files. If you are running the shell using a menu item with the **Type** field set to **screen**, you should set this flag by typing any character into the **Save all files** box. This slows down the processing of the menu item (if you have edited any ASCII text files), but it is important when using the shell. The **Save all files** flag ensures that if you run another program (such as *grep(1)*) from the shell on a text file that you have edited, the latest version of this file is processed.

2.4 HELP Options

To access the **HELP Options** part of your profile, **ZOOM-OUT** to the top level of your profile, move the cursor down to **HELP Options**, then **ZOOM-IN**:

HELP Options		
Description shown in menu	Type	Name of file or program
Alphabetic List of Editor Commands	file	\$\$SYS/help/e.cmds
How do I ... ?	file	\$\$SYS/help/e.howtos
How to Customize the Editing System	file	\$\$SYS/help/editorprf.hdq
Suggestions for Your MENU	file	\$\$SYS/help/emenu
Suggestions for Print Menu	file	\$\$SYS/help/prthelp
Keyboard Layouts	file	\$\$SYS/help/keys.map

The **HELP Options** screen is used to determine what the menu for **HELP** will look like. The first field, **Description shown in menu**, contains the description that appears in the menu. The second field, **Type**, is usually **file**; it tells the editor to switch to the corresponding file when this item is

selected. The third field, **Name of file or program**, contains the name of the help file to be displayed.

To add a **HELP** menu option, create a help file and add the appropriate line on this screen. The help file need not be installed in `$SYS/help`.

2.5 Telling the Editor to Watch Specific Files

The editor can be set up to watch for changes to specific files. The editor is usually set up to watch either mail or reminder files for new mail or new reminders, but you can have the editor watch any file. When the editor notices that one of these files has changed, it prints a message in a popup box.

To specify the files to be watched, **ZOOM-IN** to the **Files the Editor Should Watch** option at the top level of your editor profile:

Files the Editor Should Watch		
Name of file	Message to display	Program to run
<code>\$HOME/.reminder</code>		<code>\$SYS/bin/showrem -d</code>

The first field, **Name of file**, should contain the name of the file the editor is to watch. This file name can contain shell variables (§2.3). The second field, **Message to display**, should contain the message the editor will display when the file changes. The third field, **Program to run**, should contain the name of the program to run when the file changes.

An entry must be made in the **Name of file** field and at least one of the other fields. If there is an entry in the **Message to display** field and not in the **Program to run** field, the indicated message appears in a popup box when the editor notices that the file has changed. If there is an entry in the **Program to run** field and not in the **Message to display** field, the editor runs the indicated program and prints the output from the program in a popup box. If there is an entry in both fields, first the indicated

message appears in a popup box, then the editor runs the indicated program. When the program completes, the original popup box disappears and the output from the program appears in a new popup box.

Here is a sample watch files screen with two sample entries:

Files the Editor Should Watch		
Name of file	Message to display	Program to run
\$HOME/.smail \$HOME/.reminder	You have new mail	\$\$SYS/bin/newmail \$HOME/.s \$\$SYS/bin/showrem -d

The first sample entry monitors your incoming mail, if your incoming mail is put in `$HOME/.smail`. When the editor discovers that the file has changed, the editor displays the message `You have new mail` in a popup box, then runs the `newmail` program. The `newmail` program displays the header lines of the new mail in a popup box. You can use *mail* or the mail helper to read your new messages.

The second sample entry monitors the reminder file. When the editor discovers that the file `$HOME/.reminder` has changed, the `showrem` program runs and displays the contents of the `$HOME/.reminder` file in a popup box.

2.6 Editor Search Paths

The `Editor Search Paths` option at the top level of the editor profile is used for specifying the search paths that the editor uses to find forms, helpers, messages, and scripts. If you **ZOOM-IN** to this option, you will see:

Editor Search Paths

This section is used to specify the places the editor should look for forms and helpers. Each line in each box should be the name of a directory; the directories are searched from top to bottom.

Forms	Helpers
<code>\$\$SYS/\$LANG/forms</code>	<code>\$\$SYS/helpers</code>
Messages	Scripts
<code>\$\$SYS/\$LANG/hmgs</code>	<code>\$\$SYS/\$LANG/scripts</code>

To make your own forms directory, insert a line at the top of the Forms field and type a directory name; for example, `$HOME/forms`. This tells the editor to look for a directory named `forms` under your login directory before it checks the standard system directory. You can then put your own forms (or personalized versions of the standard editor forms) in your own forms directory, and the editor will automatically use these. This procedure also is used to make your own helper, message, and script directories.

3. THE PRINT PROFILE

The standard Print Menu includes four options:

```
Print on default printer
Print (ask for options)
Print to file (overwrite)
Print to file (append)
```

The Print Helper, invoked by using **PRINT**, creates a temporary file that represents a combination of the text on the screen and the form through which that text is viewed. **PRINT** also utilizes the data in the print profile to generate a menu of options describing how the temporary file is processed. Typically, the temporary file is stored in another file, for subsequent disposition, or is output to a printing device.

You can include additional menu options by modifying the print profile file, `printprf`, located in the `profiles` directory. Sample print profile entries can be found by selecting the option **Suggestions for Print Menu** option from the Help Menu.

3.1 Creating Your Print Profile

To create your print profile file, use **ENTER** `$HOME/profiles/printprf` **USE**, and select the option to create a structured file. If a `profiles` directory does not already exist in your home directory, one will be created. After a brief pause, the top level of the print profile will appear on the display:

Print Options	
ZOOM-IN to see more detail about command.	
Description shown in menu	Command
Print on default printer	lp -t=\$FILE
Print (ask for options)	lp <u>Print options:</u> \$FILE
Print to file (overwrite)	>
Print to file (append)	>>

Note that your system may utilize a different print command. Check with your system administrator or refer to your user's manual for the correct command to insert in your menu.

3.2 The Top Level of the Print Profile

The top level of the print profile has two fields: **Description shown in menu** and **Command**. The **Description shown in menu** field is used to enter the description that displays on the menu when **PRINT** is used. The **Command** field is used to enter print output instructions.

Four types of entries can be entered in the **Command** field:

- UNIX* System Commands
- Pipe Commands (|)
- Redirect and Overwrite Commands (>)
- Redirect and Append Commands (>>)

The first column in the **Command** field can contain a pipe symbol (|), a redirect symbol (> or >>), or a UNIX System command. If the first column contains a pipe symbol, the output is piped through the specified program. If the first column contains a redirect symbol, the output is redirected to the specified file. If the first column contains a UNIX System command, the specified command is executed.

If the **Command** field is left blank, the Print Helper prompts for a UNIX command. If a file or program name is not entered after a pipe or redirect symbol, the Print Helper prompts for the missing information.

The output instructions entered in the **Command** field can contain shell variables and user prompts. Shell variables are denoted by typing a dollar sign followed by the variable name. The Print Helper understands all editor environment variable names, and two additional variable names that are specific to the Print Helper:

\$PRTCMD:

the expanded command (used only in the **Description for popbox**)

\$PRTFILE:

the name of the temporary print output file (useful in shell commands)

User prompts are strings of underlined characters. When the editor processes a command with a prompt, it displays the prompt in a popup box. The editor replaces the prompt with whatever the user types into the box before executing the command. If the command is not entered in full, the editor prompts for the command input

required. **CANCEL** can be used to remove the popup box and cancel the operation of the command.

The first command in the default print profile displayed above pipes the **PRINT** output through *print(1)* and the title on the header page is set to the name of the file currently being edited (**\$FILE**). The second command pipes the **PRINT** output through *print*, sets the title on the header page to the name of the file currently being edited (**\$FILE**), then prompts for print options. The third command prompts for a file name and overwrites the specified file with the output. If the redirection symbols are followed by a file name, the system automatically prints to the specified file. The fourth command prompts for a file name and appends the output to the specified file. You can explore any of the options listed at the top level of the print profile by using **ZOOM-IN**.

3.3 Print Option Details

When you **ZOOM-IN** to an option listed at the top level of the print profile, the details for the option appear on the display. For example, when you **ZOOM-IN** to the first option, **Print on default printer**, the screen display looks like this:

Details of Print Option	
Description shown in menu	Command
Print on default printer	! print -tl=\$FILE
Description for popbox	
Printing on the default printer	
Flags: any non-space character means true	
<input type="checkbox"/> Save all ASCII files	<input type="checkbox"/> Print graphics box (use w/ printer filter)
<input type="checkbox"/> Clear screen and run command	<input type="checkbox"/> Display all output of command

The first two fields, **Description shown in menu** and **Command**, display the detail from the top level of the profile. The **Description for popbox** field is used to enter the text that will appear in a popbox when the command is executing; it can contain any of the editor environment variables, **\$FILE**, **\$PRTFILE**,

or \$PRTCMD (which expands to the command entered in the Command field). The Save all ASCII files flag should be set if all ASCII files opened during the editing session are to be saved before the command is executed. (If this flag is not set, an attempt to print an ASCII file may not print the most recent version.) The Clear screen and run command flag should be set if the screen should be cleared before the command is executed. The Print graphics box flag should be set if the **PRINT** output is to be run through a filter that will convert graphic characters to characters that the printer understands. Usually, **PRINT** converts the graphics box characters to + and - characters. If this flag is set, this conversion is not performed. The Display all output of command flag should be set if the command output should appear in a popup box.

3.4 Changing Your Print Profile

You can add new options to your Print Menu by editing your `printprf` file. Sample print profile entries can be found by using **HELP** and selecting the Suggestions for Print Menu option from the menu. You should examine these suggestions and use **PICK-COPY** and **PUT-DOWN** to move the desired options to your own `printprf` file.

4. THE FILE MANAGER PROFILE

The file manager profile file, `indexprf`, allows you to customize your directory listing. It is used to specify the directory with which the index file should be synchronized, the directory for deleted files, and the files that should be hidden.

4.1 Creating Your File Manager Profile

To create your file manager profile file, use **ENTER** `$HOME/profiles/indexprf` **USE**, and select the option to create a structured file. If a `profiles` directory does not already exist in your home directory, one will be created. After a brief pause, the top level of the index profile will appear on the display:

```

Directory Helper Options

Synchronize the index file with the directory: x

Directory for deleted files: $HOME/.putdir

Files
to be
hidden
    *.bak
    *.old
    *.index
    .*
  
```

4.2 The Top Level of the File Manager Profile

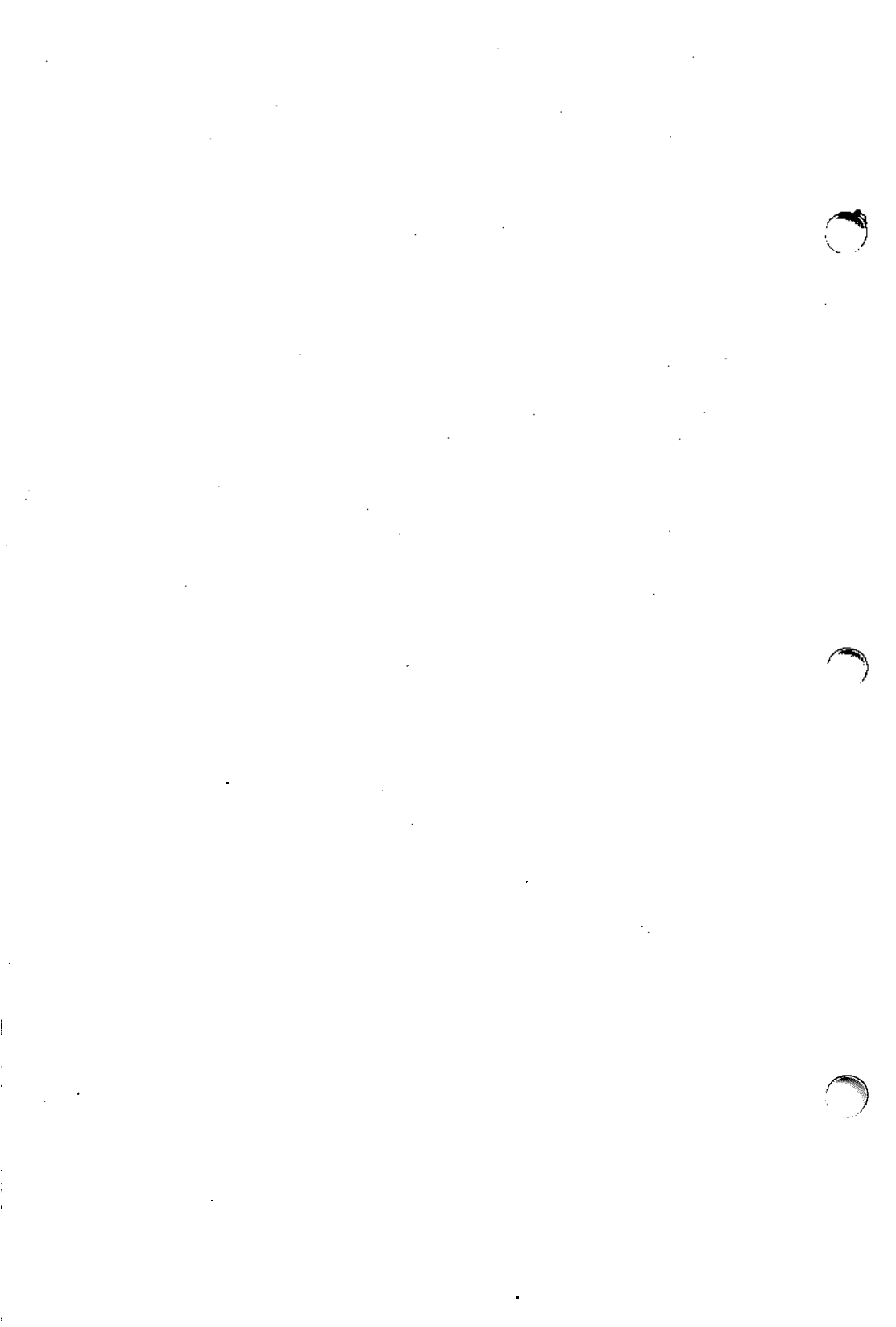
The first option in the top level of the file manager profile allows you to specify whether the directory listing will be updated (that is, synchronized with the actual state of the system) whenever you create files through the **ENTER** `filename` **USE** sequence. The `x` in the first field specifies that synchronization will take place; this is the default. Unless there is an `x` in this field, synchronization will not take place. (Note that any other character will *not* cause synchronization to take place.) In that case, files created through the **ENTER** `filename` **USE** sequence will not show in the directory listing until you manually list them by selecting **Display visible files from the LOCAL-MENU**.

The second option specifies the directory in which the system is to place your deleted files and directories. The default is `$HOME/.putdir`. You can change this default by replacing it with a new directory name. For example, you might decide to place

deleted files and directories in a directory called `removed` in your home directory. To do this, type the path name `$HOME/removed` over the existing path name in this option.

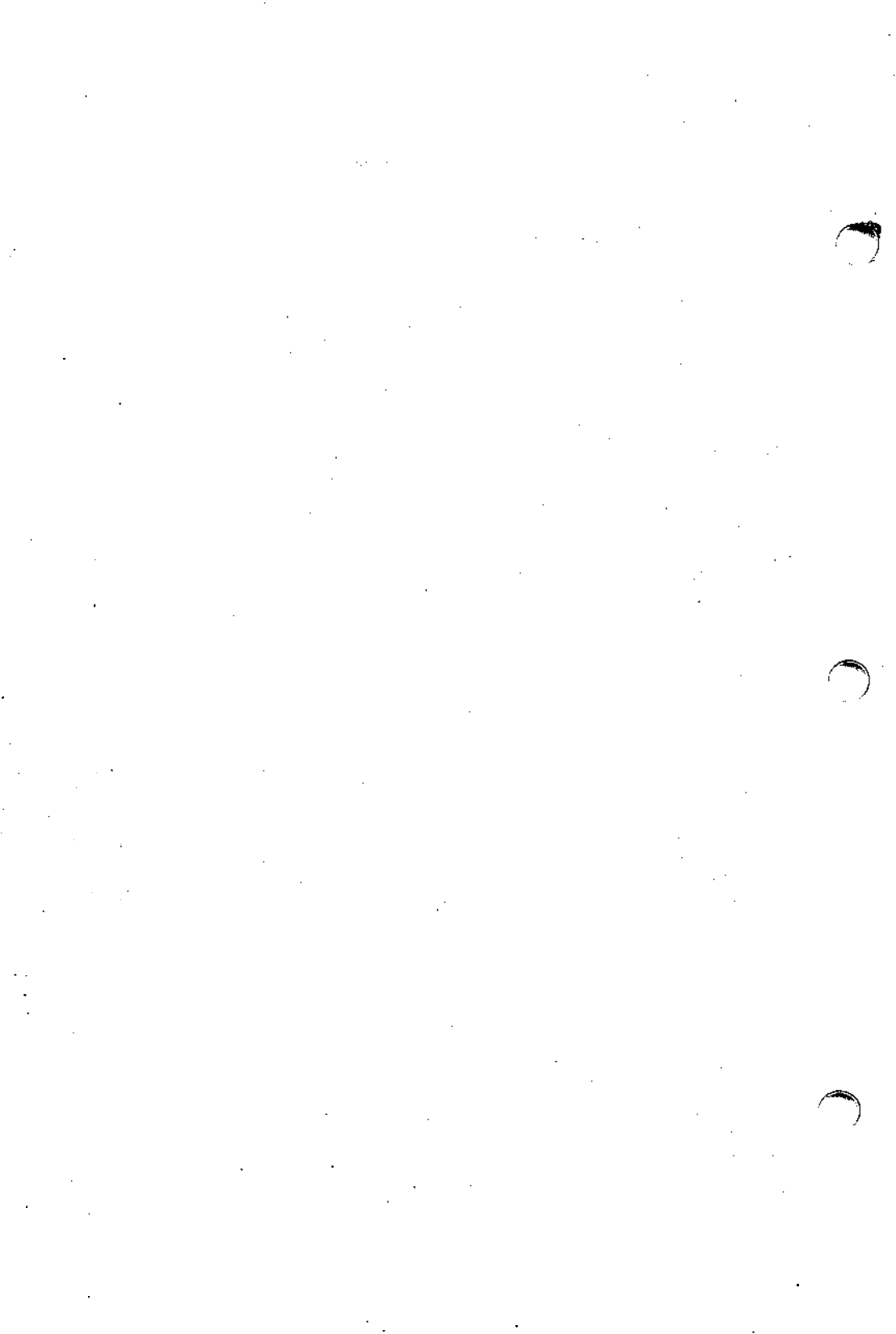
The third option specifies files to be hidden. By default, all files beginning with a dot (`.`) or ending in `.bak`, `.old`, or `.index` will be hidden. The asterisk (`*`) you see in the display is used as a wild card, which means that it is equivalent to any character or characters. Files ending in `.bak` are created by the system each time you complete an editing session on an ASCII file. The previous version of each of these files is saved under a new name, created by truncating the file name as necessary and appending `.bak`. Any previous `.bak` file is overwritten. A `.index` file is a structured file created and used by the File Manager to edit a directory. You cannot access a `.index` file directly.

You can specify which files are to be hidden in your directory listing by modifying the list of file names contained in this option. However, you can always see a listing of all files by using **LOCAL-MENU** and choosing option (2) `Display all files`.



INDEX

- \$ALTFILE 4
- .bak file 15
- CANCEL 12
- command, pipeline 4
- commands, print profile 11
- deleted files and directories, storing 14
- directory listing, synchronizing 14
- editor, customizing 2
- editor profile 2
- editor profile, creating 2
- editor profile, MENU Options 3
- editor profile, top level 2
- editor profile, using 2
- editor search paths 8
- editor search paths, establishing 9
- Editor Search Paths option 8
- editorprf 2
- environment variable 11, 12
- \$FILE 4, 12
- file, hidden 15
- file manager profile 14
- file manager profile, creating 14
- file manager profile, top level 14
- file, previous versions 15
- Files the Editor Should Watch option 7
- files, watch 7
- \$FORM 4
- HELP, adding menu options 7
- HELP, menu options 6
- HELP Options 6
- \$HELPER 4
- hidden file 15
- hidden files, specifying 15
- history, removing 5
- .index file 15
- indexprf 14
- indexprf, creating 14
- indexprf, top level 14
- menu entries, sample 3
- menu for PRINT 10
- MENU Options, editor profile 3
- menu options, HELP 6
- MENU Options, Type entries 3
- New Task Menu 3
- .old file 15
- pipeline command 4
- PRINT 10
- PRINT, adding menu options 10
- print file, temporary 10
- Print Helper 10
- PRINT, menu 10
- print option details 12
- print profile 10
- print profile, adding menu options 13
- print profile commands 11
- print profile, creating 10
- print profile, default 12
- print profile entries, sample 10
- print profile, modifying 10
- print profile, sample entries 13
- print profile, top level 11
- printprf 10
- prompt, multiword 5
- prompt, user 4, 11
- \$PRTCMD 11, 12
- \$PRTFILE 11, 12
- .putdir directory 14
- redirection of standard input 4
- removing history 5
- search paths, editor 8
- search paths, establishing editor 9
- shell variable 4, 11
- shell variable, denoting 4, 11
- standard input, redirection 4
- storing deleted files and directories 14
- synchronizing directory listing 14
- \$SYS 4
- user prompt 4, 11
- variable, denoting a shell 11
- variable, environment 11, 12
- variable names, special 4
- variable, shell 4, 11
- watch files 7
- watch files, establishing 7
- wild cards 15

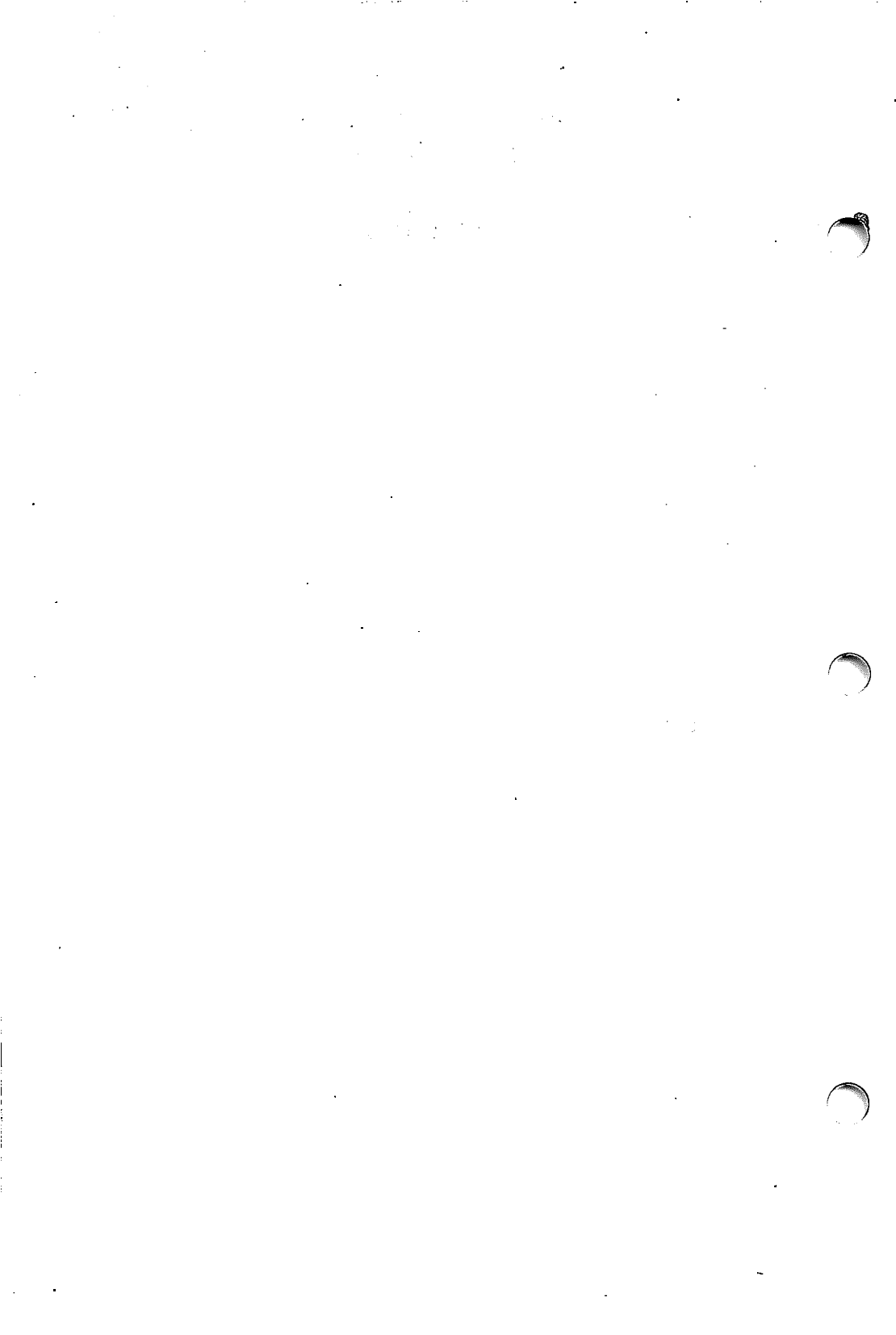


TEN/PLUS User Interface

Manual Entries

CONTENTS

catsf(1)
e(1)
fc(1)
fill(1)
filters(1)
ghost(1)
history(1)
newfile(1)
prtsf(1)
prtty(1)
readfile(1)
rmhist(1)
rpl(1)
sortsf(1)
tconvert(1)
tdigest(1)
versions(1)
ined(4)
terms(4)



NAME

catsf – concatenate structured files

SYNOPSIS

catsf [*-c*[*criteria*]] [*-rn*] [*-s*] *files* *outputfile*

DESCRIPTION

The *catsf* utility catenates together all the records of files and writes the results in to *outputfile*. It can be used to combine structured files. No input file may be the same as the output file.

The option *criteria* selects which records to write; if no *criteria* are given, all records are selected. If the *-c* option is given alone, *catsf* reads the standard input for the *criteria*. If the *-r* option is used, only the record number specified by *n* is selected. The *-s* option causes *catsf* to be silent about nonexistent files.

The *criteria* option can contain expressions in the following form:

```

criterion:: pattern is in PATH |
             pattern is not in PATH;
criteria::  criterion |
            ( criteria ) |
            criteria and criterion |
            criteria or criterion;

```

where *pattern* is a string that can be compiled by *regex(3)*, for example, **Frank**, **2/*/84**, **\${6-9}[0-9]{4}**, and *PATH* is the TEN/PLUS path to the node to be checked for the value.

A sample *criteria* might contain:

```

(California is in Address/*/State or
 Nevada is in Address/*/State ) and
(Peter is in Name or
 Paul is in Name or
 Mary is in Name)

```

This example selects all records that have Peter, Paul, or Mary in the *Name* field and California or Nevada in the *State* field.

BUGS

The *catsf* utility does not attempt to verify that the record structures of the various *inputfiles* are consistent; a bizarre *outputfile* will result if they are not.



NAME

e – INed screen editor

SYNOPSIS

e [[-ar] filename [line [col [searchkey]]]]

DESCRIPTION

The e command invokes the INed screen editor. INed may be initialized in several ways, depending on the arguments given to the e command.

e *filename*

Initialize INed at the first page of the indicated file. If the file does not exist, an instruction box indicating “You are attempting to create file: *filename*.” appears.

e Initialize INed to the file and cursor position displayed the last time the user exited from INed. If multiple windows were in use, only the file in the last active window is displayed.

The file specifier *filename* may consist of up to four arguments, as follows:

filename

Specifies a file.

filename line

Specifies the line number (*line*) at which the cursor is to be positioned. If zero is specified, line one is assumed.

filename line col

Specifies the line number (*line*), and column number (*col*) where the cursor should be initially positioned.

filename line col searchkey

Specifies the line (*line*) and column number (*col*) where the cursor should be initially positioned. A **[+SEARCH]** is then executed to find the next occurrence of the search key (*searchkey*). Use *filename 0 0 searchkey* to specify a search from the beginning of the file. On systems that have a terminal description file, the shell variables TERM and TDESC may be used. The shell variable TERM may be used to indicate the terminal type in the terminal description file. The shell variable TDESC may be used to specify the full path name of an alternative terminal description file to be used in place of the default. The alternative terminal description file must have been produced by *idigest*(1).

The -ar option can be used if the TEN/PLUS editor runs in a windowing environment that supports the resizing of windows. When a user resizes the window, the TEN/PLUS editor window will also be resized.

For usage details, see the “TEN/PLUS Reference Manual.”

FILES

<code>/usr/lib/INed/profiles/editorprf</code>	system editor profile
<code>\$HOME/profiles/editorprf</code>	personal editor profile
<code>\$HOME/.estate</code>	records editor state from invocation to invocation
<code>...namxxxxxxxx</code>	temporary "dots" file for editing ASCII file <i>name</i>
<code>name.bak</code>	backup copy of ASCII file <i>name</i>
<code>/usr/bin/e</code> or <code>/usr/bin/te</code>	editor
<code>/usr/lib/INed/termcap/terms.bin</code>	standard terminal description file (if used)

SEE ALSO

ghost(1), history(1) newfile(1), readfile(1), rmhist(1), tdigest(1), versions(1), ined(4),
 "TEN/PLUS Reference Manual."

NAME

fc – compile TEN/PLUS forms

SYNOPSIS

fc [formfile ...]

DESCRIPTION

Fc compiles one or more **.frm** files into the corresponding **.ofm** files. For example, the file **phone.frm** is compiled into **phone.ofm**. A **.frm** file is a structured TEN/PLUS file containing a description of a form to be used to view files containing a certain type of data. In the case of the example above, a form for *phone* data is being compiled. The **.ofm** file contains a digested version of the form file that can easily be read by the TEN/PLUS editor.

Multiple **.frm** files can be compiled by listing them as arguments on the command line.

The same functionality is available from the TEN/PLUS forms helper.

FILES

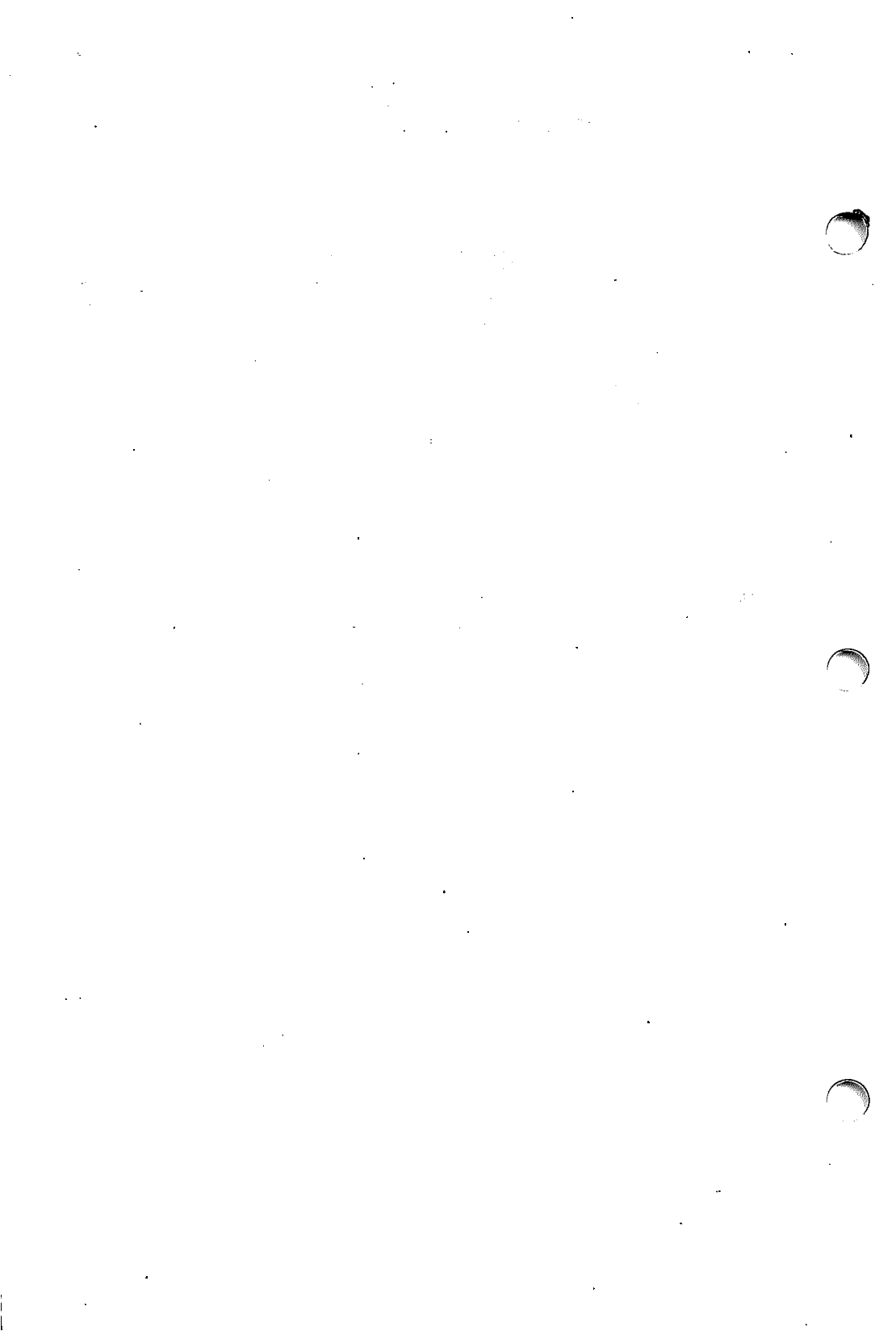
*.frm form source file
*.ofm compiled form file

SEE ALSO

e(1).

BUGS

A fatal error compiling any form causes *fc* to stop. It should go on to the next form file.



NAME

ffill, *fjust* — fill and justify arbitrarily formatted text using INed

SYNOPSIS

ffill [*-ln*]

fjust [*-ln*]

DESCRIPTION

Ffill is a filter that reads text from the standard input file, fills each paragraph, and writes the result on the standard output file. It indents the first and second lines of each paragraph exactly as they are indented in the input file, and all subsequent lines in the paragraph are indented to match the second line. Thus, even block paragraphs with hanging labels on the first line are correctly formatted. Multiple spaces and tabs within a line are converted to single spaces. An extra space is inserted after each word ending with a period, exclamation mark, and question mark.

Ffill is designed for interactive text processing with the **[DO]** function of the INed editor (see *e(1)*), treating a blank line as the end of the current paragraph. With INed it is possible to free format a paragraph beginning the first two lines with the proper indentations and then using *ffill* to fill the entire paragraph. Alternatively, the user may edit already filled paragraphs and then refill the text.

Fjust is also a filter normally invoked with the **[DO]** function of the INed editor. It reads text from the standard input file, fills and justifies each paragraph, and writes the result on the standard output file. *Fjust* is identical to *ffill*, except that it justifies each line to produce an aligned right margin. This is accomplished by replacing spaces within short lines with multiple spaces.

The *-l* flag sets the right margin for *ffill* or *fjust* at column *n*. The default is 65.

SEE ALSO

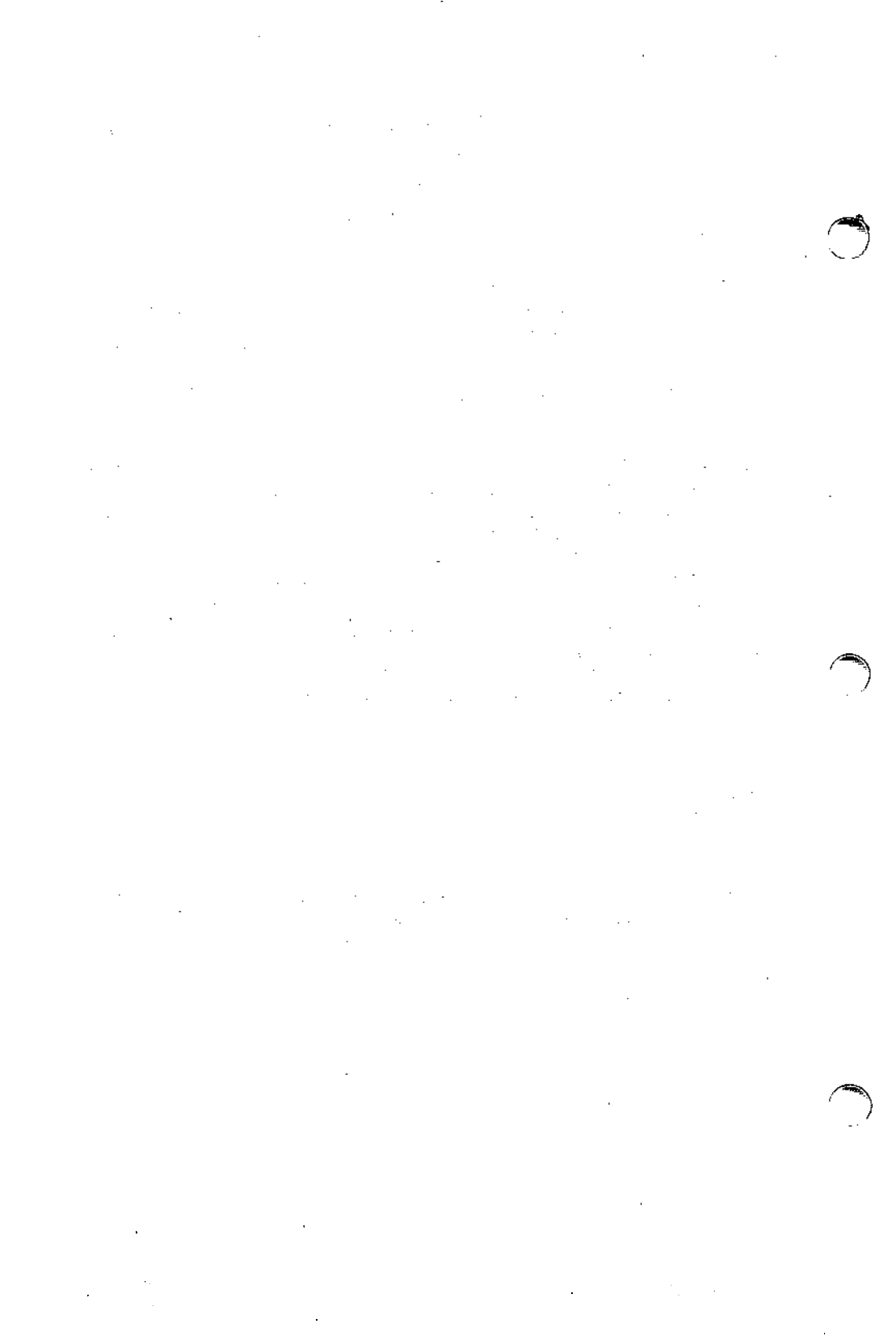
e(1),

“TEN/PLUS Reference Manual.”

BUGS

Ffill and *fjust* set the left margin incorrectly if the first line of any paragraph is more than twice as long as the specified right margin.

Incorrect output is produced for input lines longer than 512 characters.



NAME

filters – filters to be used with the TEN/PLUS editor

SYNOPSIS

box [width]

unbox

indent [count]

undent [count]

space [count]

unspace

tab [filename]

untab [filename]

DESCRIPTION

The **box** filter puts a box of stars around its input. It is particularly useful to C programmers because these boxes are valid as comment sections for the C programming language. The optional **width** argument can be used to specify the width of the box. The default width is 78.

The **unbox** filter removes the box of stars previously created with **box** from around the input.

The **indent** filter indents all input lines by a given amount of spaces. If no amount is specified as an argument to the **indent** command, the indentation will be 8 columns to the right.

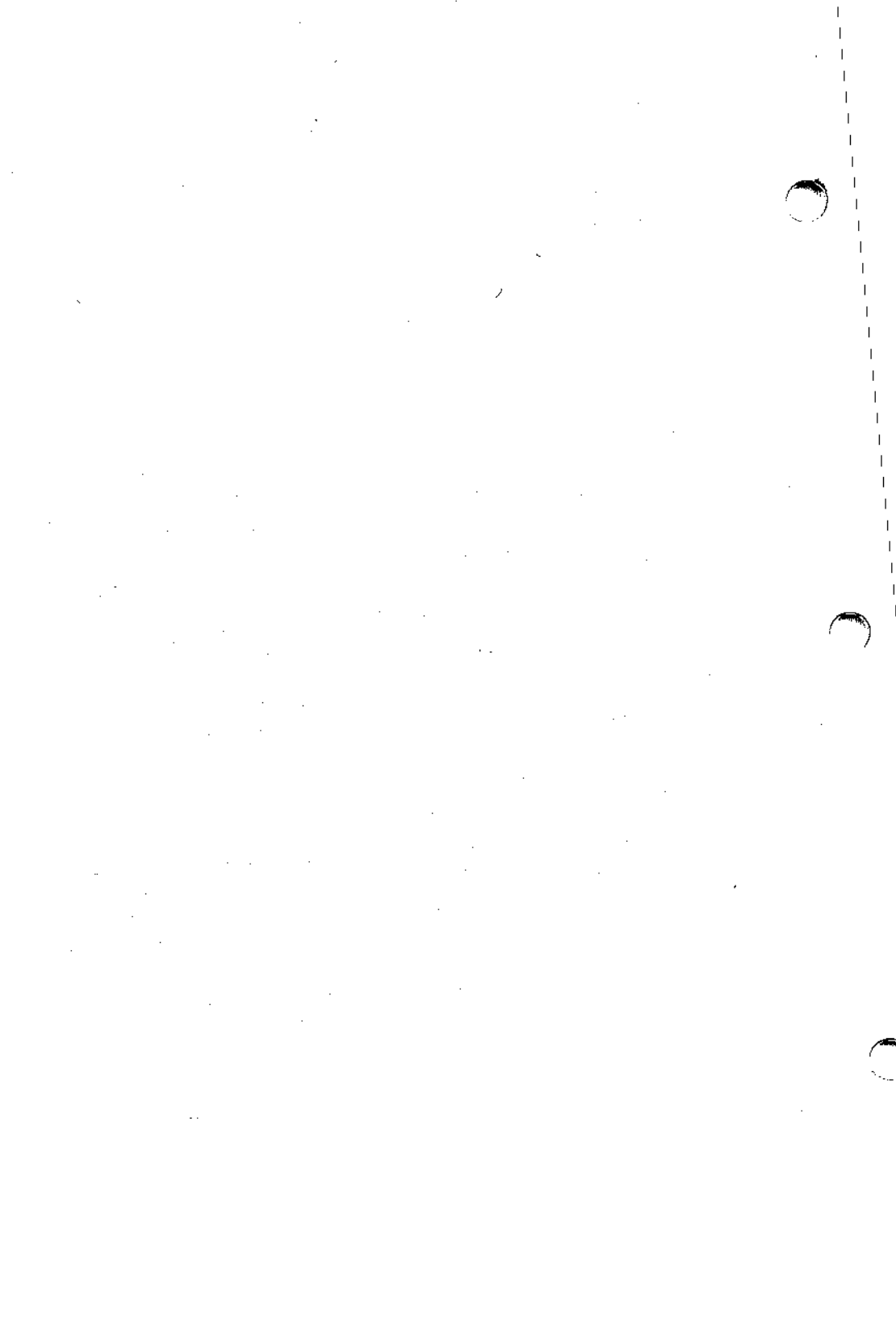
The **undent** filter removes up to a given number of spaces from the input lines. The default is 8 spaces; a different amount can be given as an argument.

The **space** filter adds two blank lines after each line of input. More blank lines may be specified.

The **unspace** filter removes all blank lines from the input.

The **tab** filter reads the input and replaces all strings of two or more blanks with tabs. When argument(s) are specified, they are treated as file names. All strings of two or more blanks are replaced by tabs. This filter is especially useful for TEN/PLUS users who edit with the TEN/PLUS editor, which replaces tabs with spaces, because utilities such as *make* require tabs.

The **untab** filter performs the opposite operation. It replaces all tabs with strings of blanks. Tabs are assumed to occur every 8 spaces.



NAME

ghost – reconstruct previous versions of an INed structured file

SYNOPSIS

ghost [**-d**] [**-p**] *oldname* [*newname* [*m/d[/y]* [*h:m[:s]*]]]

DESCRIPTION

Ghost reads the INed structured file *oldname* and reconstructs a previous version of it in the output file *newname*. If only one argument (*oldname*) is given, the output file (*newname*) is taken to be the same, and the old file is backed up as *oldname.bak*.

If the **-p** option is specified, *ghost* reconstructs the most recent version prior to the time specified. If the **-d** option is specified, a *.bak* file is not created.

The optional [*m/d[/y]* [*h:m[:s]*]] arguments specify a date and time threshold for the reconstruction. The default is the current date and time. If only the month and day are specified, the current year is assumed. If only the date is specified, the time is set to zero (midnight). If only the hour and minute are specified, the seconds are set to zero. Note that the hours are based on a 24-hour clock.

EXAMPLES

ghost *oldfile* *newfile*

Reconstructs the current version of **oldfile** as **newfile**. This is useful if **oldfile** is broken.

ghost *oldfile*

Reconstructs the current version of **oldfile**. Output is to **oldfile**, and the old file is saved as **oldfile.bak**.

ghost -d *oldfile*

Reconstructs the current version of **oldfile**. Output is to **oldfile**, and the old file is not saved.

ghost *oldfile* *newfile* 7/15

Reconstructs the July 15th version of **oldfile** as **newfile**.

ghost *oldfile* *newfile* 7/15/80 15:10

Reconstructs the version of **oldfile** that existed on July 15th, 1980 at 3:10 in the afternoon.

ghost -p *oldfile* *newfile* 7/15/80 15:10

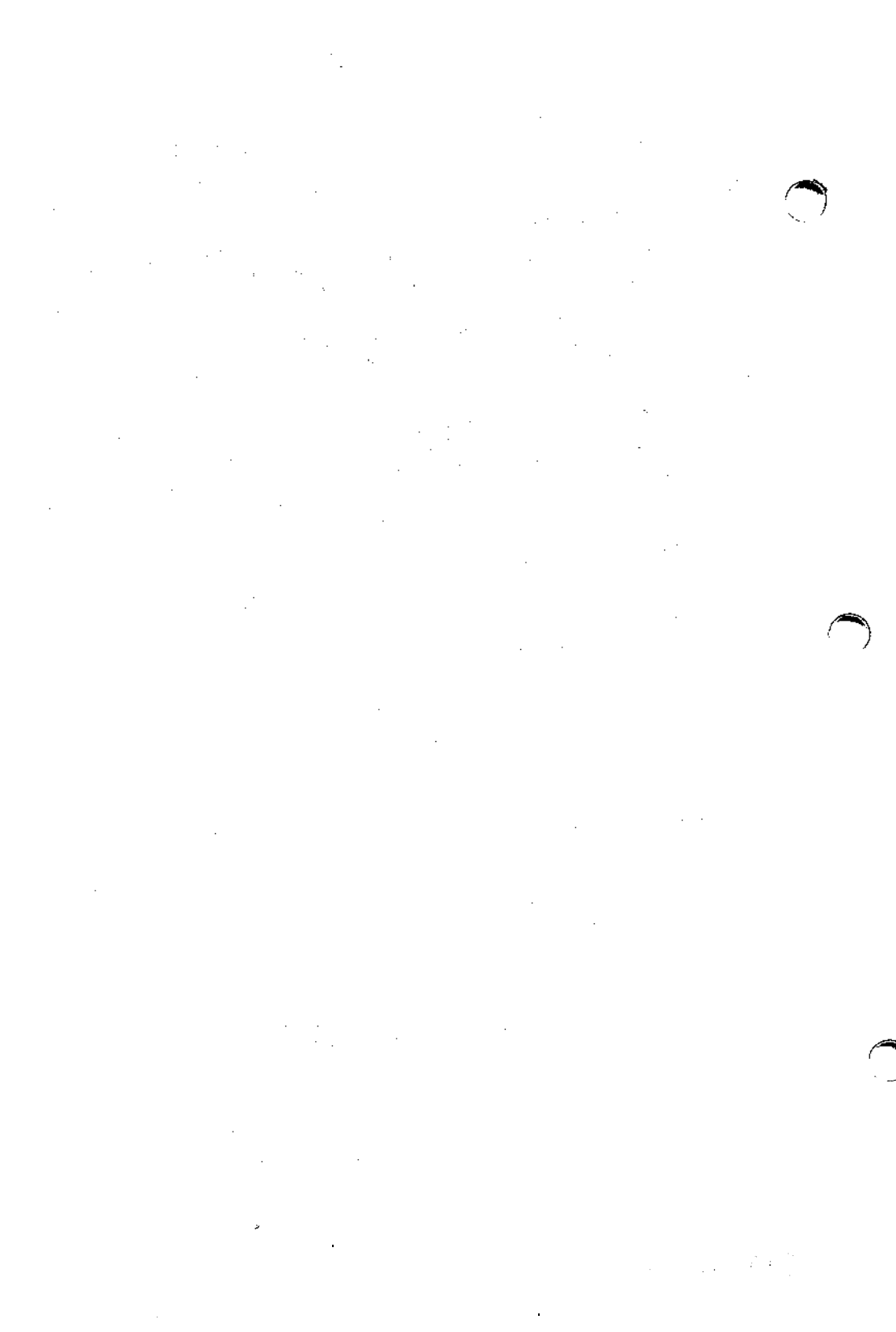
Reconstructs the most recent version of **oldfile** that existed prior to July 15th, 1980 at 3:10 in the afternoon.

ghost *oldfile* *newfile* 7/15/80 15:10:45

Specifies the version of **oldfile** down to the second. This is useful if several changes were made to a file in a very short time.

SEE ALSO

e(1), history(1), newfile(1), readfile(1), rmhist(1), versions(1).



NAME

history – print the history of an INed structured file

SYNOPSIS

history file

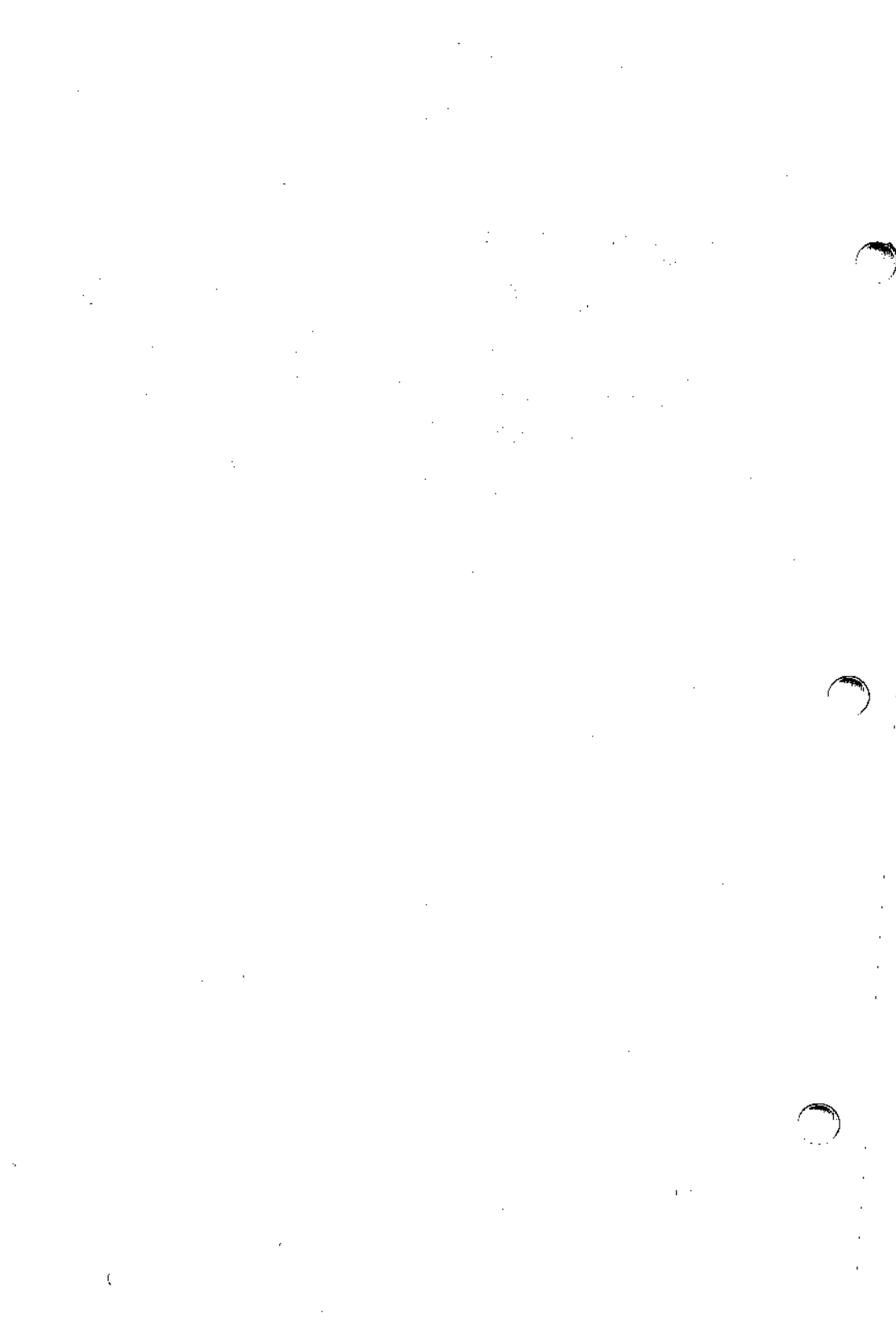
DESCRIPTION

History prints a description of the incremental changes that have been made to *file* since its creation.

The format of a structured file is record-oriented. For a simple text file, these records are the text lines in the file. Along with the records is extra information used for inserting lines, deleting lines, setting the current index, specifying start information, and storing user comments. At the end of the file is information about where the current records are located in the file, so that it can be accessed quickly. *History* uses this same extra information to describe the changes made to the records during each editing session.

SEE ALSO

e(1), ghost(1), newfile(1), readfile(1), rmhist(1), versions(1).



NAME

newfile – convert a text file into an INed structured file

SYNOPSIS

newfile text_file [new_file]

DESCRIPTION

Newfile converts an ASCII text file to a structured file. If only *text_file* is given, the structured file name is taken to be the same, and the text file is backed up by appending a **.bak** to the text file name. *Newfile* does a fatal error exit if the text file does not exist or if it cannot create the structured file. For example:

newfile document sdoc

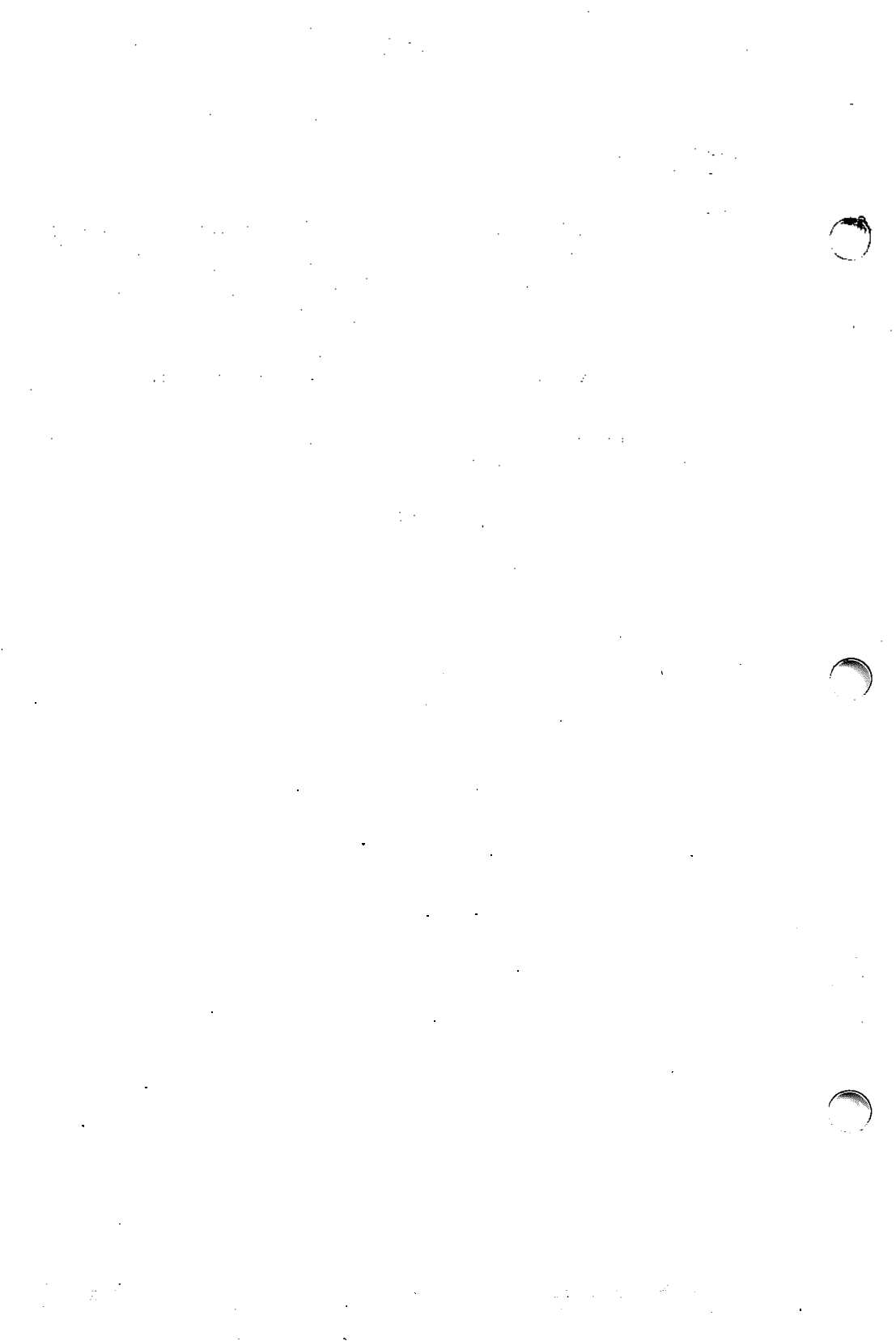
converts the ASCII text file **document** into the structured text file **sdoc**.

newfile document

converts the ASCII file **document** into a structured text file and saves the ASCII version as **document.bak**.

SEE ALSO

e(1), ghost(1), history(1), readfile(1), rmhist(1), versions(1).



NAME

prtsf - print the contents of a structured file

SYNOPSIS

prtsf [-f [formatfile]] [-c [criteria]] [-e [string]] [-r#] files

DESCRIPTION

The *prtsf* utility is a simple report generator that prints information from a structured data file (as created by the TEN/PLUS editor) on to the standard output. Users can select the records from the file from which they want to print information, and they can also choose how the document should look.

The *criteria* option selects the records to print; if no *criteria* are given, all records are selected. If the *-c* option is given alone, *prtsf* reads the standard input for the *criteria*. The *formatfile* specifies the contents and format of the output; *prtsf* formats each selected record according to this specification and outputs the result. If the *-f* option is given alone, *prtsf* reads the standard input for the format. If no *formatfile* is given, *prtsf* outputs the contents of the character nodes in the file, one per line. If the *-e* option is used, a form-feed is output between each selected record's output. If a string is given with the *-e* option, it is output along with a newline instead of the form-feed. If the *-r* option is used, only the record number specified by *n* is selected.

The *formatfile* is made up of invariant text (which is output unchanged) and field specifiers. The format of the field specifiers is a field name surrounded by either square brackets ([and]) or curly brackets ({ and }). When the *prtsf* program sees a field specifier, it looks to see if the current record contains a field of the given name and substitutes the field value for the field specifier in the output. If the record does not have the specified field, a null string is used for the substitution. If square brackets are used to delimit the field specifier, the text is padded with spaces or truncated to the same length as the field specifier (including the brackets). If curly brackets are used, the entire field value is substituted. For example:

name: [name], age: {age}, title: [title]

has two fixed-width fields (name, with a width of 6, and title, with a width of 11) and one variable-width field (age).

It is possible to use relative TEN/PLUS path names instead of simple field names. This can be used to display multiple lines from a field. For example:

```
[message/0      ]
[message/1      ]
[message/2      ]
[message/3      ]
[message/4      ]
```

would put the first five lines of the "message" field into the five fields in the form (or blanks of that width if there are fewer than five lines). If a simple field name is used (such as "message," without the line specifier), and the field has multiple lines, only the first line will be substituted. These path names can be arbitrarily complex and can include index numbers (which start at zero), names, and asterisks (*)

in any part of the path. This means that the *prtsf* program can be used on any type of structured data, provided that all records in the data file have similar children. For example, the entire message field could be printed by:

```
[message/*          ]
```

In order to deal with long path names, the format files have an alias mechanism. An alias line is one that begins with a pound sign (#) in column one, an "a" in column two, one or more spaces followed by an alias, and followed by one or more spaces and a path name. The following example would print all lines of the message field.

```
#a mess message/*
```

```
[mess              ]
```

All lines with a pound sign (#) in column 1 immediately followed by a blank are ignored; thus comments may be put into a format.

```
# this is a comment
```

There is also a mechanism for printing header and footer information once per file. Lines beginning with "#h " are header lines and are printed beginning with the character after the blank before any records are processed. Lines beginning with "#f " are footer lines and are printed after all records have been processed.

```
#h File: {=FNAME}.
```

```
#f End of messages for file {=FNAME}.
```

In addition to the headers that can be printed once per file, there are grand headers that can be printed once per invocation. Lines beginning with "#gh " are grand header lines and are printed starting with the character after the blank before any files are processed. Lines beginning with "#gf " are grand footer lines and are printed after all files have been processed.

```
#gh Start of Mail Messages.
```

```
#gf End of all messages.
```

There are eleven special field names. The first seven take no arguments.

```
=FNAME is the name of the file being processed.
```

```
=NRECS is the number of records in that file.
```

```
=RECPRT is the number of records printed for the current file.
```

```
=GRECPRT is the number of records printed for all files.
```

```
=RNUM is the number of the record being processed.
```

```
=DATE is the current date.
```

```
=TIME is the current time.
```

These take two arguments: the *fieldname* to keep totals on and the *format* to print it in. The format is a simplified **printf**-style string *m.n*, where *m* is the minimum width of the field and *n* is the number of digits to the right of the decimal point.

```
=TOTAL is the total for a field per record.
```

```
=FTOTAL is the total for a field per file.
```

```
=GTOTAL is the grand total for a field for all files.
```

For example, to use =TOTAL *fieldname format*, you would say:

```
=TOTAL price 5.2
```

=RTOTAL, the total of a row per line, is a special field that takes more than one *fieldname*. Its use is =RTOTAL *format fieldname [field-names]*.

In the following example, this *formatfile* would print a header consisting of the name of the mailbox and the number of messages it contains, followed by the messages each with its record number in the file.

```
#h Mailbox {=FNAME} contains {=NRECS} messages.
Message {=RNUM}
From: {From}
Message: {Message/*}
```

The *criteria* file contains expressions in the following form:

```
criterion:: pattern is in PATH |
            pattern is not in PATH |
            PATH > number |
            PATH < number |
            PATH = number |
            PATH >= number |
            PATH <= number;
```

```
criteria:: criterion |
            ( criteria ) |
            criteria and criterion |
            criteria or criterion;
```

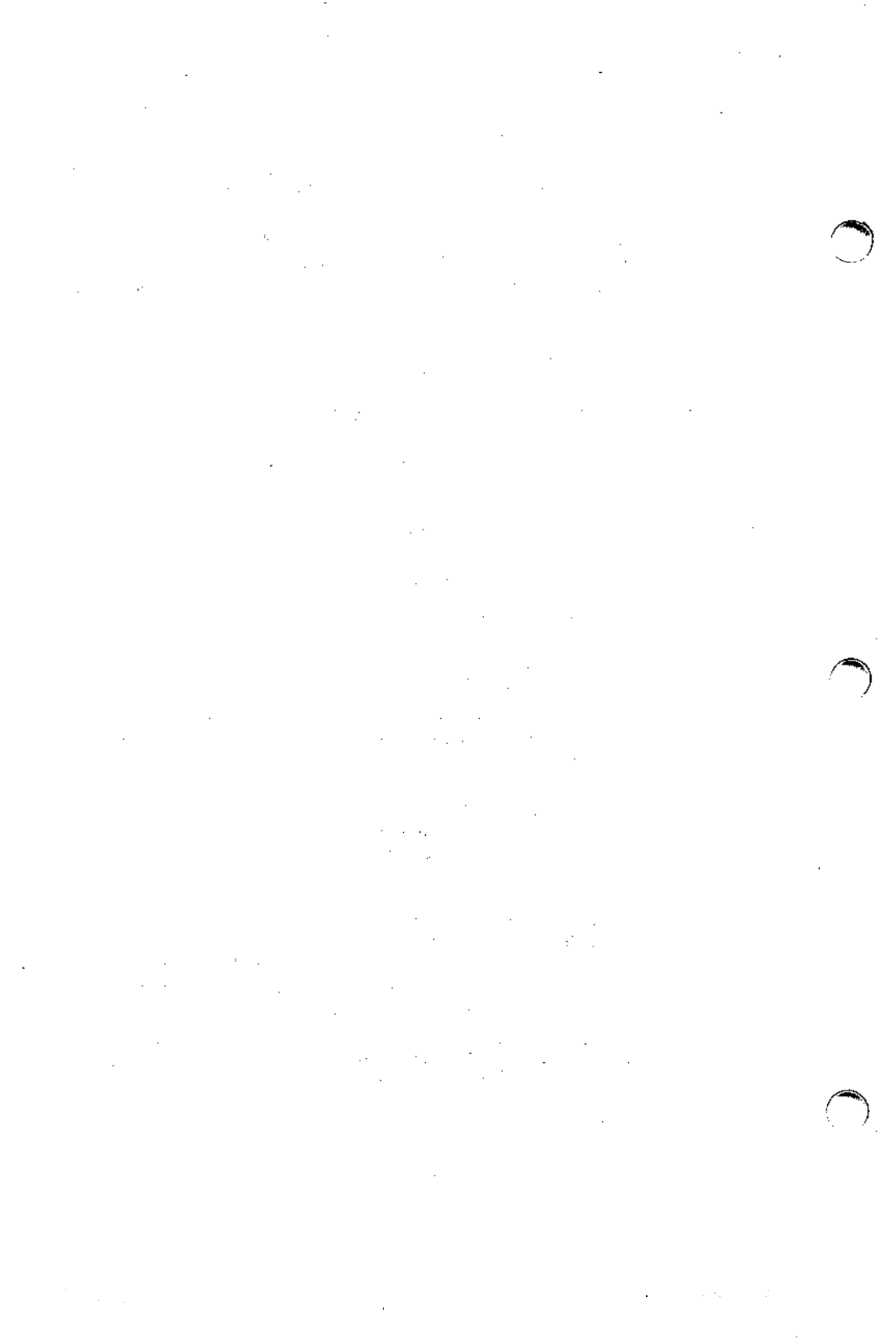
where *pattern* is a string that can be compiled by *regex(3)*, for example, **Frank, 2/* /84**, or **\$(6-9)[0-9]{4}**, *PATH* is the TEN/PLUS path to the node to be checked for the value, and *number* is a float, for example, **35** or **98.6**.

A sample *criteria* file might contain:

```
(California is in Address/* /State or
 Nevada is in Address/* /State ) and
(Peter is in Name or
 Paul is in Name or
 Mary is in Name) and
(Earnings > $40,000)
```

The above example would print all records that have Peter, Paul, or Mary in the *Name* field, California or Nevada in the *State* field, and have an amount in the *Earnings* field greater than \$40,000.

The *format* file may contain the *criteria* information on special control lines. Lines beginning with "#c" designate a *criteria* line. If the -c option is used on the command line, it overrides the criteria in the *specfile*.



NAME

prtty - print to printer port of terminal

SYNOPSIS

prtty [-l [number]] [file] ...

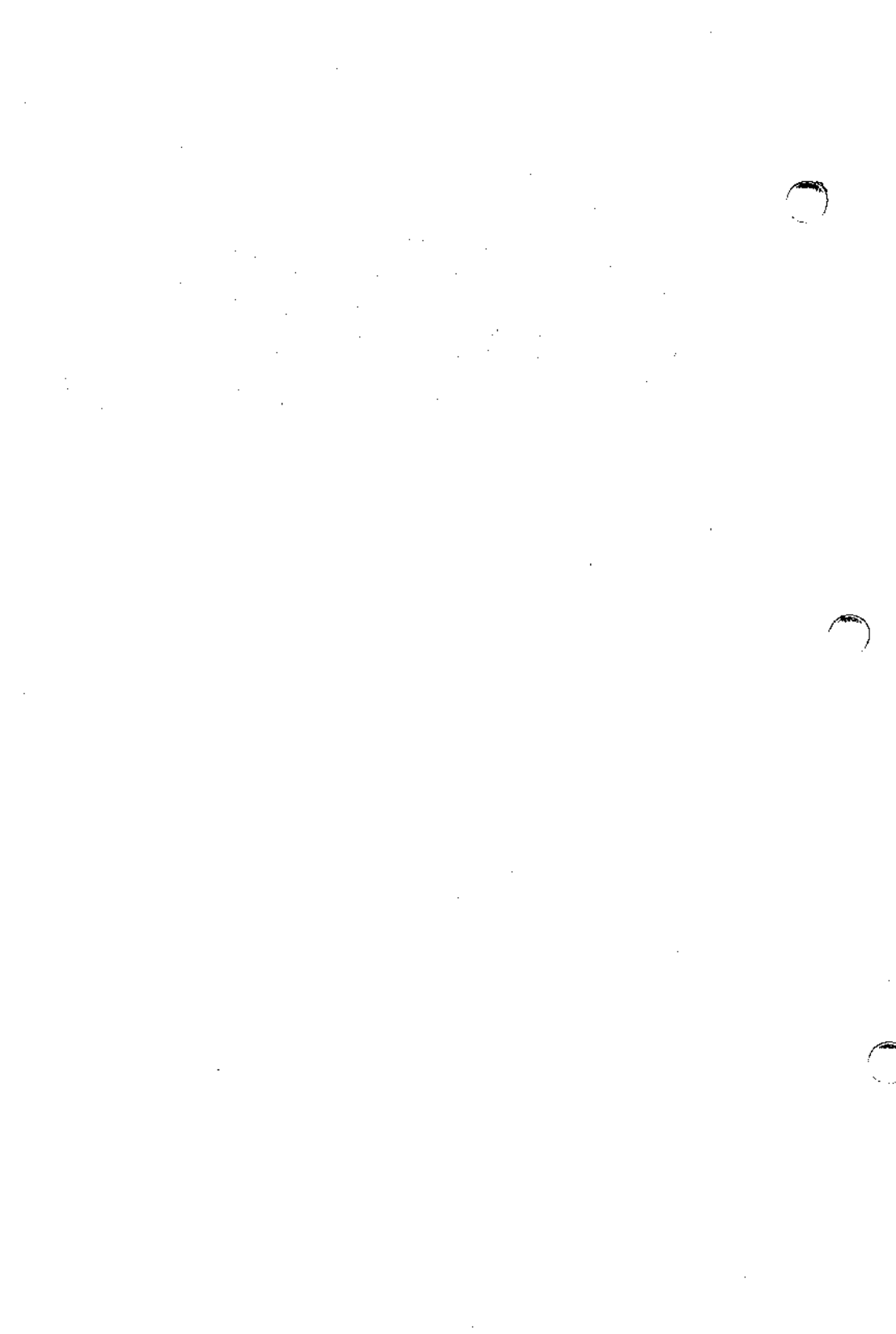
DESCRIPTION

Prtty prints the named files to the printer port of your terminal. If the **-l** flag is specified, it will prompt you to start the printing and will reprompt after *number* lines. The default number of lines is 60. If no files are specified, it reads the standard input.

Prtty reads the environment variable **\$TERM**, and looks for that terminal in **/usr/lib/INed/termcap/terms.bin**, the TEN/PLUS terminal description file, in order to get the sequences k2 (enable printer port) and k3 (disable printer port).

SEE ALSO

terms (4).



NAME

readfile – display structured files

SYNOPSIS

```
readfile [ -dghstu? ] [ +n ] [ -n ] [ files ] [ -o outfile ]  
[ files ]
```

DESCRIPTION

Readfile takes a list of structured files and produces an ASCII printout of the contents of each file that illustrates their tree structure. *Readfile* behaves like *cat*(1) on simple ASCII files. On structured text files, it prints out only the file's string data without indentation.

When no arguments are specified, *readfile* reads standard input as an ASCII file.

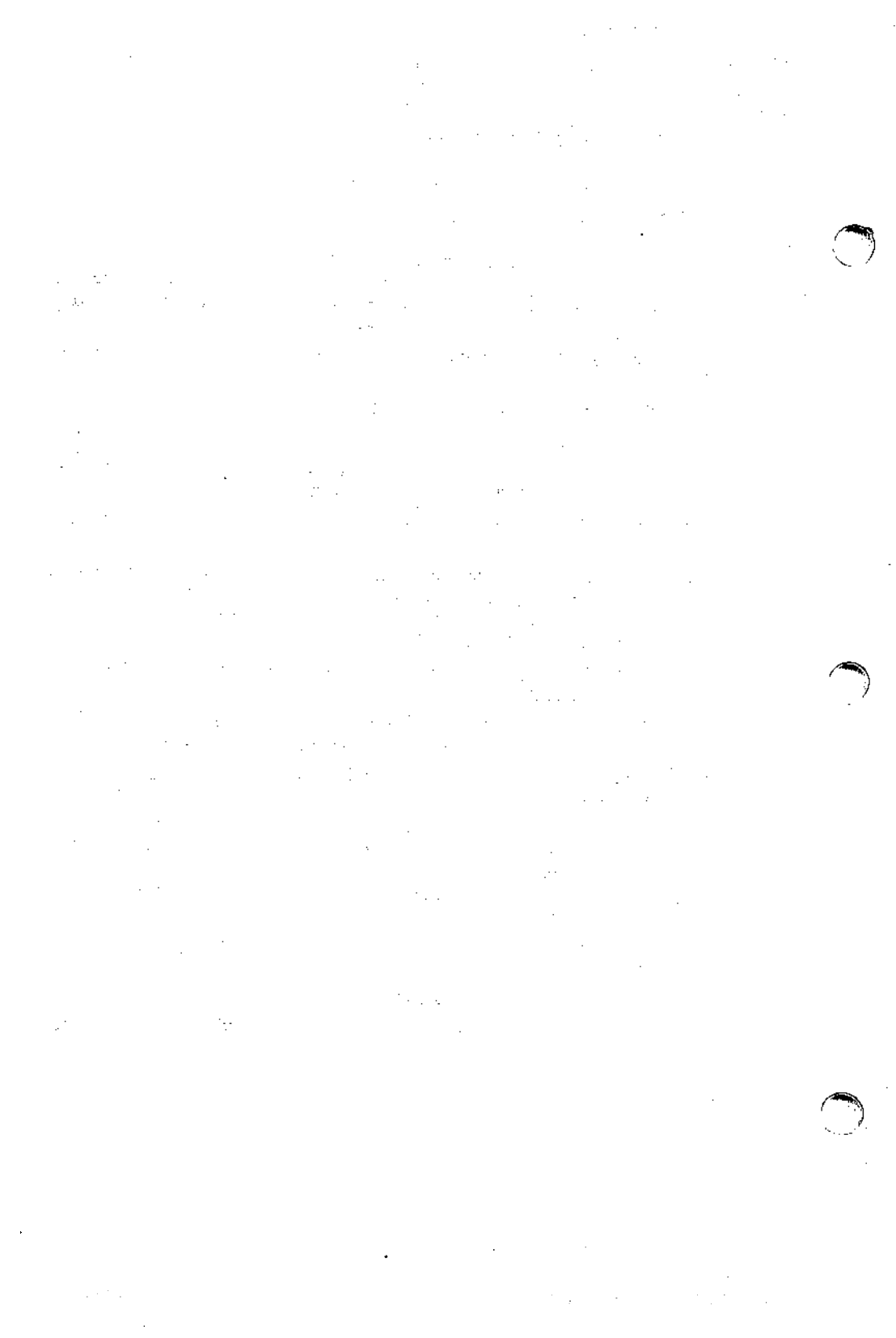
The following options are recognized:

- d Detailed formatted dump of the contents of the input files.
- g Convert the control characters that are used as graphics characters in forms to printing characters that appear similar.
- h Separate the output into sections, with headers that identify the input file for each section.
- o Indicate that the next argument is to be the name of an output file to be used instead of the standard output. This file can be the same as one of the input files, in which case *readfile* backs up the contents of the input file in a *.bak* file.
- s Silence messages that report unstructured and nonexistent files among arguments.
- t Illustrate the structure of the file with a tree diagram. This option is ignored on structured text files and ASCII files.
- u Suppress buffering of output (default buffer size is the size of the disk block).
- n Set to *n* columns the size of the increments and decrements in the indentation when printing a non-text structured file. The default value is five columns. This option is meaningless when option -t is specified. The option must be separated from adjacent ones by spaces.
- +n Begin reading file at the *n*th record. This option must be separated from adjacent ones by spaces.
- ? Prompts for syntax of usage.

Options can be combined and can appear in any order, with the exceptions mentioned above.

SEE ALSO

cat(1), *newfile*(1).



NAME

rmhist – remove history from INed structured files

SYNOPSIS

rmhist [**-d**] [**-f**] [**-kn**] file ...

DESCRIPTION

Rmhist takes a list of file names and removes the history information from each file. The files are backed up so that the old version of the file is available as *file.bak*. If any of the files are not structured files, a warning will be printed and no action will be taken.

Following are the options to *rmhist*:

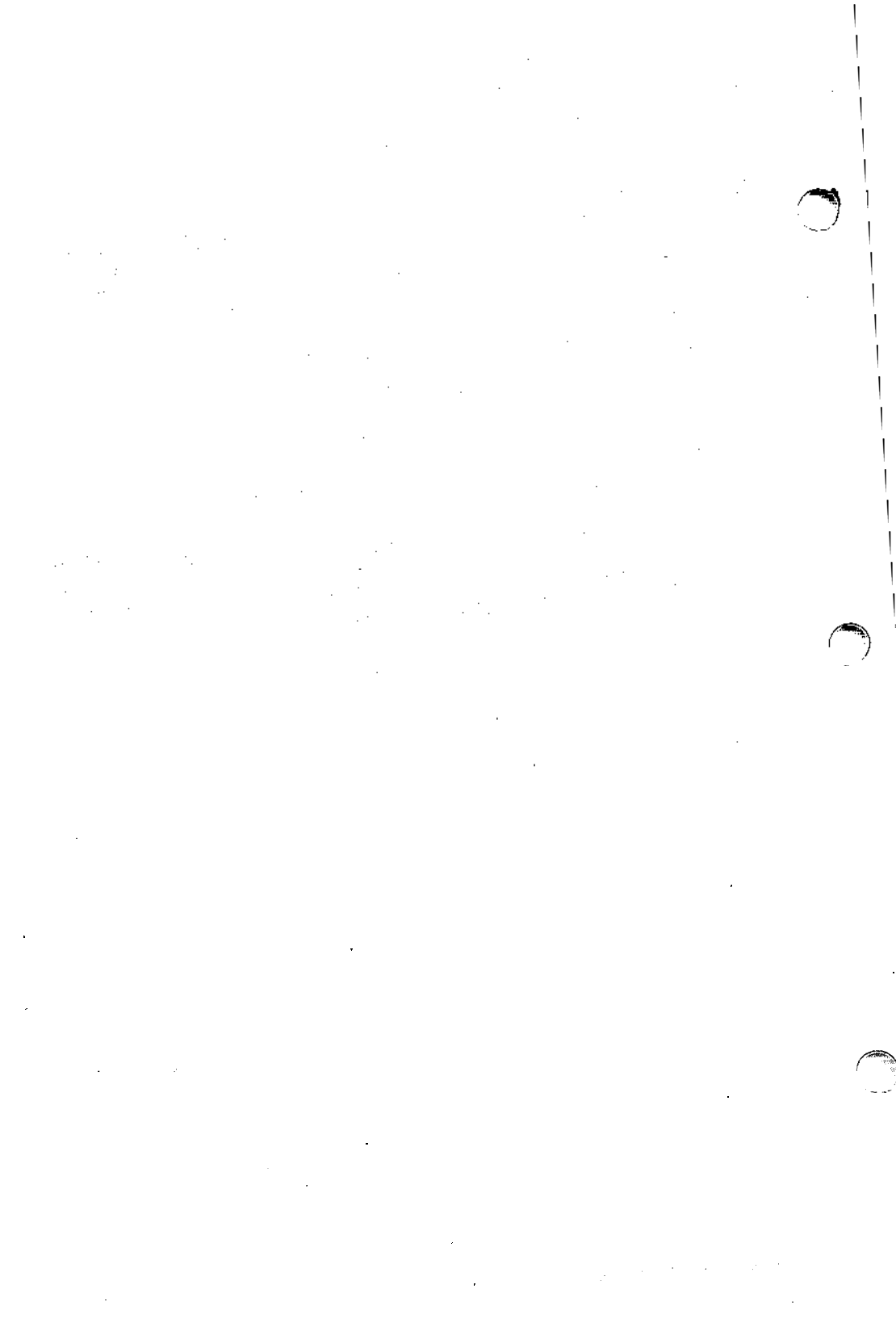
- d** Delete **.bak** file after removing history.
- f** Do not print warnings about attempts to remove history from file(s).
- kn** Keep only the last *n* days of history from the current date.

SEE ALSO

e(1), ghost(1), history(1), newfile(1), readfile(1), versions(1).

WARNING

If the system date has been improperly set to a future date, history time stamp information will be invalid, and *rmhist* will fail. Correct the problem by using *ghost*(1) with the future date as an argument. Then use *rmhist* to remove the history.



NAME

rpl — replace all occurrences of a string in a file being edited by INed

SYNOPSIS

rpl expr substitute

DESCRIPTION

Rpl reads the standard input and, on the standard output, writes the file obtained by replacing all occurrences of the character string that is its first argument by the string that is its second argument.

Rpl is intended primarily for use with the **[DO]** key of the INed editor, but it can also be used for general text processing applications.

The first argument is usually a simple text string. However, certain special characters may be used to form *regular expressions*, which can match classes of text strings. The symbols that have special meanings in forming regular expressions are listed below.

symbol *meaning*

- ^ At the beginning of a regular expression, matches the empty string at the beginning of a line.
- .
- Matches any character except a new-line character.
- \$ Matches the end of a line.
- *n* Matches the character whose ASCII code is given by *n*, where *n* is up to three octal digits, except for 012 (because *rpl* works on a single line at a time, it is impossible to match the new-line character).
- [] A string of characters enclosed in square brackets matches any character in the string but no others. If, however, the first character of the string is ^, the regular expression matches any character except new-line and the characters in the string.
- Within brackets the minus means “through.” For example, [a-z] is equivalent to [abcd..xyz]. The - can appear as itself only if used as the last or first character. For example, the character class expression []-] matches the characters] and -.
- + A regular expression followed by + means “one or more times.” For example, [0-9]+ is equivalent to [0-9][0-9]*.
- *
- A regular expression followed by * means “zero or more times.”
- {*m*}
- {*m*,}
- {*m*,*u*}
- Integer values enclosed in { } indicate the number of times the preceding regular expression is to be applied. The minimum number is *m*, and *u* is a number, less than 256, which is the maximum. If only *m* is present (i.e., {*m*}), *m* indicates the exact number of times the regular expression is to be applied. {*m*,} is analogous to {*m*,infinity}. The plus (+) and star (*) operations are equivalent to {1,} and {0,}, respectively.

(...) Parentheses are used for grouping. An operator (e.g., *, +, {}) can work on a single character or a regular expression enclosed in parentheses. For example, (a*(cb+)*\$).

Of necessity, all the above defined symbols are special. Therefore, if they are to be used as themselves, the ^, ., \$, [, +, *, {, \, (, and) must be escaped by preceding each of them with a \. Also, to avoid conflicts with shell special characters, it is a good idea to enclose the *expr* in single quotes when it is other than a simple text string.

Example:

```
rpl '([A-Za-z][A-Za-z0-9]{0,5})' xxx
```

This example will replace a string of up to six alphanumeric characters (the first of which must be alphabetic) with the string *xxx*.

To delete a string using *rpl*, replace it by the null string "".

SEE ALSO

e(1), regex(3x),
"TEN/PLUS Reference Manual."

DIAGNOSTICS

If lines are longer than 256 characters, if two arguments are not given, or if the first argument is the null string, *rpl* exits with status -2.

NAME

sortsf — sort a structured data file

SYNOPSIS

sortsf [**-bdfinrv**] +field [+field] file

DESCRIPTION

The *sortsf* utility sorts records in a TEN/PLUS structured file according to the specified field. Field is a TEN/PLUS path name. If the field has multiple lines, the first line is used. When there are multiple fields, later fields are compared only after all earlier fields are determined to be equal. Lines that otherwise compare equally are ordered with all bytes significant. The default ordering is lexicographic by bytes in machine collating sequence. The ordering is affected by the following options, one or more of which may appear:

- b** Ignore leading blanks (spaces and tabs) in field comparisons.
- d** “Dictionary” order: only letters, digits, and blanks are significant in comparisons.
- f** Fold uppercase letters into lowercase letters.
- i** Ignore characters outside the ASCII range 040-0176 in non-numeric comparisons.
- n** An initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, is sorted by arithmetic value. Option **-n** implies option **-b**.
- r** Reverse the sense of comparisons.

The **-v** option causes *sortsf* to be verbose about its operations.

EXAMPLES

Sort a mailbox in reverse dictionary order by sender and subject.
sortsf -rd +Sender +Subject mbx

FILES

/tmp/???

SEE ALSO

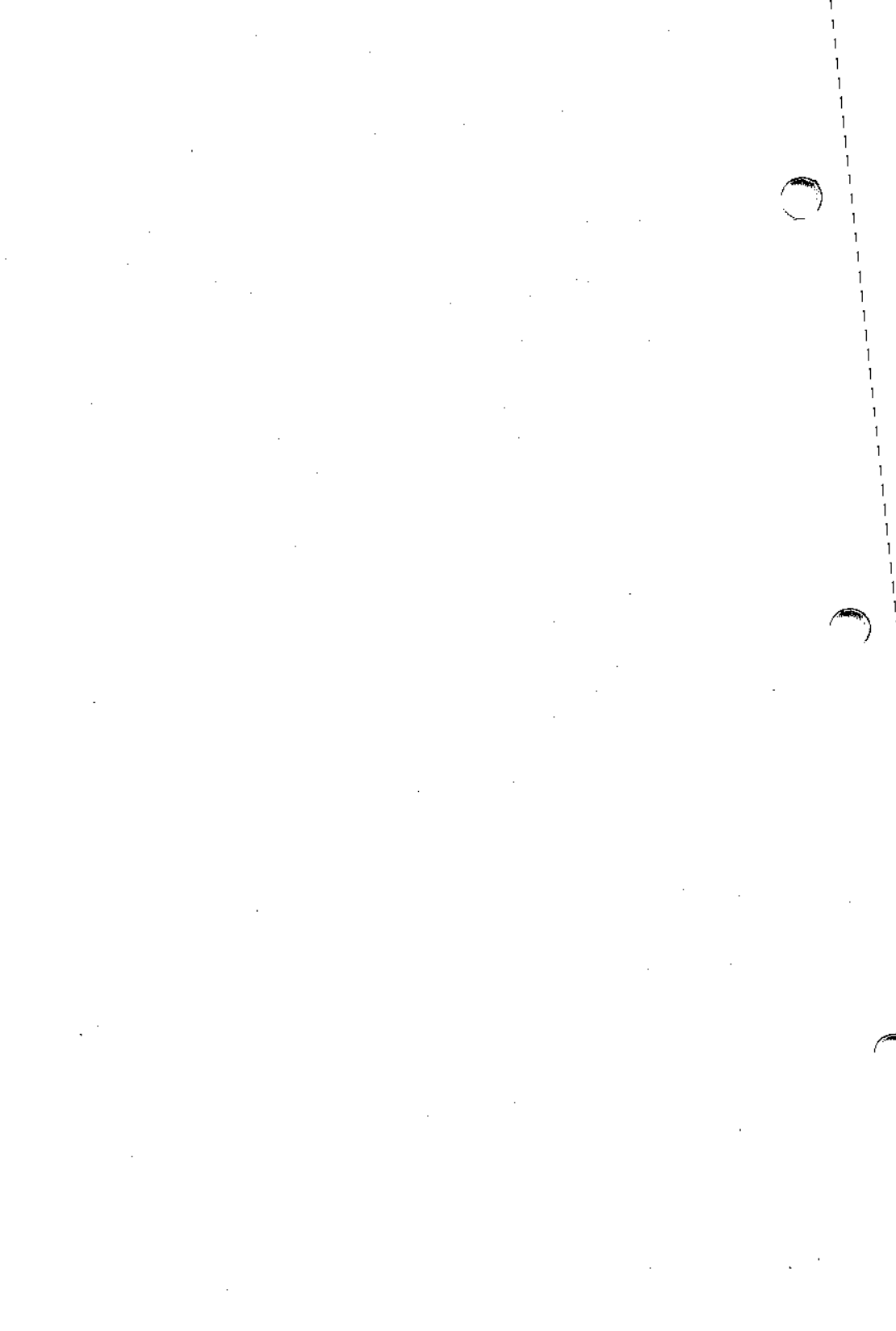
catsf, prtsh.

DIAGNOSTICS

Comments and exits with nonzero status for various trouble conditions.

WARNINGS

The *sortsf* program preserves history, so large files tend to grow quickly when being sorted.



NAME

tconvert — convert /etc/termcap to a structured file

SYNOPSIS

tconvert *inputfile* *outputfile*

DESCRIPTION

Tconvert reads record 0 from the file *def.trm* to get the default input mappings and graphics output mappings (*g0-g9* and *x0-x1*), then copies all records from *inputfile* to *outputfile*. Next, *tconvert* scans /etc/termcap, or the file that the environment variable **TERMCAP** (if set) points to, for terminal descriptions of terminals not already listed in the *inputfile*, and builds a record array of the *termcap* data for each new terminal to be added while merging in the default input and output mappings. Finally, *tconvert* appends the new record(s) to *outputfile*.

The resulting merged terminal descriptions are only rough drafts, generally sufficient for use by the INed *termcap* editor (*e(1)*), but likely in need of improvement to make use of any special function keys or capabilities that the newly-added terminal(s) may offer. Such improvements must be made to the new records in *outputfile* prior to using it as input to *tdigest(1)*. To facilitate changes in *outputfile* via the editor, the name of the *outputfile* must be *trm* or contain the suffix *.trm*, otherwise the proper forms needed for editing of the file will not be called by the editor.

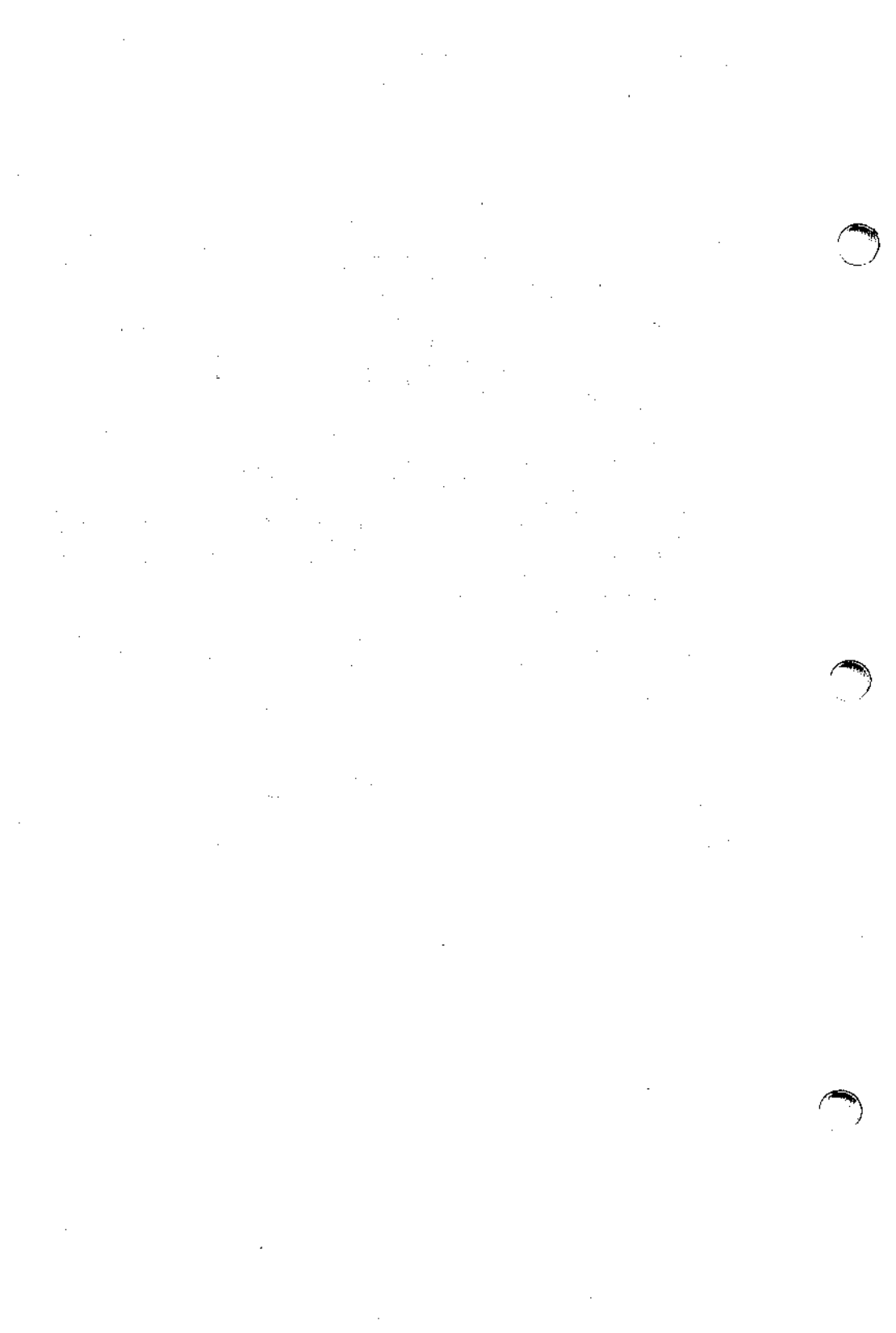
The terminal names *tconvert* uses are the ones in the second field of the *termcap* entries. Records from the *inputfile* override *termcap* entries.

FILES

/usr/lib/INed/termcap/def.trm	default terminal descriptions for INed termcap editor (a structured file)
/etc/termcap	terminal capability database

SEE ALSO

e(1), *tdigest(1)*.



NAME

tdigest – digest the terms files

SYNOPSIS

tdigest inputfile outputfile

DESCRIPTION

Tdigest converts the terminal description entries in the structured *inputfile* into binary entries in *outputfile* as required for efficient usage by the INed *termcap* editor.

Upon startup the editor checks the TDESC environment variable to determine the pathname of the binary description file that should be read. If the TDESC variable is not set the editor checks to see if the SYS variable is set. If SYS is set the editor reads **\$\$SYS/termcap/terms.bin**. If SYS is not set then the editor looks in the default system-wide description file, **/usr/lib/INed/termcap/terms.bin**. This mechanism allows for the testing of a new binary terminal description file prior to making it available to the general user community.

FILES

/usr/lib/INed/termcap/def.trm	terminal descriptions (a structured file)
/usr/lib/INed/termcap/terms.bin	the digested terminal descriptions file (a binary file)

SEE ALSO

e(1), tconvert(1).



NAME

versions — print out modification dates in an INed structured file

SYNOPSIS

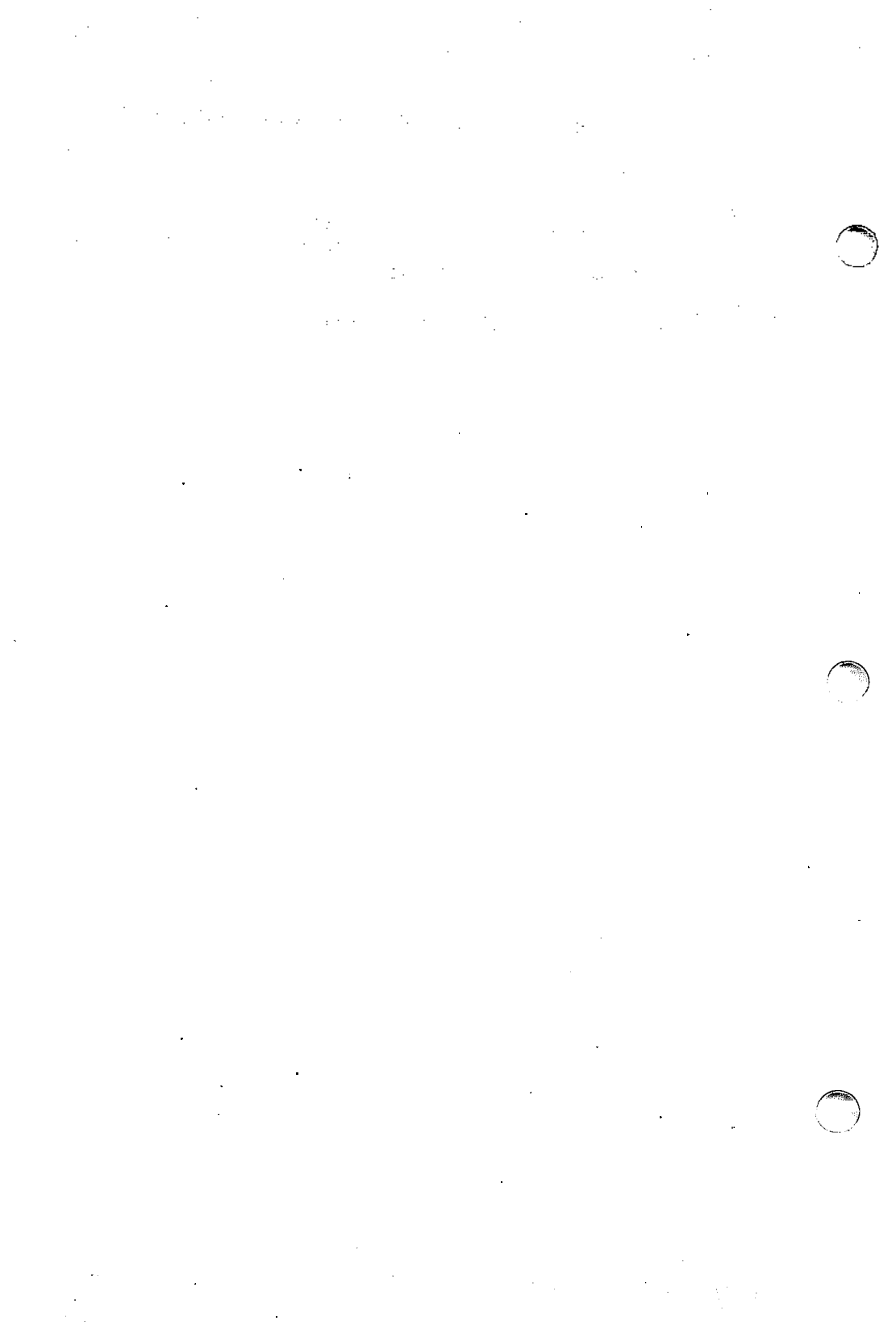
versions file

DESCRIPTION

Versions prints out the modification dates from a structured file. These are the times that the file was opened for modification and can be used to reconstruct the file using the *ghost*(1) program.

SEE ALSO

e(1), ghost(1), history(1), newfile(1), readfile(1), rmhist(1).



NAME

ined – files used by the INed system

DESCRIPTION

The directory `/usr/lib/INed` contains a number of files and subdirectories used internally by the INed editing system.

forms is a directory containing forms used by the INed system. Files ending in `.x` or named `x` use the form `x.ofm`. The forms are binary files used directly by INed in generating displays.

helpers is a directory containing programs invoked by INed to help work on certain kinds of data. Files ending in `.x` or named `x` use the helper named `x.help`. Helpers typically supply the functions listed by the INed `LOCAL-MENU`.

help is a directory containing files to display when the INed `H LP` function is used.

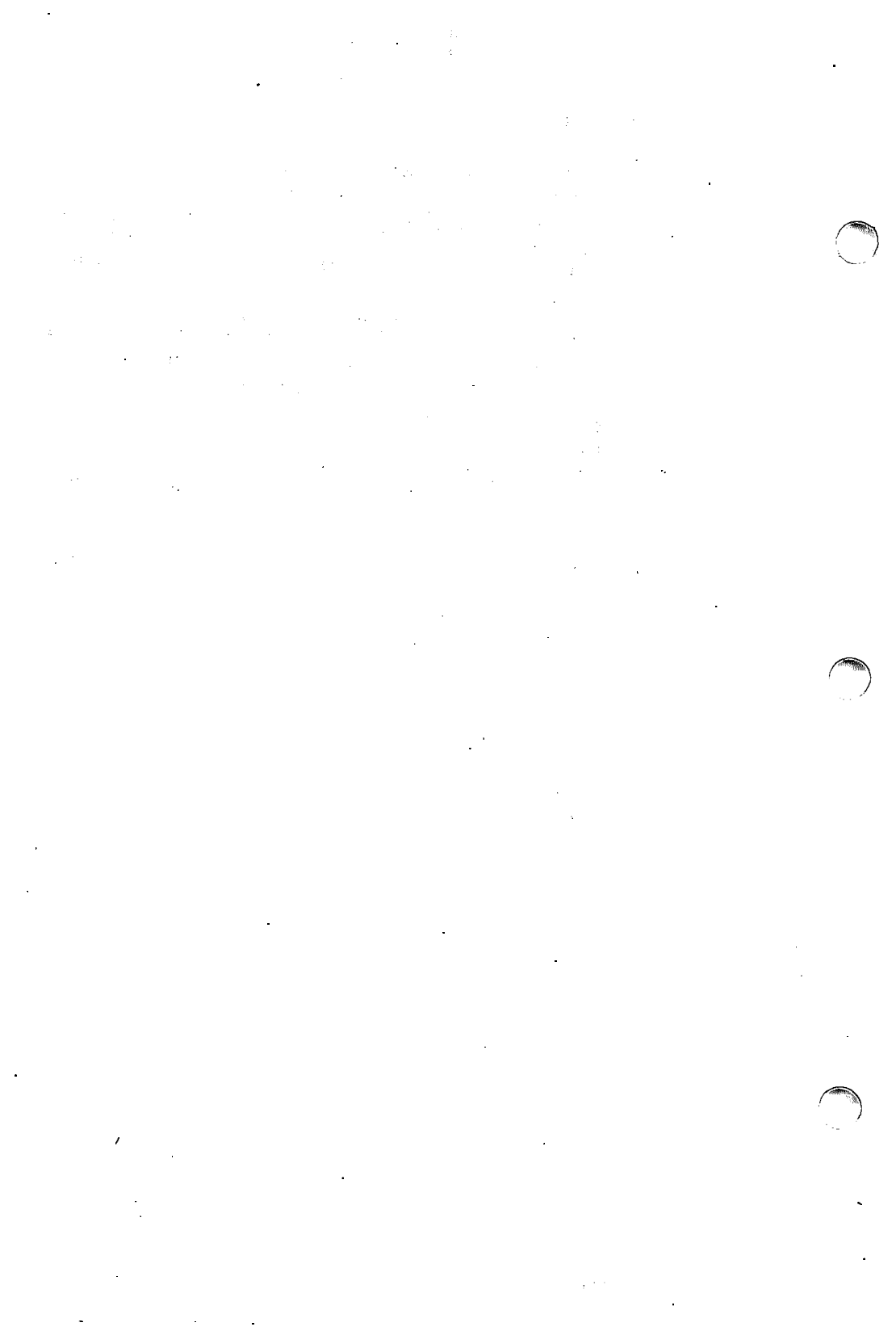
hmgs is a directory containing all the messages displayed by the INed `HELP` function when an error message or menu is displayed.

FILES

`/usr/lib/INed`

SEE ALSO

`e(1)`,
“TEN/PLUS Reference Manual.”



NAME

def.trm, terms.bin – INed terminal description file

DESCRIPTION

The structured file **def.trm** contains an entry for each terminal supported by INed. Each terminal description contains two sections: a mapping of editor functions to the keyboard (**Input Sequences**), and a list of the terminal escape sequences used to update the screen (**Output Sequences**).

A default version of **def.trm**, with various terminal descriptions, is installed in the directory **/usr/lib/INed/termcap**. The definitions in this file, which can be modified, may be available to all users. Alternately, individual users can modify private copies of **def.trm**, to customize their terminal capabilities to their needs.

Def.trm consists of a list of records, each entry of which is one terminal description. Record 0 of **def.trm** contains a set of default input mappings and graphics output mappings that are used by *tconvert*(1) to create new INed terminal descriptions from a *termcap*() database. The fields in each record are **Terminal Type**, **Output Sequences**, and **Input Sequences**. To view the output sequences for a specific terminal type, **ZOOM-IN** on the **Output Sequences** for that type. To view the input sequences for a specific terminal type, **ZOOM-IN** on the **Input Sequences** for that type.

Input sequences map INed functions to specific keys or combinations of keys. The fields included in each entry are **Editor Function**, **Value**, and **Key Name**. The **Editor Function** field lists each INed function by name. The **Value** field lists the code produced by the terminal (obtained from the terminal's technical manual) that initiates the named function. The **Key Name** field contains a mnemonic identifier for the keys that invoke the function listed as the **Editor Function**. The following are sample input sequences from the VT100 terminal description:

<i>Editor Function</i>	<i>Value</i>	<i>Key Name</i>
DELETE-CHARACTER	\0177	DELETE
HELP	^_	CTRL-?
MENU	^[OP	PF 1

The escape sequences defined as output sequences map terminal capabilities, such as clear screen and cursor motion, to the specific terminal commands that initiate those capabilities. In addition, other capabilities that are generally thought of as input-related, such as the definitions of the **BREAK** and **QUIT** characters, are described as output sequences because input sequences are used to define only INed function mappings. Because the input sequences contain no provisions for boolean values, such as whether **ENTER** and **EXECUTE** are mapped to the same key, these are also defined as output sequences. The output sequences include terminal-dependent output string sequences, modifiers such as boolean flags (e.g., whether the terminal supports automatic margin-wrapping) and numbers (e.g., the number of lines and columns on the screen), and input-related escape sequences.

Each entry in the **Output Sequences** includes three fields: **Name**, **Meaning**, and **Value**. The **Name** field contains the two-character

mnemonic for the terminal capability, as defined by the INEd editor or in *termcap*(). The **Meaning** field contains the data type (string, numeric, or boolean) of the capability, and a short description of it. The **Value** field contains the terminal code that produces the desired action (as defined in the technical manual for the specific terminal), or TRUE/FALSE if the capability has a boolean value. The following are sample output sequences from the VT100 terminal description:

<i>Name</i>	<i>Meaning</i>	<i>Value</i>
al	Str: insert line	\E[L
bs	Bool: terminal can backspace with ^h	TRUE
ee	Bool: enter/execute	TRUE
k0	Str: DEL character	^]
k1	Str: QUIT character	

The first entry indicates the terminal code that causes a new line to be inserted; the second entry indicates that the terminal can backspace using CTRL-h; the third entry indicates that enter and execute are mapped to the same key; the fourth entry indicates that the DEL character is CTRL-]; and the fifth entry indicates that the QUIT character is the default as determined by the environment.

The **Value** field is interpreted as follows. For boolean fields, the first character is examined. If it is "T" or "t," the value is taken to be true. Otherwise, it is taken to be false. For string fields, the defaults are changed only if a new value is explicitly specified.

Following is a list of the escape sequences used by the editor:

<i>Name</i>	<i>Meaning</i>
al	Str: Insert line
am	Bool: automatic margins, i.e. cursor wraps at EOL
bc	Str: backspace character
bs	Bool: terminal can backspace with ^h
bw	Bool: backspace wraps to end of cur lin
ce	Str: kill to eol
cl	Str: clear screen
cm	Str: cursor motion
cr	Str: carriage return
dc	Str: delete char
dl	Str: delete line
ei	Str: end insert mode
g0 *	Str: graphics ULC (underline char)
g1 *	Str: graphics VBAR (vertical bar)
g2 *	Str: graphics LLC (lower left corner)
g3 *	Str: graphics URC (upper right corner)
g4 *	Str: graphics LRC (lower right corner)
g5 *	Str: graphics HBAR (horizontal bar)
g6 *	Str: graphics TEE
g7 *	Str: graphics INVTEE (inverted tee)
g8 *	Str: graphics TEEONL (tee on left)
g9 *	Str: graphics TEEONR (tee on right)
ge *	Str: graphics mode off
gs *	Str: graphics mode off
ho	Str: home cursor
ic	Str: insert char

im	Str:	insert mode on
ip	Str:	pad after insert char
is	Str:	terminal initialization string
nd	Str:	cursor right
nl	Str:	newline
ti	Str:	terminal initialization, part 2
te	Str:	undoes effects of is/ti
uc	Str:	underline following character
ue	Str:	underline mode off
uk *	Str:	unlock terminals
up	Str:	cursor up
us	Str:	underline mode on
x0 *	Str:	graphics CROSS
x1 *	Str:	graphics blot (region start mark)
ee *	Bool:	EXECUTE at top level means ENTER
fe *	Bool:	Filter in a menu means EXECUTE
df *	Bool:	disable flow-control (^s ^q)
gu *	Str:	unknown graphics char
gd *	Str:	graphics dot
ANSI *	Bool:	does the terminal support ansi style seqs?
NOSTRIP *	Bool:	should we turn off istrip and get 8 bits?
md *	Bool:	memory-mapped screen supported
abyte *	Num:	attribute byte for memory mapped normal text
ubyte *	Num:	attribute byte for memory mapped underlined text

The terminal capabilities that are not standard sequences found in `/etc/termcap` are flagged with asterisks. There are many more terminal capabilities that are defined by `termcap()`; however, those that are not listed here are not used by the editor.

The `tconvert(1)` program is used to convert termcap-defined terminal capabilities into a record array of input and output mappings in the structured `def.trm` file. The `tconvert(1)` program reads the first record from `def.trm` to get the default input mappings and output graphics mappings. It then reads the file `/etc/termcap`, or the file the TERMCAP environment variable points to, builds a record array of the termcap data while merging in the default input and output mappings, and appends the new data to the output file. The `tdigest(1)` program must then be used to produce a rapid-access, binary version of `def.trm`, named `terms.bin`. This is the data INed actually uses.

It is recommended that `tdigest(1)` be run from within INed because if `tdigest(1)` does not complete successfully, it may produce a zero-length `terms.bin` file. The zero-length `terms.bin` file would prevent the editor from being re-entered. (Note, however, that `tdigest(1)` first backs up the existing `terms.bin` file into `terms.bin.bak`.) From within the editor, you can continue to make changes to the terminal descriptions and run `tdigest(1)` until it executes successfully, at which point you can exit the editor and re-enter, in order to test the changes. The `terms.bin` file is read when the editor is started, when returning from a full screen command, and when returning from popup boxes that use more than the full screen.

FILES

/usr/lib/INed/termcap/def.trm
/usr/lib/INed/termcap/terms.bin
/etc/termcap

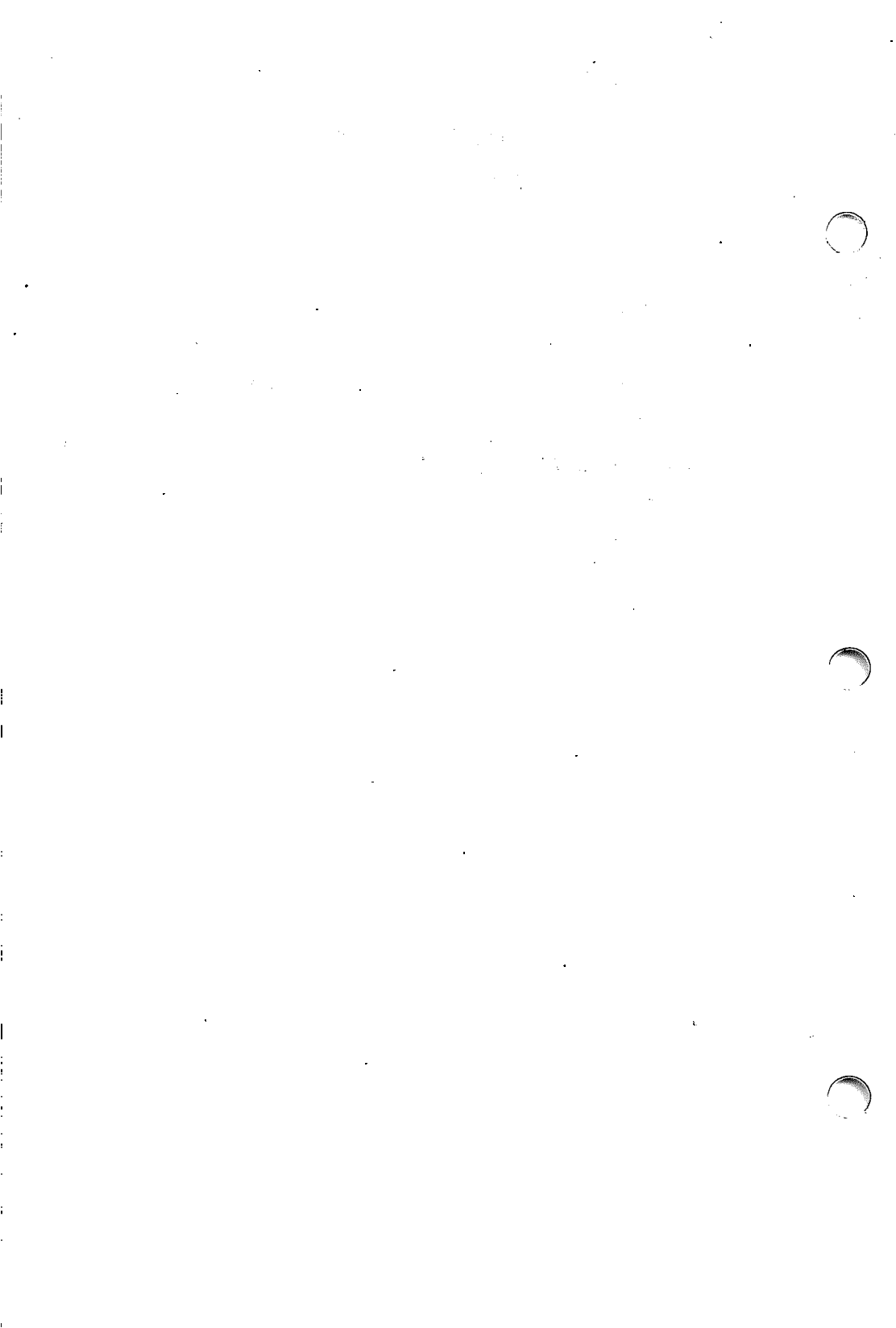
SEE ALSO

tconvert(1), tdigest(1), termcap().

TEN/PLUS User Interface Installation Instructions

CONTENTS

1. OVERVIEW	1
2. INSTALLING THE TEN/PLUS USER INTERFACE	2
3. SETTING THE TERMINAL ENVIRONMENT	4
3.1 Introduction	4
3.2 Determining Your TERM Variable	4
3.3 Setting Your TERM Variable	4



TEN/PLUS* User Interface Installation Instructions

1. OVERVIEW

The TEN/PLUS User Interface is installed on your fixed disk using the `sysadm` utility. It takes about 1.7 MB of space and requires that the Core subset already be installed. You should read the following documents before attempting to install the TEN/PLUS User Interface on your system:

- “INTERACTIVE UNIX Operating System Installation Instructions”
- “INTERACTIVE UNIX Operating System Maintenance Procedures”
- “INTERACTIVE UNIX Operating System Primer”

These documents were delivered with your INTERACTIVE UNIX Operating System.

2. INSTALLING THE TEN/PLUS USER INTERFACE

- To begin the installation, use the System Administration command, `sysadm`, or log in as `sysadm` to access the Main menu. Your screen will look similar to this:

```

                                SYSTEM ADMINISTRATION

1 diskmgmt      disk management menu
2 filemgmt     file management menu
3 machinmgmt   machine management menu
4 packagemgmt  package management menu
5 softwaregmt  software management menu
6 syssetup     system setup menu
7 ttygmt       tty management menu
8 usermgmt     user management menu

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, q to QUIT:

```

- Type 5 to access the Software Management Menu. Your screen will then look similar to this:

```

                                SOFTWARE MANAGEMENT

1 installpkg   install new software package onto built-in disk
2 listpkg     list packages already installed
3 removepkg   remove previously installed package from built-in disk
4 runpkg      run software package without installing it

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, ^ to GO BACK, q to QUIT:

```

- Select option 1, `installpkg`. The system prompts you to insert the first diskette into the diskette drive. The screen will look similar to this:

```

Insert the removable medium for the package you want to
install into the diskette drive.
Press <RETURN> when ready. Type q to quit.

```

4. Insert the TEN/PLUS User Interface diskette into the diskette drive. The system asks you to confirm that this is the package you want to install. Use **RETURN** to start the installation process.

```

Install the TEN/PLUS USER INTERFACE package? (y):
Installing the TEN/PLUS USER INTERFACE.
Copyright (c) 1987 AT&T
All Rights Reserved
The following files are being installed:
/usr/bin/cleandir
/usr/bin/e
/usr/bin/ffill
/usr/bin/ghost
.
.
.
/usr/lib/INed/help/histhelp
/usr/options/t1.name
Floppy diskette number 1 is complete
Remove floppy and insert floppy number 2
Type <return> when ready:

```

5. Remove the first diskette and insert the next one. Then use **RETURN** to continue.

```

The following files are being installed:
/usr/lib/INed/help/keys.map
.
.
.
/usr/lib/INed/termcap/terms.bin
/usr/options/t1.name
Floppy diskette number 2 is complete
Installation of the TENPLUS USER INTERFACE is complete.
You may now remove the medium from the diskette drive.

```

(The names of some of the files have been omitted for the sake of brevity.)

6. The TEN/PLUS User Interface is now installed on your fixed disk. Your terminal must be correctly configured before you can use TEN/PLUS. The following section explains how to configure your terminal.

3. SETTING THE TERMINAL ENVIRONMENT

3.1 Introduction

The TEN/PLUS User Interface is designed to be terminal-independent and can be configured to support most terminals. Before you can use the system, you must determine that your terminal is correctly configured to support the TEN/PLUS User Interface. The characteristics of each terminal must be established and made accessible to TEN/PLUS.

Your system is delivered with a number of predefined terminal types. This section explains how to set the terminal type on your system using a currently supported terminal configuration (see §3.2).

3.2 Determining Your TERM Variable

A default terminal type is usually established for you when the system is installed. You may determine the default terminal type on your system by typing:

```
$ echo $TERM
```

at the system prompt. The system displays the value of the `$TERM` variable on your system. Your screen will look similar to this:

```
AT386
```

If `TERM` is not set, the system will simply return the prompt.

If you are using the standard AT* console, the `TERM` variable should be assigned `atcon` or `AT386` as its value. These two values are synonymous and can be typed in either upper- or lower-case. If you are not using a standard AT console or if no `TERM` variable is set, you must assign the `TERM` variable the correct value.

3.3 Setting Your TERM Variable

If all the users on your system are using the same terminal type, you may set the terminal variable for the entire system by adding the following two lines to the file `/etc/profile`:

```
TERM=termvariable  
export TERM
```

where *termvariable* is replaced by the variable name of your terminal. The table below lists all of the valid terminal types available

on your system. If you are not using one of the supported terminals, you must contact your vendor to determine how you can update your system to include a correct terminal description.

Here are the terminal types supported by the TEN/PLUS User Interface. Use this table to determine the correct **TERM** variable for your system.

<i>Terminal Type</i>	<i>Variable</i>
Typical dumb terminals	default
DEC* VT100*	vt100
DEC VT220*	vt220
DEC VT240*	vt240
IBM* 3101	ibm3101
AT&T 5425	5425
AT&T 4425	4425
Esprit 6310	esp6310
IBM AT or compatible	atcon or AT386
WYSE 60*	wyse60

If several different terminal types are attached to your system, each user can define an appropriate **TERM** variable. There are two ways to define a **TERM** variable. You may assign a value for the current login session, or you may set it permanently by modifying a user's `.profile` file.

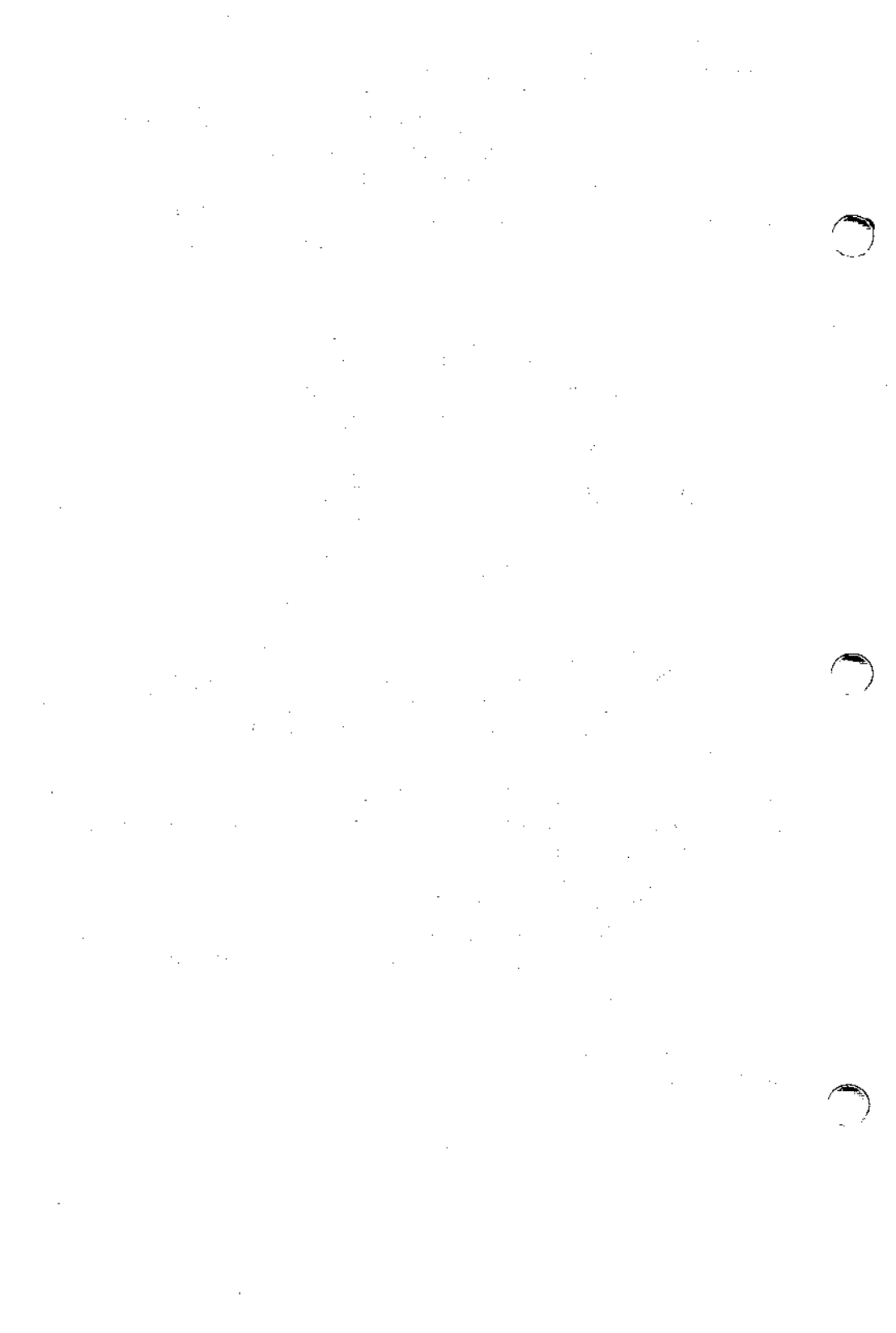
For example, if you are using a VT100 terminal and want to set the **TERM** variable for the current login session only, type the following commands at the system prompt:

```
$ TERM=vt100
  export TERM
```

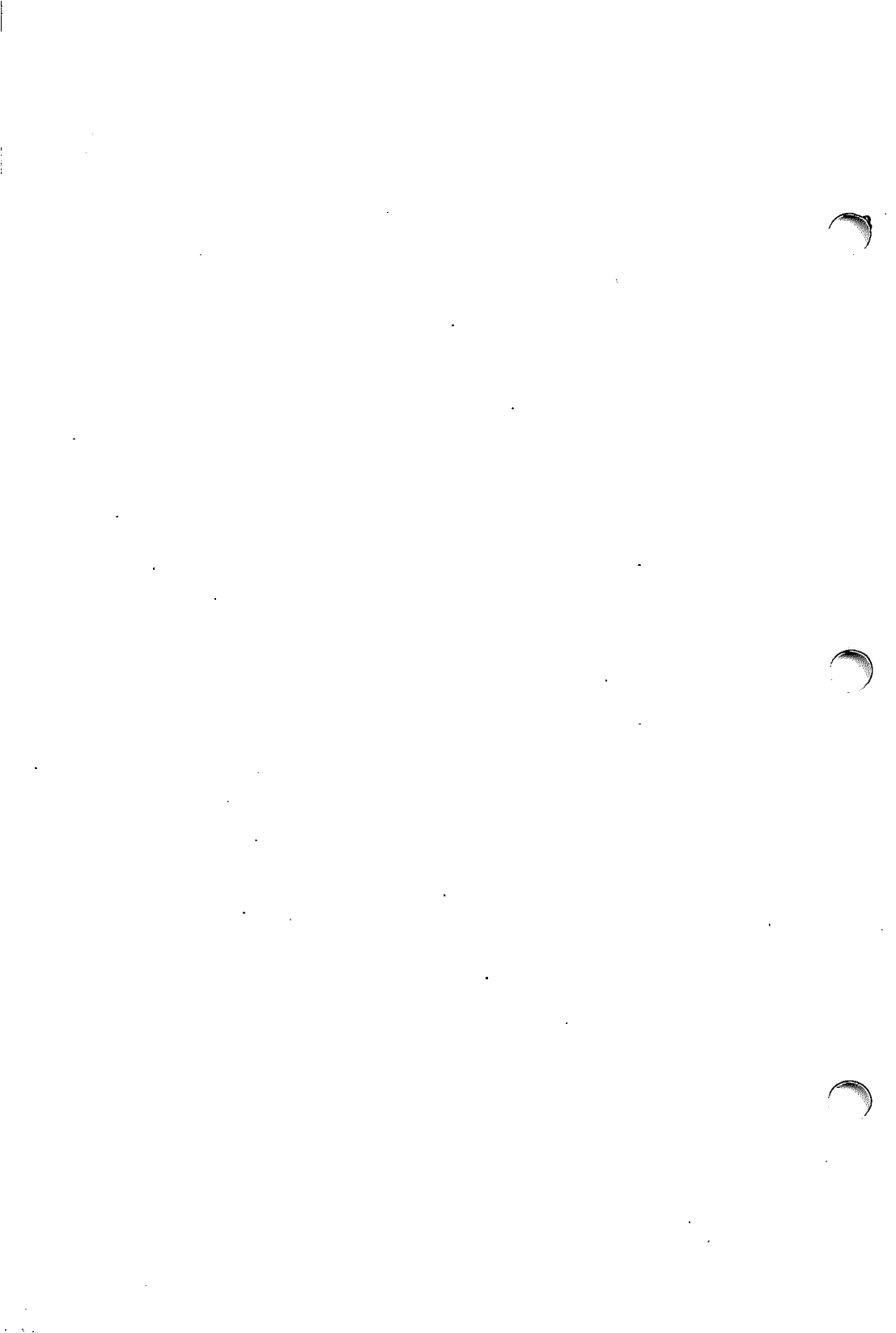
If you want to set the **TERM** variable permanently, edit the file `.profile` located in your **HOME** directory to include these lines:

```
TERM=vt100
export TERM
```

To initialize the new terminal type, log out of the system, then log back in again.



NOTES





INTERACTIVE



A Kodak Company