AIX Version 3 for
RISC System/6000™

Editing Concepts and Procedures

SC23-2212-00

First Edition (March 1990)

This edition of the Editing Concepts and Procedures for IBM RISC System/6000 applies to IBM AIX Version 3 for RISC System/6000 and to all subsequent releases until otherwise indicated in new releases or technical newsletters.

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS MANUAL "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Any reference to an IBM licensed program in this publication is not intended to state or imply that you can use only IBM's licensed program. You can use any functionally equivalent program instead.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Department 997, 11400 Burnet Road, Austin, Texas 78758–3493. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks and Acknowledgements

The following trademarks and acknowledgements apply to this information:

AIX is a trademark of International Business Machines Corporation.

IBM is a registered trademark of International Business Machines Corporation.

INed is a trademark of INTERACTIVE Systems Corporation.

RISC System/6000 is a trademark of International Business Machines Corporation.

RT is a trademark of International Business Machines Corporation.

UNIX was developed and licensed by AT&T and is a registered trademark of AT&T Corporation.

About This Book

This book, *AIX Editing Concepts and Procedures for RISC System/6000,* describes some of the editors that are available for use with the RISC System/6000 system unit and includes suggestions of some effective uses of these editors. This book also contains procedural information for using particular functions provided with editors.

Who Should Use This Book

This book is intended for users and programmers who use editors to create and modify programs, text files, and other lines of data in the AIX operating environment.

How This Book is Organized

This book contains six chapters:

- Chapter 1. Editors Overview, introduces the AIX editors and their use.
- Chapter 2. Editing with the ed Editor, describes how to use the ed editor and its subcommands.
- Chapter 3. Editing with the INed Editor, describes how to use the INed editor.
- Chapter 4. Editing with the vi Editor, describes how to use the vi editor and its options.
- Chapter 5. Addressing Lines in a File with the ex Editor, describes how to use the ex editor.
- Chapter 6. Editing a File with the edit Editor, describes how to use the ed editor and its subcommands.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, keywords, files, directories, and other items whose names are predefined by the system.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Related Publications

The following books contain information related to using AIX editors:

- *AIX General Concepts and Procedures for IBM RISC System/6000*, Order Number SC23–2202.
- *AIX Commands Reference for IBM RISC System/6000*, Order Number SC23–2199.
- *AIX Files Reference for IBM RISC System/6000*, Order Number SC23–2200

Ordering Additional Copies of This Book

To order additional copies of this book, use Order Number SC23–2212.

Table of Contents

Editors Overview

An editor is a program you can use to create and change files containing text, programs, or other data. An editor does not provide the formatting and printing capabilities of a text or word processor.

You start an editor by entering a command on the command line (for example, `vi` for the **vi** editor). Then, by interactively entering a combination of subcommands and data, you can:

- Create, read, and write files
- Display and search data
- Add, replace, and remove data
- Move and copy data
- Run AIX commands.

Your editing takes place in an edit buffer, which you can save or discard.

AIX Editors

The main editors available to you in AIX are: the **INed** editor, the **vi** editor, and the **ed** editor. Each has its own set of subcommands and rules for entering them and its own way of displaying subcommands and data.

INed Editor

The **INed** editor is a full–screen editor that allows you to:

- Enter subcommands by command keys rather than on a command line
- Edit multiple files in multiple windows on the screen
- Scroll the screen horizontally (as well as vertically) through files
- Move, copy, and delete blocks (as well as lines) of data.

The **INed** editor also includes:

- A file manager for:
  - Creating, changing, deleting, and recovering files and directories
  - Moving among files and directories
  - Moving and copying files and directories
  - Restricting access to files and directories
- Commands for working with structured files and their histories
- Commands for limited text formatting.

vi Editor

The **vi** editor is a full–screen editor that is available on many UNIX systems and popular with many programmers. The **vi** editor has a command mode for entering subcommands and an input mode for entering data. The **vi** editor allows you to:

- Move and copy partial (as well as complete) lines of data
- Set options to select editing preferences.

The **vi** editor includes:

- An extensive set of subcommands
- A line editing mode.

ed Editor

The **ed** editor is a line editor that is available on all UNIX systems and accessible from any terminal at any speed. Like the **vi** editor, the **ed** editor has command and input modes, but the **ed** editor allows you to move and copy only complete lines of data. The **ed** editor can be used to:

- Edit very large files
- Edit within a shell program.

The **ed** editor includes:

- Powerful search and replace subcommands
- Commands and syntax fundamental to other programs such as grep, awk, sed, and lex.

Editing with the ed editor

The **ed** editor is a line editor. To edit a new or existing file, you enter the **ed** command at the prompt on the command line. The file is placed into the edit buffer as lines of text that can be addressed (that is, referred to) by number. The current line indicates your position within the file, and you move the current line to change position. The current line starts out as the last line in the file and moves as lines in the file are changed.

Modes

The **ed** editor has two modes of operation:

- Command mode, in which keystrokes are interpreted as **ed** subcommands to be carried out
- Text input mode, in which keystrokes are interpreted as text to be added to the file.

You start editing in command mode. The **a**, **i**, and **c** subcommands switch you to text input mode, and a **.** (period) on a line by itself switches you back.

Subcommands

An **ed** subcommand allows you to tell the editor what operation to perform on how many of which objects.

With one or more subcommands, you can:

- Display prompts and helps
- Display line addresses or lines of text
- Position the current line
- Search or mark text
- Add, change, or delete text
- Move or copy text
- Undo additions, changes, deletions, moves, or copies
- Run AIX commands
- Manipulate files
- Quit the **ed** editor

Subcommands operate on the following objects:

- The current line
- Lines indicated by line addresses
- Character strings or lines indicated by matches to literal strings or regular expressions from searches
- Lines indicated by marks (**a** through **z**).

Related Information

The following items are discussed in this chapter:

"How to Start the ed Editor" contains a procedure for starting the editor.

"How to Display Prompts or Helps with the ed Editor" contains procedures for starting or stopping the display of prompts or help messages.

"How to Address Lines and Display Line Addresses with the ed Editor" contains procedures for addressing a line or group of lines and for displaying a line's address.

"How to Position the Current Line with the ed Editor" contains a procedure for positioning the current line by line number, search, or mark.

"How to Display Text with the ed Editor" contains procedures for displaying a line, group of lines, or selected lines.

"How to Add Text with the ed Editor" contains procedures for adding text after or before a line or selected lines.

"How to Search Text with the ed Editor" contains a procedure for searching text for a literal string or regular expression.

"How to Mark Text with the ed Editor" contains a procedure for marking lines.

"How to Change Text with the ed Editor" contains procedures for substituting text within or for changing a line, group of lines, or selected lines.

"How to Delete Text with the ed Editor" contains procedures for deleting text within or for deleting a line, group of lines, or selected lines.

"How to Move or Copy Text with the ed Editor" contains procedures for moving or for copying a line, group of lines, or selected lines.

"How to Undo Text with the ed Editor" contains procedures for undoing the last add, change, move, copy, or delete.

"How to Run AIX Commands from the ed Editor" contains procedures for running one or more AIX commands.

"How to Manipulate Files with the ed Editor" contains procedures for naming, reading, writing, and editing files.

"How to Quit the ed Editor" contains procedures for quitting with or without writing.

Reference information supporting the following command is located in AIX Commands Reference for IBM RISC System/6000:

The **ed** command contains reference information (including syntax, description, flags, and subcommands) for the **ed** editor.

How to Start the ed Editor

Prerequisite Tasks or Conditions

1. Start AIX.
2. Be at the prompt on the command line.

Procedure

1. Type `ed`.
2. Press the spacebar and type the name of an existing or new file to edit (or skip this step to edit a new file with no name).
3. Press the **Enter** key to start editing with the ed editor (in command mode).

   For an existing file, the number of characters in the file is displayed. For a new file, a ? (question mark) and the name of the file are displayed.

Requesting Help with the ed Editor

h The **h**elp subcommand gives a short explanation (help message) for the most recent ? diagnostic or error message.

H The **H**elp subcommand causes the **ed** command to display the help messages for all subsequent ? diagnostics. The **H** subcommand also explains the previous ? if there was one. The **H** subcommand alternately turns this mode on and off; it is initially off.

How to Display Prompts or Helps with the ed Editor

Prerequisite Tasks or Conditions

1. Start the ed editor.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedures

To Start or Stop Displaying Prompts:

1. Type **P**.
2. Press the **Enter** key.

   An * (asterisk) starts or stops being displayed as a prompt from the ed editor.

To Display the Last Help Message:

1. Type **h**.
2. Press the **Enter** key.

   A help message is displayed for the last ? (question mark) response from the ed editor.

To Start or Stop Displaying Help Messages:

1. Type **H**.
2. Press the **Enter** key.

   Help messages start or stop being displayed for ? (question mark) responses from the ed editor.

Editing a File with the ed Editor

The ed subcommands for editing a file allow you to perform the following tasks:

Adding Text to a File

Changing Lines

Changing the Current Line

Copying Lines

Deleting Lines

Displaying Text and Finding Out the Current Line

Joining Lines

Locating Text in a File

Making Global Changes

Making Substitutions

Marking Lines

Moving Text

Saving Text

Undoing a Change.

ed Subcommands for Editing a File

In the following lists of **ed** subcommands, default addresses are shown in parenthese. Do not key in the parentheses. The address . (period) refers to the current line. When a . (period) is shown in the first position of an otherwise empty line, it is the signal to return to command mode.

Adding Text

(.)**a**
<*Text*>
.

The **a**ppend subcommand adds text to the buffer after the addressed line. The **a** subcommand sets the current line to the last inserted line, or, if no lines were inserted, to the addressed line. Address 0 causes the **a** subcommand to add text to the beginning of the buffer. See addressing lines in a file for more information about addressing.

Type your text, pressing Enter at the end of each line. If you do not press Enter at the end of each line, the ed program automatically moves your cursor to the next line after you fill a line with characters. However, the ed program treats everything you type before you press Enter as one line, regardless of how many lines it takes up on the screen. This may make it difficult to edit your file. When you have entered all your text, enter a . (period) at the start of a new line.

(.)**i**
<*Text*>
.

The **i**nsert subcommand inserts text before the addressed line and sets the current line to the last inserted line. If no lines are inserted, **i** sets the current line to the addressed line. You cannot use a 0 address for this subcommand. See addressing lines in a file for more information about addressing.

Type your text, pressing Enter at the end of each line. If you do not press Enter at the end of each line, the ed program automatically moves your cursor to the next line after you fill a line with characters. However, the ed program treats everything you type before you press Enter as one line, regardless of how many lines it takes up on the screen. This may make it difficult to edit your file. When you have entered all your text, enter a . (period) at the start of a new line. The **i** subcommand differs from the **a** subcommand only in the placement of the text.

Changing Lines

(., .)**c**
<*Text*>
.

The **c**hange subcommand deletes the addressed lines, then replaces them with new input. the **c** subcommand sets the current line to the last new line

of input, or, if there were none, to the first line that was not deleted. See addressing lines in a file for more information about addressing.

The **c** subcommand first deletes the line(s) you want to replace and then lets you enter the new lines. Type in the text you want, pressing Enter at the end of each line. When you have entered all of the new text, type a . (period) on a line by itself.

Copying Lines

(., .)t*A*

The **t** (transfer) subcommand inserts a copy of the addressed lines after address *A*. The **t** subcommand accepts address 0 (for inserting lines at the beginning of the buffer). See addressing lines in a file for more information about addressing.

The **t** subcommand sets the current line to the last line copied.

Deleting Lines

(., .)**d**

The **d**elete subcommand removes the addressed lines from the buffer. The line after the last line deleted becomes the current line. If the deleted lines were originally at the end of the buffer, the new last line becomes the current line. See addressing lines in a file for more information about addressing.

Displaying Text and Finding Out the Current Line

(.,.)**l**

The **l**ist subcommand displays the addressed lines. The **l** subcommand wraps long lines and, unlike the **p** subcommand, represents nonprinting characters, either with mnemonic overstrikes or in hexadecimal notation. An **l** subcommand may be appended to any ed subcommand except: **e**, **f**, **r**, or **w**. For example, the **dl** subcommand deletes the current line and displays the new current line.

(., .)**n**

The **n**umber subcommand displays the addressed lines, each preceded by its line number and a tab character (displayed as blank spaces); **n** sets the current line to the last line displayed. An **n** subcommand may be appended to any ed subcommand except: **e**, **f**, **r**, or **w**. For example, the **dn** subcommand deletes the current line and displays the new current line and line number.

(., .)**p**

The **p**rint subcommand displays the addressed line(s) and sets the current line set to the last line displayed. A **p** subcommand may be appended to any ed subcommand except: **e**, **f**, **r**, or **w**. For example, the **dp** subcommand deletes the current line and displays the new current line.

(.)=

Without an address, the = (equal sign) subcommand displays the current line number. With the address $, = displays the number of the last line in the buffer. The = subcommand does not change the current line and cannot be included in a **g** subcommand or **v** subcommand list.

Joining Lines

(.,.+1)**j**

The **j**oin subcommand joins contiguous lines by removing the intervening new–line characters. If given only one address, the **j** subcommand does nothing. See addressing lines in a file for more information about addressing. (For splitting lines, see the **s** subcommand.)

Making Global Changes

(1,$)**g**/*Pattern*/*Subcommand–List*

The **g**lobal subcommand first marks every line that matches the *Pattern*.

The pattern can be a fixed character string or a regular expression. Then, for each marked line, this subcommand sets the current line to that line and executes *Subcommand–List*. A single subcommand or the first subcommand of a list should appear on the same line with the **g** subcommand; subsequent subcommands should appear on separate lines. Except for the last line, each of these lines should end with a \ (backslash).

The *Subcommand–List* can include the **a**, **i** , and **c** subcommands and their input. If the last command in *Subcommand–List* would normally be the . (period) that ends input mode, the . is optional. If there is no *Subcommand–List*, the current line is displayed. The *Subcommand–List* cannot include the **g** , **G**, **v**, or **V** subcommands.

Note: The **g** subcommand is similar to the **v** subcommand, which executes *Subcommand–List* for every line that does not contain a match for the pattern.

(1,$)**G**/*Pattern*/

The interactive **G** (**G**lobal) subcommand first marks every line that matches the *Pattern*, then displays the first marked line, sets the current line to that line, and waits for a subcommand. A pattern can be a fixed character string or a regular expression.The **G** subcommand accepts any but the following ed subcommands: **a**, **c**, **i**, **g**, **G**, **v**, and **V**. After the subcommand finishes, the **G** subcommand displays the next marked line, and so on. The **G** subcommand takes a new–line character as a null subcommand. A :& (colon ampersand) causes the **G** subcommand to execute the previous subcommand again, if there was one. The **G** subcommand can be terminated by pressing INTERRUPT (Ctrl–C).

(1,$)**v**/*Pattern*/*Subcommand–List*

The **v** subcommand executes the subcommands in *Subcommand–List* for each line that does not contain a match for the *Pattern*. A pattern can be a fixed character string or a regular expression.The **v** subcommand accepts any but the following ed subcommands: **a**, **c**, **i**, **g**, **G**, and **V**.

Note: The **v** subcommand is a complement for the global subcommand, the **g** subcommand, which executes *Subcommand–List* for every line that does contain a match for the pattern.

(1,$)**V**/*Pattern*/

The **V** subcommand first marks every line that does not match the *Pattern*, then displays the first marked line, sets the current line to that line, and waits for a subcommand. A pattern can be a fixed character string or a regular expression. The **V** subcommand accepts any but the following ed subcommands: **a**, **c**, **i**, **g**, **G**, and **v**.

Note: The **V** subcommand complements the **G** subcommand, which marks the lines that do match the pattern.

Making Substitutions

(.,.)**s**/*Pattern*/*Replacement*/

(.,.)**s**/*Pattern*/*Replacement*/**g**

The **s**ubstitute subcommand searches each addressed line for a string that matches the pattern and then replaces the string with the specified *Replacement* string. A pattern can be a fixed character string or a regular expression. See addressing lines in a file if you need more information about addressing. Without the global indicator (**g**), the **s** subcommand

replaces only the first matching string on each addressed line. With the **g** indicator, the **s** subcommand replaces every occurrence of the matching string on each addressed line. If the **s** subcommand does not find a match for the pattern, it returns the error message ? (question mark). Any character except a space or a new–line character can separate (delimit) the pattern and *Replacement*. The **s** subcommand sets the current line to the last line changed.

An & (ampersand) in the *Replacement* string is a special symbol that has the same value as the *Pattern* string. For example, the subcommand `s/are/&n't/` has the same effect as the subcommand `s/are/aren't/` and replaces `are` with `aren't` on the current line. A \& (reverse slash ampersand) removes this special meaning of & in *Replacement*.

A subpattern is part of a pattern enclosed by the strings \(and \); the pattern works as if the enclosing characters were not present. In *Replacement*, the characters *\Number* refer to strings that match subpatterns; *Number*, a decimal number, refers to the occurrence of subpattern specified by *Number*, counting from the left. (For example, `s/\(t\)\(h\)\(e\)/t\1\2ose)` replaces `the` with `those` if there is a match for the pattern `the` on the current line.) Whether subpatterns are nested or in a series, *\Number* refers to the occurrence specified by *Number*, counting from the left of the delimiting characters, \).

The % (percent sign) character, when used by itself as *Replacement*, causes the **s** subcommand to use the previous *Replacement* again. The % character does not have this special meaning if it is part of a longer *Replacement* or if it is preceded by a \ (reverse slash).

Lines may be split by substituting new–line characters into them. In *Replacement*, the sequence \Enter quotes the new–line character (not displayed) and moves the cursor to the next line for the remainder of the string. New–lines cannot be substituted as part of a **g** subcommand or **v** subcommand list.

Marking Lines

(.)**k***x* The **mark** subcommand marks the addressed line with the name specified by *x*, which must be a lowercase ASCII letter. See addressing lines in a file if you need more information about addressing. The address *'x* (single quotation mark before the marking character) then addresses this line. The **k** subcommand does not change the current line.

Moving Text

(.,.)**m***A* The **move** subcommand repositions the addressed line(s). The first moved line follows the line addressed by *A*. Address 0 for *A* cuases **m** to move the addressed line(s) to the beginning of the file. Address *A* cannot be one of the lines to be moved. See addressing lines in a file for more information about addressing. The **m** subcommand sets the current line to the last moved line.

Saving Text

(1,$)**w** *File* The **write** subcommand copies the addressed lines from the buffer to the file specified by *File*. If the file does not exist, the **w** subcommand creates it with permission code 666 (read and write permission for everyone), unless the **umask** setting specifies another file creation mode. (For information about file permissions, see File and Directory Access Overview.) The **w** subcommand does not change the default file name (unless *File* is the first

file name used since you started the ed program). If you do not provide a file name, the **w** subcommand uses the default file name, if any (see the **e** subcommand and the **f** subcommand). The **w** subcommand does not change the current line.

If the **ed** program successfully writes the file, it displays the number of characters written. If you specify *!AIX–Command* instead of a file name, the **w** subcommand reads the output of the AIX command specified. The **w** subcommand does not save shell command names as default file names.

Note: 0 is not a legal address the the **w** subcommand,. Therefore, it is not possible to create an empty file with **ed**.

Undoing a Change

u The **u** (undo) subcommand restores the buffer to the state it was in before it was last modified by an ed subcommand. The subcommands that the **u** subcommand cannot undo are: **e**, **f**, and **w**.

How to Add Text with the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedures

To Add Text after or before a Line:

1. Type the address of the line to add text after or before (or skip this step to add text after or before the current line).
2. Type:

   a to add (append) text after the line
   i to add (insert) text before the line.

3. Type **p**, **l**, or **n** to display the added text (or skip this step).
4. Press the **Enter** key.
5. Type the text. (Press the **Enter** key to start a new line.)
6. Press the **Enter** key, type . (period), and press the **Enter** key to return (from text entry mode) to command mode.

To Add Text after or before Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by the search pattern in step 4
   v to select the lines not indicated by the search pattern in step 4.

3. Type / (slash).
4. Type the pattern for a search.
5. Type / (slash).
6. Type:

   a to add (append) text after the selected lines
   i to add (insert) text before the selected lines.

7. Type **p**, **l**, or **n** to display the added text (or skip this step).
8. Type \ (reverse slash) and press the **Enter** key.
9. Type the text. (Type \ (reverse slash) and press the **Enter** key to start a new line.)
10. Press the **Enter** key.

How to Change Text with the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedures

To Substitute Text within a Line or Group of Lines:

1. Type the address of a line or group of lines to substitute text within (or skip this step to substitute text within the current line).
2. Type **s**.
3. Type / (slash).
4. Type *OldPattern* to substitute for the character string indicated by *OldPattern*, where *OldPattern* is the pattern for a search.
5. Type:

   /NewString/ to substitute the character string *NewString* for the first *OldPattern* within each line
   */NewString/***g** to substitute *NewString* for every *OldPattern* within each line.

6. Type **p**, **l**, or **n** to display the changed text (or skip this step).
7. Press the **Enter** key.

To Substitute Text within Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by *Pattern* in step 4
   v to select the lines not indicated by *Pattern* in step 4.

3. Type / (slash).
4. Type *Pattern*, where *Pattern* is the pattern for a search.
5. Type / (slash).
6. Type **s**.
7. Type:

   /NewString/
   to substitute the character string *NewString* for the first *Pattern* within each selected line

   */NewString/***g**
   to substitute *NewString* for every *Pattern* within each selected line

   /OldPattern/NewString/
   to substitute *NewString* for the first character string indicated by *OldPattern* within each line selected by *Pattern*, where *OldPattern* is the pattern for a search

   */OldPattern/NewString/***g**
   to substitute *NewString* for every *OldPattern* within each line selected by *Pattern*.

8. Type **p**, **l**, or **n** to display the changed text (or skip this step).
9. Press the **Enter** key.

To Change a Line or Group of Lines:

1. Type the address of a line or group of lines to change (or skip this step to change the current line).
2. Type **c**.
3. Type **p**, **l**, or **n** to display the changed text (or skip this step).
4. Press the **Enter** key.
5. Type the new text. (Press the **Enter** key to start a new line.)
6. Press the **Enter** key, type . (period), and press the **Enter** key to return (from text entry mode) to command mode.

To Change Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by the search pattern in step 4
   v to select the lines not indicated by the search pattern in step 4.

3. Type / (slash).
4. Type the pattern for a search.
5. Type / (slash).
6. Type **c**.
7. Type **p**, **l**, or **n** to display the changed text (or skip this step).
8. Type \ (reverse slash) and press the **Enter** key.
9. Type the new text. (Type \ (reverse slash) and press the **Enter** key to start a new line.)

10. Press the **Enter** key.

To Join Lines:

1. Type the address of a group of lines to join (or skip this step to join the current and next lines).
2. Type **j**.
3. Type **p**, **l**, or **n** to display the joined lines (or skip this step).
4. Press the **Enter** key.

To Split a Line:

1. Type the address of a line to split (or skip this step to split the current line).
2. Type **s**.
3. Type / (slash).
4. Type *Pattern* to split the line after the character string indicated by *Pattern*, where *Pattern* is the pattern for a search.
5. Type / (slash).
6. Type *Pattern*.
7. Type \ (reverse slash), press the **Enter** key, and type / (slash).
8. Type **p**, **l**, or **n** to display the split line (or skip this step).
9. Press the **Enter** key.

How to Delete Text with the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedures

To Delete Text within a Line or Group of Lines:

1. Type the address of a line or group of lines to delete text within (or skip this step to delete text within the current line).
2. Type **s**.
3. Type / (slash).
4. Type *Pattern* to delete the character string indicated by *Pattern*, where *Pattern* is the pattern for a search.
5. Type:

//	(two slashes) to delete the first *Pattern* within each line
//**g**	to delete every *Pattern* within each line.

6. Type **p**, **l**, or **n** to display the deletion (or skip this step).
7. Press the **Enter** key.

To Delete Text within Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by *Pattern* in step 4
   v to select the lines not indicated by *Pattern* in step 4.
3. Type / (slash).
4. Type *Pattern*, where *Pattern* is the pattern for a search.
5. Type / (slash).
6. Type **s**.
7. Type:

   /// (three slashes) to delete the first *Pattern* within each selected line
   ///**g** to delete every *Pattern* within each selected line
   /*OtherPattern*//
   to delete the first character string indicated by *OtherPattern* within each line selected by *Pattern*, where *OtherPattern* is the pattern for a search
   /*OtherPattern*//**g**
   to delete every *OtherPattern* within each line selected by *Pattern*.
8. Type **p**, **l**, or **n** to display the deletions (or skip this step).
9. Press the **Enter** key.

To Delete a Line or Group of Lines:

1. Type the address of a line or group of lines to delete (or skip this step to delete the current line).
2. Type **d**.
3. Type **p**, **l**, or **n** to display the deletion (or skip this step).
4. Press the **Enter** key.

To Delete Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by the search pattern in step 4
   v to select the lines not indicated by the search pattern in step 4.
3. Type / (slash).
4. Type the pattern for a search.
5. Type / (slash).
6. Type **d**.
7. Type **p**, **l**, or **n** to display the deletions (or skip this step).
8. Press the **Enter** key.

How to Mark Text with the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedure

1. Type the address of the line to mark (or skip this step to mark the current line).
2. Type **k***Letter*, where *Letter* is the letter (**a** through **z**) for a mark.
3. Press the **Enter** key.

How to Move or Copy Text with the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedures

To Move a Line or Group of Lines:

1. Type the address of a line or group of lines to move (or skip this step to move the current line).
2. Type **m**.
3. Type the address of the line to move the text after (or skip this step to move the text after the current line).
4. Type **p**, **l**, or **n** to display the move (or skip this step).
5. Press the **Enter** key.

To Move Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by the search pattern in step 4
   v to select the lines not indicated by the search pattern in step 4.

3. Type / (slash).
4. Type the pattern for a search.
5. Type / (slash).
6. Type **m**.
7. Type the address of the line to move the text after (or skip this step to move the text after the current line).
8. Type **p**, **l**, or **n** to display the moves (or skip this step).
9. Press the **Enter** key.

To Copy a Line or Group of Lines:

1. Type the address of a line or group of lines to copy (or skip this step to copy the current line).
2. Type **t**.
3. Type the address of the line to copy the text after (or skip this step to copy the text after the current line).
4. Type **p**, **l**, or **n** to display the copy (or skip this step).
5. Press the **Enter** key.

To Copy Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by the search pattern in step 4
   v to select the lines not indicated by the search pattern in step 4.
3. Type / (slash).
4. Type the pattern for a search.
5. Type / (slash).
6. Type **t**.
7. Type the address of the line to copy the text after (or skip this step to copy the text after the current line).
8. Type **p**, **l**, or **n** to display the copies (or skip this step).
9. Press the **Enter** key.

How to Undo Text with the ed Editor

Prerequisite Tasks or Conditions

1. Add, change, move, copy, or delete text.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedure

1. Type **u**.
2. Press the **Enter** key.

   The last add, change, move, copy or delete done to the text is undone.

How to Display Text with the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type **.** (period), and press the **Enter** key.)

Procedures

To Display a Line or Group of Lines:

1. Type the address of the line or group of lines to display (or skip this step to display the current line).
2. Type:

   p to display the line or group of lines
   l to display the line or group of lines including nonprinting characters
   n to display the line or group of lines including line numbers.

3. Press the **Enter** key.

To Display Selected Lines:

1. Type the address of a group of lines to select from (or skip this step to select from all lines).
2. Type:

   g to select the lines indicated by the search pattern in step 4
   v to select the lines not indicated by the search pattern in step 4.

3. Type / (slash).
4. Type the pattern for a search.
5. Type / (slash).
6. Type:

   p to display the line or lines
   l to display the line or lines including nonprinting characters
   n to display the line or lines including line numbers.

7. Press the **Enter** key.

How to Search Text with the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type **.** (period), and press the **Enter** key.)

Procedure

1. Type:

   / (slash) to search forward from the current line
   ? (question mark) to search backward from the current line.

2. Type *Pattern*, where *Pattern* is a character string that is the pattern for a search.

   The pattern can be a literal string or a regular expression made up of literal characters and the special characters ^ (circumflex accent), $ (dollar sign), . (period), [(left bracket),] (right bracket), * (asterisk), \ (reverse slash), & (ampersand), and % (percent sign).

3. Press the **Enter** key.

   To repeat the search, type:

   / (slash) to search in the same direction
   ? (question mark) to search in the opposite direction

Procedures

To Name a File or Display a File's Name:

1. Type **f**.
2. Press the spacebar and type a new name for the file in the edit buffer (or skip this step to display the current name).
3. Press the **Enter** key.

   The name of the file in the edit buffer is displayed.

   and press the **Enter** key.

Manipulating Files with the ed Editor

The subcommands for manipulating files allow you to do the following tasks:

Editing Additional Files

Changing the Default File Name

Adding Another File to the Current File

Saving Text to a File.

How to Manipulate Files with the ed Editor

Prerequisite Tasks or Conditions

1. Start the ed editor.
2. Position the current line.
3. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

To Read a File:

1. Type the address of the line to insert the file after (or skip this step to insert the file after the current line).
2. Type **r**.
3. Press the spacebar and type the name of the file to read.
4. Press the **Enter** key.

   The number of characters read is displayed.

To Write a File:

1. Type the address of the line or group of lines to write to the file (or skip this step to write all lines).
2. Type **w**.
3. Press the spacebar and type the name of the file to write to (or skip this step to write to the current file).
4. Press the **Enter** key.

   The number of characters written is displayed.

To Edit a File:

1. Type:

   e to edit another file if the current file is unchanged or was written after the last change

   E to edit another file even though the current file was not written after the last change.

2. Press the spacebar and type the name of an existing or new file to edit (or skip this step to edit a new copy of the current file).
3. Press the **Enter** key.

   If **e** was entered and the current file was not written after the last change, a ? (question mark) is displayed. Enter **e** again to edit another file anyway. (The changes to the current file are lost.)

   For an existing file, the number of characters in the file is displayed. For a new file, a ? (question mark) and the name of the file are displayed.

---

ed Subcommands for Manipulating Files

Editing Additional Files

e *File*

The **e** (edit) subcommand first deletes any contents from the buffer, sets the current line to the last line of the buffer, and displays the number of characters read into the buffer. If the buffer has been changed since its contents were last saved (with the **w** subcommand), the ed program displays a ? (question mark) before it clears the buffer.

The **e** subcommand stores *File* as the default file name to be used, if necessary, by subsequent **e**, **r**, or **w** subcommands. (To change the name of the default file name, use the **f** subcommand.)

When an ! (exclamation mark) replaces *File*, the **e** subcommand takes the rest of the line as an AIX shell command and reads the command output.

The **e** subcommand does not store the name of the shell command as a default file name.

E *File* The **E**dit subcommand works like the **e** subcommand, with one exception: the **E** subcommand does not check for changes made to the buffer since the last **w** subcommand.

Changing the Default File Name

f [*File*] The **f** (file name) subcommand changes the default file name (the stored name of the last file used) to *File*, if *File* is given. If *File* is not given, the **f** subcommand prints the default file name. (The **e** subcommand stores the default file name.)

Adding Another File to the Current File

(**$**)**r** *File* The **r** (read) subcommand reads a *File* into the buffer after the addressed line. The **r** subcommand does not delete the previous contents of the buffer. When entered without the *File* parameter, the **r** subcommand reads the default file, if any, into the buffer (see the **e** subcommand and the **f** subcommand). The **r** subcommand does not change the default file name. Address 0 causes the **r** subcommand to read a file in at the beginning of the buffer. After it reads a file successfully, the **r** subcommand displays the number of characters read into the buffer and sets the current line to the last line read.

If ! (exclamation point) replaces *File* in an **r** subcommand, the rest of the line is taken as an AIX shell command whose output is to be read. The **r** subcommand does not store the names of AIX commands as default file names.

How to Address Lines and Display Line Addresses with the ed Editor

Prerequisite Tasks or Conditions

1. Start the ed editor.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedures

To Address a Line:

1. Type:

Number	to address the line by its *Number*
.	(period) to address the current line
0	to address the (imaginary) line before the first line
$	(dollar sign) to address the last line
/Pattern/	to address the line indicated by */Pattern/* (slash followed by *Pattern* followed by slash), where *Pattern* is the pattern for a search
'Letter	to address the line indicated by *'Letter* (apostrophe followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark that you used to mark text.

2. Type:

   +*Number* (plus sign followed by *Number*) to address the line a *Number* of lines after the line addressed in step 1

   –*Number* (minus sign followed by *Number*) to address the line a *Number* of lines before the line addressed in step 1

   or skip this step.

To Display the Address of a Line:

1. Type:

   . (period) to display the address of the current line

   $ (dollar sign) to display the address of the last line

   /Pattern/ to display the address of the line indicated by */Pattern/* (slash followed by *Pattern* followed by slash), where *Pattern* is the pattern for a search

   'Letter to display the address of the line indicated by *'Letter* (apostrophe followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark.

2. Type = (equal sign).
3. Press the **Enter** key.

To Address a Group of Lines:

1. Type the address of the first line in the group.
2. Type , (comma).
3. Type the address of the last line in the group.

To Address the First Line through the Last Line

1. Type , (comma).

   This is equivalent to typing **1,$** to address a group of lines.

To Address the Current Line through the Last Line

1. Type ; (semicolon).

   This is equivalent to typing **.,$** to address a group of lines.

How to Position the Current Line with the ed Editor

Prerequisite Tasks or Conditions

1. Start the ed editor.
2. Be in command mode. (If you are not sure, press the **Enter** key, type **.** (period), and press the **Enter** key.)

Procedure

1. To position the current line forward one line, omit this step.

   Type:

   Number to make the line indicated by *Number* the current line

   $ (dollar sign) to make the last line the current line

   /Pattern to make the line indicated by */Pattern* (slash followed by *Pattern*) the current line, where *Pattern* is the pattern for a search

   ?*Pattern* to make the line indicated by **?***Pattern* (question mark followed by *Pattern*) the current line, where *Pattern* is the pattern for a search

'Letter	to make the line indicated by *'Letter* (apostrophe followed by *Letter*) the current line, where *Letter* is the letter (**a** through **z**) for a mark
+Number	(plus sign followed by *Number*) to position the current line forward a *Number* of lines
–	(minus sign) to position the current line backward one line
–Number	(minus sign followed by *Number*) to position the current line backward a *Number* of lines.

2. Press the **Enter** key.

Changing the Prompt String for the ed Editor

P The **P**rompt subcommand turns on or off the ed prompt string * (asterisk). Initially, the **P** subcommand is off.

Entering System Commands from the ed Editor

!AIX–Command

The *!AIX–Command* allows AIX commands to be run from with the ed program. Anything following the ! on an ed subcommand line is interpreted as an AIX command. Within the text of that command string, the ed program replaces the unescaped % (percent sign) with the current file name, if there is one.

When used as the first character of a shell command (after the ! that runs a subshell), the ed program replaces the ! character with the previous AIX command; for example, the command !! repeats the previous AIX command. If the AIX command interpreter (the **sh** command) expands the command string, the ed program echoes the expanded line. The ! subcommand does not change the current line.

How to Run AIX Commands from the ed Editor

Prerequisite Tasks or Conditions

1. Position the current line.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedures

To Run One AIX Command:

1. Type **!** (exclamation point).
2. Type an AIX command.
3. Press the **Enter** key to run the command and display its output and return to command mode.

   The editor displays ! (exclamation point) when the command completes.

   To run the same command again, type **!** (exclamation point) in place of step 2.

To Run Several AIX Commands:

1. Type ! (exclamation point).
2. Type **sh** and press the **Enter** key.

   A prompt appears.
3. Type an AIX command.
4. Press the **Enter** key to run the command and display its output.
5. To run other commands, repeat steps 3 and 4.
6. Press the **Ctrl–D** keys to display ! (exclamation point) and return to command mode.

Ending and Exiting the ed Editor

q The **q** (quit) subcommand exits the ed program. Before ending the program, the **q** subcommand checks to determine whether the buffer has been written to a file since the last time it was changed. If not, the **q** subcommand displays the ? message. (See the **w** subcommand for information about writing text to a file.)

Q The Quit subcommand exits the ed program without checking for changes to the buffer since the last **w** subcommand (compare with the **q** subcommand).

How to Quit the ed Editor

Prerequisite Tasks or Conditions

1. Start the ed editor.
2. Be in command mode. (If you are not sure, press the **Enter** key, type . (period), and press the **Enter** key.)

Procedure

1. Type:

   q to quit if the current file is unchanged or was written after the last change
   Q to quit even though the current file was not written after the last change.
2. Press the **Enter** key to quit the ed editor.

   If **q** was entered and the current file was not written after the last change, a ? (question mark) is displayed. Enter **q** again to quit the ed editor anyway. (The changes to the current file are lost.)

Japanese Language Support in the ed Editor

Several characters can have the same collating value; for instance, the ASCII letter a and the SJIS roman letter a could be collated together. In such a case, the expression [a–a] would match both the ASCII and the SJIS roman letters.

You can mix ASCII and SJIS characters in a range expression, as long as the character preceding the minus sign collates equal to or lower than the character following the sign, the system interprets the range as consisting only of the two end points.

A common use of the range expression is to match a character class. For example, `[0–9]` is used to mean all digits, and `[a–z A–Z]` is used to mean all letters. This form may produce unexpected results when ranges are interpreted according to the current collating sequence.

Instead of the preceding form, use a character class expression within brackets to match characters. The system interprets this type of expression according to the current character class definition. However, you cannot use character class expressions sin range expressions.

Following is the syntax of a character class expression:

`[:`*characterclass*`:]`

that is, a left bracket, followed by a colon, followed by the name of the character class, followed by another colon and a right bracket.

Japanese Language Support supports the following character classes:

`[:upper:]`	ASCII uppercase letters
`[:lower:]`	ASCII lowercase letters
`[:alpha:]`	ASCII uppercase and lowercase letters
`[:digit:]`	ASCII digits
`[:alnum:]`	ASCII alphanumeric characters
`[:xdigit:]`	ASCII hexadecimal digits
`[:punct:]`	ASCII punctuation character (neither a control character nor alphanumeric)
`[:space:]`	ASCII space, tab, carriage return, new–line, vertical tab, or form–feed character
`[:print:]`	ASCII printing character
`[:jalpha:]`	SJIS roman characters
`[:jdigit:]`	SJIS Arabic digits
`[:jxdigit:]`	SJIS hexadecimal digits
`[:jparen:]`	SJIS parentheses characters
`[:jpunct:]`	SJIS punctuation characters
`[:jspace:]`	SJIS space characters
`[:jprint:]`	SJIS printing characters
`[:jkanji:]`	kanji characters
`[:jhira:]`	Full–width hiragana characters
`[:jkana:]`	Half–width and full–width katakana characters

The brackets are part of the character class definition. To match any uppercase ASCII letter or ASCII digit, use the following regular expression:

```
[[:upper:] [:digit:]]
```

Do not use the expression `[A–Z0–9]`.

Editing with the INed Editor

The **INed** editor is a full–screen text editor that allows you to view, enter, and revise (edit) text at any place in the editor window. (The window is the rectangular portion of the editor screen.) You enter text with a keyboard in the same way you type text on a typewriter. As you type, the text displays on the window.

You can edit or revise the text on the window by using the editor command keys, which allow you to manipulate and format text by pressing the appropriate keys instead of typing and entering commands on a command line. Command key names are always indicated by uppercase **BOLDFACE** type. You can use "INed Editor Functions for the Standard Keyboard" or the **keymaps** command to locate the appropriate command keys.

Editing Functions

You can use the command keys to do the following text editing and text processing functions:

- Move the cursor and scroll the window
- Divide a window into two or more windows and:
  - Edit the same file at two or more places
  - Edit two or more files at one time
  - Copy text from one file to another file
- Insert and delete characters, lines, or blocks of text
- Copy or move lines and blocks of text
- Restore deleted lines and blocks of text
- Search for and replace specified text.

You can also use the command keys to:

- Reset margins and tabs, center lines, and enter control characters
- Print files
- Run AIX, **INed**, and filter commands from the editor
- Obtain help information
- Display the history of a structured file.

The Editor Screen

The top of the editor screen shows you the left and right margins and the tab stops. The bottom of the screen gives you the following information:

- Full path name: The sequence of directories from the root directory (/) to the file name of the current file. (The current file is the file in the window.) Each level in the directory is indicated by a / (slash) symbol.
- Insert/overwrite indicator: The mode of the typewriter keyboard. `INSERT` means the new text you type is inserted at the cursor position. `OVERWRITE` means the new text you type replaces the existing text.
- Line number: The line number of the cursor's present position (the current line) counting from the top of the file.
- Total lines: The total number of lines in the file.
- Text indicators: The symbols show that the window does not completely contain all of the text in the file. For example, a > pointing right means there is text to the right of the window.

The File Manager Screen

The File Manager screen is a list of the files in your current directory together with their descriptions. You can use the File Manager screen and the command keys to manipulate files and directories.

Editing Sessions

The **INed** editor stores information (programs, memos, and other text documents) as files. When you are working with a file or directory, it is called an editing session. You start an editing session either by creating a new file or by accessing an existing file or directory.

When you want to type new information such as a letter, memo, or program, you must create a file. The **INed** editor can edit two types of files, text and structured. Text files are files coded in the American Standard Code for Information Interchange (ASCII). Most of the files you create will probably be in pure ASCII text format. Structured files store specialized data and contain a history, which you can use to recover past versions of the file. (Structured files may become quite large after several changes.) You can use commands such as the **ghost** command, **newfile** command, and **rmhist** command to work with structured files.

While you are editing the file, you can save changes without ending the editing session. You can also make a copy of the file you are editing by saving it in a different file or use the Print Menu to print the file or append it to another file. When you end the session, you store the file. You can either store the file with your new changes or store the original file with no changes.

Splitting and Scrolling Windows and Moving the Cursor

You can split the editor window to create additional windows, which can contain either the file you are editing or a different file. This means that you can edit one file while you view another file, view several files on the same screen, or move text from one file window to another. You can also alternate two files in a window, so that you can use the same buffers to restore deleted text, to search and replace, or to pick up and put down text.

Cursor movement keys let you move the cursor within the editor window, without changing any text. You can move the cursor in these directions:

- Horizontally or vertically
- To the beginning of the next line
- To the top left of the editor window
- To the beginning or end of a line.

When you finish editing one window of text, you can move (scroll) the window to view different parts of your file. You can scroll down to view the next window of text or scroll up to view the previous window of text. You can also scroll right and left to view text that is wider than the window.

Manipulating and Formatting Text

By using the appropriate command keys, you can manipulate text in an efficient way. For example, there are several ways to delete existing text. You can delete one character at a time, delete complete lines, delete several lines, or delete the remainder of a line, and you can restore any of the deleted lines or deleted portions of lines. You can also manipulate text by inserting or overwriting characters or by inserting blank lines.

Three basic steps allow you to move or copy portions of text from one area to another:

1. Define (mark) the text.
2. Pick up the marked text (or a copy) and store it in a buffer.
3. Put the text down in the proper area.

You can mark text by complete lines, by boxes, or by text lines. You can then pick up the marked text (or a copy) and store it in the pick–up stack buffer, or you can delete it. You can place (put down) the contents of the pick–up stack buffer in other areas of your text. You can put down a copy of the text and leave the text in the buffer, or you can put down the text and remove it from the buffer. You can also put down multiple copies of the text

When you access a file, the editor's standard format is in effect. For example, the margins are set in column 1 and column 77. You can change the margins, format text within the margins, and center text. You can also enter and delete tabs for an editing session and enter special control characters.

Manipulating Files and Directories

The AIX file system is an arrangement of directories and files. The top of the file system is a directory called the root directory, which is indicated by the / (slash) symbol. One of the directories at the next level is the **u** directory, which (depending on how the system is configured) can contain the home directories of users in the system.

By using the File Manager screen and the appropriate command keys, you can:

- Create files and directories
- Access files and move among directories
- Copy and move files into directories
- Rename files and directories
- Delete and recover files and directories.

Running AIX and Filter Commands

The **INed** editor allows you to run AIX (shell) commands and filter commands from the editor. You can run AIX commands either by putting the file you are editing on hold while the command runs or by running the command in a box. Filters are programs that take standard input, transform it in some way, and write to standard output. For example, you can use filter commands to replace specified text in a paragraph or file or to sort text into alphabetic or arithmetic order.

Using Profiles

The Editor Profile (**editorprf**) is a structured file that you can use to customize the editor by specifying:

- What the New Task Menu contains
- What the Help Menu contains
- Which files the editor should watch for changes
- Which directories the editor should search to locate forms, helpers, messages, and scripts.

You can also create a print profile (**printprf**) to modify your Print Menu and a File Manager profile (**indexprf**) to change your File Manager screen.

Using the History Display

You can use the History Display to access different levels (history) of a structured file to keep track of changes to the file. The History Display lets you see previous versions of a file. By using menus and forms, you can save the current version or re–create an earlier version of a file, or you can remove the history of structured files.

Related Information

The following items are discussed in this chapter:

"INed Editor Functions for the Standard Keyboard" is a table of what the **INed** command keys are on a standard keyboard.

"How to Start an Editing Session with the INed Editor" contains procedures for accessing the File Manager screen, the last file edited, or a specific file.

"How to Control Editor Boxes with the INed Editor" contains procedures for creating an ENTER box, repeating the last argument, and removing editor boxes.

"How to See Help Messages with the INed Editor" contains procedures for seeing the Help Menu or an explanation of an error message, menu, or screen condition.

"How to Control Windows and Refresh the Screen with the INed Editor" contains procedures for creating and removing windows, moving among windows, alternating files in a window, and clearing and redrawing the screen.

"How to Move the Cursor with the INed Editor" contains procedures for moving horizontally or vertically, to the top left of the window, to the beginning or end of lines, or to tab stops.

"How to Scroll a Window with the INed Editor" contains procedures for scrolling up or down, left or right, to the top or bottom of a file, or to a specific line.

"How to Insert Text with the INed Editor" contains procedures for switching between insert and overwrite modes or current and alternate fonts, resetting or displaying fonts, and inserting one or more lines.

"How to Search for and Replace Text with the INed Editor" contains procedures for searching for a character string and replacing (or deleting) text.

"How to Move, Copy, and Delete Marked Text with the INed Editor" contains procedures for marking lines, boxes, or text, picking up or putting down marked text (or a copy), and deleting marked text.

"How to Delete and Restore Text with the INed Editor" contains procedures for deleting and restoring characters and lines.

"How to Format Text with the INed Editor" contains procedures for changing tabs and margins, formatting paragraphs, and centering text.

"How to Manipulate Files and Directories with the INed Editor" contains procedures for creating, accessing, printing, saving, moving, copying, renaming, and deleting files, for creating, moving among, renaming, and deleting directories, and for displaying details and changing permissions of files and directories.

"How to Run AIX Commands and Filter Commands from the INed Editor" contains procedures for running AIX commands in an editor subshell or box and running filter commands for sorting and replacing text.

"How to Use Profiles with the INed Editor" contains procedures for creating an Editor Profile File, changing the New Task Menu, changing the Help Menu, entering files to watch, changing the search paths, and creating a print profile or File Manager profile.

"How to Use the History Display with the INed Editor" contains procedures for accessing the history of a structured file, saving a version of a structured file, and removing the history of structured files.

"How to End an Editing Session with the INed Editor" contains procedures for saving changes and exiting, ignoring changes and exiting, and canceling the editor.

"INed Files" contains reference information about the files used by the **INed** editor.

For reference information about the following commands, see AIX Commands Reference for IBM RISC System/6000.

The **e** command contains reference information (including syntax, description, and flags) for starting the **INed** editor.

The **keymaps** command contains reference information (including syntax and description) for listing which keys correspond to which **INed** functions for all keyboards.

The **fformat** command contains reference information about formatting a text paragraph.

The **fill** command contains reference information about filling arbitrarily broken lines of text.

The **ghost** command contains reference information about reconstructing previous versions of a structured file.

The **history** command contains reference information about displaying the history of a structured file.

The **just** command contains reference information about filling and justifying unevenly indented paragraphs of text.

The **newfile** command contains reference information about converting a text file into a structured file.

The **prtty** command contains reference information about printing to the printer port of a terminal.

The **readfile** command contains reference information about displaying the text of structured files.

The **rmhist** command contains reference information about removing history from structured files.

The **rpl** command contains reference information about replacing all occurrences of a string in a file.

The **tdigest** command contains reference information about converting **.trm** files into **.bin** files.

The **versions** command contains reference information about printing the modification dates of a structured file.

How to Start an Editing Session with the INed Editor

Prerequisite Tasks or Conditions

1. Start AIX.
2. Be at the system prompt on the command line.

Procedures

To Access the File Manager:

1. Type `e .` (e followed by space followed by period) or type `e $HOME` (e followed by space followed by dollar sign followed by `HOME`).

   A lowercase `e` is the command to begin the INed editor, and the `.` (period) starts the File Manager screen at the current directory. (If you enter `e $HOME`, the editor displays the File Manager screen of your home directory.)

2. Press **RETURN**.
3. The editor displays the File Manager screen, which contains all of the files in a directory.

   Press **EXIT** to return to the system prompt.

To Access the Last File Edited:

1. Type `e`.
2. Press **RETURN**.

   Typing `e` and pressing **RETURN** is a quick way to return to the last file you edited. This command is helpful if you accidentally exit your file, because the editor returns to the same location of the cursor when you last stored the file.

To Access a Specific File:

1. Type `e` followed by a space.
2. Type the name of the file you want to edit.

   Remember to type the file name exactly as you created it.
3. Optionally, specify where you want to begin editing.

   If you want to start editing at the beginning of the file, skip this step.

   If you want to begin editing somewhere else in the file, type a space and then type one of the following:

   - The beginning line number.
     For example, to go to line `55` of the file `NYexpenses`, type:

     ```
     e NYexpenses 55
     ```

   - The beginning line number and column number. (A column is a horizontal position in the file. Columns are numbered starting from the leftmost position in the file.)
     For example, to go to the same line of the same file but start in column `25`, type:

     ```
     e NYexpenses 55 25
     ```

   - The beginning line number, the beginning column number, and a search key.
     For example, to go to the file `NYexpenses` and begin editing at the first occurrence of `dinner`, type:

     ```
     e NYexpenses 0 0 dinner
     ```
4. Press **RETURN**.

   The file appears with the cursor located where you specified it should be.

INed Editor Functions for the Standard Keyboard

Command	Keys
BACKSPACE	Backspace
BEGIN LINE	Alt – cursor left
BOX MARK	Alt – B
BREAK	Ctrl – C (see Note)
CANCEL	Scroll Lock
CENTER	Alt – C

DELETE CHAR	Delete
DO	Alt – X
END LINE	Alt – cursor right
ENTER	Action
EXECUTE	Action
EXIT	Alt – D
FONT	Alt – F
FORMAT	F5
GO TO	End
HELP	F1
HOME	Home
INSERT LINE	F6
INSERT MODE	Insert
LAST ARG	Alt – A
LEFT	Alt – L
LINES UP	Page Up
LINES DOWN	Page Down
LOCAL MENU	F4
MARGIN	Alt – M
MENU	F3
NEXT	Alt – F12
NEXT WINDOW	Alt – N
PAGE UP	Alt – Page Up
PAGE DOWN	Alt – Page Down
PICK COPY	F9
PICK UP	F7
PREVIOUS	Alt – F11
PRINT	Alt – Print Screen
PUT COPY	F10
PUT DOWN	F8
QUIT	Ctrl – \ (reverse slash; see Note)
QUOTE	Alt – Q
REFRESH	Alt – Z
REPLACE	Alt – End
RESTORE	Alt – Insert
RETURN	Enter

RIGHT	Alt – R
SAVE	Alt – S
SEARCH UP	Alt – cursor up
SEARCH DOWN	Alt – cursor down
SET TAB	Alt – V
TAB	Tab
TEXT MARK	Alt – T
USE	Alt – U
WINDOW	Alt – W
ZOOM IN	F11
ZOOM OUT	F12
(1)	Alt – F1
(2)	Alt – F2
(3)	Alt – F3
(4)	Alt – F4
(5)	Alt – F5
(6)	Alt – F6
(7)	Alt – F7
(8)	Alt – F8

Note: The keys that cause the BREAK and QUIT functions can be changed. To determine which keys are in effect for your display, enter `stty -a` on the command line. The `intr` character (for example, `intr=^c`) indicates the BREAK key, and the `quit` character (for example, `quit=^\`) indicates the QUIT key.

To display the keyboard mappings for other keyboards, use the **keymaps** command.

How to See Help Messages with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.
2. If an error message, menu, or special condition is on the screen, an explanation is displayed. Otherwise, the Help Menu is displayed.

Procedures

To See the Help Menu:

1. Press the **HELP** command key.

   The Help Menu is a list of topics you can access to obtain more information. For example, you can select the topic `Keyboard Layouts` to display keyboard charts for the editor. Two other topics are `Suggestions for Your MENU` and `Suggestions for Your Print Menu`. They contain examples you can use to change the New Task Menu and create a print profile.

2. Move the cursor to the topic you want to see.
3. Press the **EXECUTE** command key.
4. The editor displays the topic you selected.

   For example, if you select the topic `Alphabetical List of Editor Commands`, the Alphabetical List of Editor Commands Menu displays, and you can do the following:

   - Move the cursor to a subtopic and press the **ZOOM IN** command key to display the subtopic.
   - Press the **USE** or **ZOOM OUT** command key to return to your file.

   If you are displaying a subtopic, you can do the following:

   - Press the **USE** key to return to your file.
   - Press the **ZOOM OUT** key to return to the Alphabetical List of Editors Commands Menu.
   - Press the **NEXT** command key to display the next item on the menu (without returning to it).
   - Press the **PREVIOUS** command key to display the previous item on the menu (without returning to it).
5. Press the **USE** command key to return to your file from a topic or subtopic (or press the **CANCEL** key to remove the Help Menu).

To See an Explanation of an Error Message, Menu, or Screen Condition:

1. Press the **HELP** command key.

   Under normal conditions, the editor displays the Help Menu. However, you can also use the **HELP** key to display additional information such as:

   - Error messages. If a pop–up error message appears on the window and you press the **HELP** key, another pop–up box explains the cause of the error and what to do.
   - Menus. If a menu is on the window, pressing the **HELP** key displays an explanation of the menu item that the cursor is on.
   - Screen conditions. If a special condition exists (for example, if the cursor is out of the editor window), pressing the **HELP** key explains about the cursor and how to continue editing.
2. The editor displays an explanation.
3. Press the **CANCEL** command key to remove the help message.

How to Move the Cursor with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Move Horizontally or Vertically:

1. To move to the right, press the cursor right (arrow) key.

   To move to the left, press the cursor left (arrow) key.

   To move up in the window, press the cursor up (arrow) key.

   To move down in the window, press the cursor down (arrow) key.

The four arrow (cursor movement) keys move the cursor in the direction of the arrow engraved on the key. You can move the cursor past the text window. However, you cannot type in any area of the screen except the text window (the area within the box). If you continue to press the cursor movement key, the cursor moves to the opposite edge of the same line or column.

To Move to the Top Left Corner of the Window:

1. Press the **HOME** command key.

To Move to the Beginning of the Next Line:

1. Press the **RETURN** command key.

   The **RETURN** key action is similar to the action of a carriage return on a typewriter. However, the **RETURN** key does not always place the cursor at the left margin of the next line. Pressing the **RETURN** key places the cursor as follows:

   - If the cursor is on the last line of a window, the editor scrolls the window up one line.
   - If the next line of the file is a text line, the editor places the cursor under the first nonblank character of that line.
   - If the next line of the file is a blank line, the editor places the cursor on the line underneath the first nonblank character of the previous line.

To Move to the Beginning or End of the Current Line:

1. To move to the beginning of the line, press the **BEGIN LINE** command key.

   To move to the end of the line, press the **END LINE** command key.

   Pressing the **BEGIN LINE** key moves the cursor to the first nonblank character on the current line. (If the line is indented, the cursor does not move to the left margin.) If the line is blank, the cursor moves to the left edge of the screen.

   Pressing the **END LINE** key moves the cursor one space to the right of the last character on the current line. (It does not move the cursor to the right margin.)

To Move to the Next or Previous Tab Stop:

1. To move the cursor to the next tab stop, press the **TAB** command key.

   To move the cursor to the previous tab stop, press the **BACKTAB** command key. (On most keyboards, this is done by pressing the Shift and Tab keys.)

   The editor sets tab stops for you. (The `t`'s at the top of the editor screen are tab markers.) You can set or remove tab stops.

   In a tree structured file, pressing the **TAB** key moves the cursor to the next field.

How to Scroll a Window with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Scroll Up or Down One–Third of a Page:

1. To scroll up (toward the beginning of the file), press the **LINES UP** command key.

   To scroll down (toward the end of the file), press the **LINES DOWN** command key.

   An editor page is equal to the number of lines in one window.

To Scroll Up or Down One or More Pages:

1. To scroll up or down one page, skip this step and step 2.

   Press the **ENTER** command key.

2. Type in the ENTER box the number of pages you want to scroll.
3. To scroll up (toward the beginning of the file), press the **PAGE UP** command key.

   To scroll down (toward the end of the file), press the **PAGE DOWN** command key.

   An editor page is equal to the number of lines in one window.

To Scroll a Specific Line to the Top or Bottom of the Window:

1. Move the cursor to the line you want to place at the top or bottom of the window.
2. Press the **ENTER** command key.
3. To make the line the top line in the window, press the **LINES DOWN** command key.

   To make the line the bottom line in the window, press the **LINES UP** command key.

To Scroll Up or Down a Specified Number of Lines:

1. Press the **ENTER** command key.
2. Type in the ENTER box the number of lines that you want to scroll.
3. To scroll up, press the **LINES UP** command key.

   To scroll down, press the **LINES DOWN** command key.

To Scroll to the Top or Bottom of a File:

1. To move to the first line of the file, skip this step.

   To move to the last line of the file, press the **ENTER** command key.

2. Press the **GO TO** command key.
3. The editor scrolls to line 1 or to the last line of the file.

   If the cursor is on any line except the first line of the file when you press the **GO TO** key, the editors scrolls to the first line. If the cursor is on the first line of the file when you press the **GO TO** key, the editor scrolls to the last line of the file.

To Scroll to a Specific Line in a File:

1. Press the **ENTER** command key.
2. Type in the ENTER box the number of the line you want to move to.
3. Press the **GO TO** command key.
4. The editor scrolls the line number you specified to the center of the window.

To Scroll Right or Left One–Third of a Window:

1. To scroll to the right, press the **RIGHT** command key.

   To scroll to the left, press the **LEFT** command key.

   Scrolling right or left is necessary when you create or edit files with long horizontal lines.

To Scroll a Specific Column to the Right or Left of the Window:

1. Move the cursor to a character in the column you want to move to the left or right side of the window.
2. Press the **ENTER** command key.
3. To scroll the cursor column to the rightmost position of the window, press the **LEFT** command key.

   To scroll the cursor column to the leftmost position of the window, press the **RIGHT** command key.

To Scroll Right or Left a Specified Number of columns:

1. Press the **ENTER** command key.
2. Type in the ENTER box the number of columns you want to move.
3. To scroll to the right, press the **RIGHT** command key.

   To scroll to the left, press the **LEFT** command key.

How to Delete and Restore Text with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.
2. Edit an existing file or insert text.

Procedures

To Delete Characters Right of the Cursor:

1. Move the cursor to the left of the characters you want to delete.
2. Hold down the **DELETE CHAR** command key until all characters you want deleted disappear.

   When you press the **DELETE CHAR** key, the cursor does not move. The character at the cursor position disappears. All characters located to the right of the cursor move left one position.

To Delete Characters Left of the Cursor:

1. Move the cursor to the right of the characters you want to delete.
2. Hold down the **BACKSPACE** command key until all characters you want deleted disappear.

   When you press the **BACKSPACE** key, the cursor moves left one space and the character to the left of the cursor disappears. In insert mode, the rest of the line of text moves left to fill the deleted character's position. In overwrite mode, a space fills the deleted character's position, and the rest of the line of text does not move.

To Delete One or More Lines:

1. Move the cursor to the first (or only) line you want to delete.
2. To delete one line, skip this step and step 3.

   To delete more than one line, press the **ENTER** command key.

3. Type in the ENTER box the number of lines you want to delete.
4. Press the **DELETE LINE** command key.

   When you delete a line of text, all of the text on the cursor line disappears, and the next line moves up to the cursor line. If you continue to hold down the **DELETE LINE** key, the editor continues to delete lines.

To Delete from the Cursor to the End of the Line:

1. Move the cursor to the first character that you want to delete.
2. Press the **ENTER** command key.
3. Press the **DELETE CHAR** command key to leave the rest of the line blank.

To Delete to the End of the Line and Join the Next Line:

1. Move the cursor to the first character that you want to delete.
2. Press the **ENTER** command key.
3. Press the **DELETE LINE** command key to delete the rest of the line and move the following line into the deleted space (join lines).

To Restore the Last Deleted Text:

1. Move the cursor to the line directly below where you want the text restored.
2. Press the **RESTORE** command key until you restore all of the deleted lines you want.

   Each time you delete lines of text, the editor stores the deleted text in a stack buffer. (A stack buffer stores text in sequence. The last text stored is the first text out.) When you press the **RESTORE** key, the editor restores the last deleted text at the cursor line. (The cursor line moves down to make room for the restored text.) If you continue to press the **RESTORE** key, the editor restores all of the text you deleted in the current editing session.

   If you delete lines 2, 3, and 4 of the following text:

```
1 uuuuuuuuuuuuuuu
2 The ABC Company
3 1234 North Drive
4 Austin, TX
5 vvvvvvvvvvvvvvv
```

   the text looks line this:

```
1 uuuuuuuuuuuuuuu
5 vvvvvvvvvvvvvvv
```

   and the delete buffer contains:

```
4 Austin, TX
3 1234 North Drive
2 The ABC Company
```

   If the cursor is under a character on line 5 and you press the **RESTORE** key one time, the result is:

```
1 uuuuuuuuuuuuuuu
4 Austin, TX
5 vvvvvvvvvvvvvvv
```

If you press the **RESTORE** key two more times, the result is:

```
1 uuuuuuuuuuuuuuu
2 The ABC Company
3 1234 North Drive
4 Austin, TX
5 vvvvvvvvvvvvvvv
```

To Restore More Than One Copy of the Last Deleted Line:

1. Move the cursor to the line where you want the text restored.
2. Press the **ENTER** command key.
3. Type in the ENTER box the number of copies you want restored.
4. Press the **RESTORE** command key.

How to Move, Copy, and Delete Marked Text with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.
2. Edit an existing file or insert text.

Procedures

To Mark Complete Lines of Text:

1. Place the cursor on either the first or last line you want to mark.
2. Press the **ENTER** command key.
3. Move the cursor up or down the column to mark complete lines of text.
4. Pick up or delete the marked lines of text.

To Mark a Box of Text:

1. Place the cursor at the upper left corner of the area you want to mark.
2. Press the **BOX MARK** command key.
3. Move the cursor one column to the right of the lower right corner of the box area.
4. Pick up or delete the marked box of text.

To Mark Text:

1. Place the cursor under the first letter of the text.
2. Press the **TEXT MARK** command key.
3. Move the cursor one column to the right of the last letter of the text.
4. Pick up or delete the marked text.

Blocks of text can be complete lines of text, boxes of text, or text lines. Mark the text depending on your need. For example, it is usually better to box mark text in a column format and text mark sentences within a paragraph.

When you mark complete lines of text, you move up or down in the same column. (If you do not stay in the same column, the editor marks the text as a box.) When you mark a box of text, you indicate the upper left hand corner and the lower right hand corner of the box.

When you mark text, you indicate the first and last letters of the text. As you mark either a box of text or text, the editor highlights the text from the beginning cursor position to the current cursor position. If you mark complete lines of text, the editor highlights only the column the cursor is in.

Note: You must end the text mark field at one character or space to the right of the text you want to mark. You must end a box mark at one column to the right of the text you want to mark.

When you press either the **BOX MARK** key or the **ENTER** key and a cursor key, a message like `****BOX/LINE****` replaces the full path name at the bottom of the screen. When you press the **TEXT MARK** key, a message like `*******TEXT*******` replaces the full path name.When either of these messages displays, you can use most of the cursor control keys or window scrolling keys. Most of the other command keys and text entering keys do not work. To cancel this command, press the **CANCEL** command key before you pick up or delete the marked text.

Note: Motion keys such as the **PAGE UP**, **PAGE DOWN**, **LEFT**, **RIGHT**, and **RETURN** command keys allow you either to begin to mark text or to continue to mark text.

To Pick Up Text:

1. Mark the text or place the cursor on a line of text.
2. Press the **PICK UP** command key.
3. The marked text disappears from the window and is placed in the pick–up stack buffer.

To Pick Up a Copy of Text:

1. Mark the text or place the cursor on a line of text.
2. Press the **PICK COPY** command key.
3. The marked text remains in the window, and a copy is placed in the pick–up stack buffer.

To Put Down Text:

1. Move the cursor to the top left position of the place where you want to put down the text.
2. To put down the latest text in the pick–up stack buffer, press the **PUT DOWN** command key.

   To put down a copy of the latest text in the pick–up stack buffer, press the **PUT COPY** command key.

   When you press the **PUT DOWN** key, the editor removes the latest text in the pick–up stack buffer and puts it down at the cursor position. The next text in the stack buffer is then available to be put down.

   When you press the **PUT COPY** key, the editor puts down a copy of the text in the pick–up stack buffer but leaves the original copy in the stack buffer. You can press the **PUT COPY** key to put a copy of this text in several places, one copy at a time.

   To put down multiple copies of the latest text in the pick–up stack buffer, press the **ENTER** key, type the number of copies you want inserted at the cursor position, and press the **PUT DOWN** or **PUT COPY** key.

   The following guidelines apply when you are putting down text or a copy of text:

   - The pick–up stack buffer only keeps text for the current editing session. When you exit the editor, the editor deletes all of the stack buffers.
   - The way the editor makes space for the text you move or copy depends on how you marked the text. If you pick up and put down box–marked text, the editor moves the

existing text to the right of the column of text you put down. If you pick up and put down text–marked text, the editor moves the existing text down. If you move text to an existing line, the text after the cursor is added to the end of the text.

To Delete Marked Text:

1. Mark the text you want to delete.
2. Press the **DELETE LINE** command key.

   Deleted text goes to the delete stack buffer. The operation of the delete stack buffer is similar to the operation of the pick–up stack buffer except that you press the **RESTORE** command key to put down deleted text and you cannot put down (restore) a copy of the deleted text.

How to Insert Text with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Switch between Insert and Overwrite Modes:

1. Press the **INSERT MODE** command key.

   Insert mode lets you type new text in the middle of existing text without deleting anything, For example, if the cursor is at the comma in the following text:

   ```
   He who laughs, laughs best.
   ```

   and you type `last`, the new text is placed directly in front of the cursor:

   ```
   He who laughs last, laughs best.
   ```

   Overwrite mode lets you type over the existing text. For example, if the cursor is on the last letter of the word `lasts` in the following text:

   ```
   He who laughs lasts.
   ```

   and you type the words `, laughs best.`, the editor replaces everything you type over with the new text:

   ```
   He who laughs last, laughs best.
   ```

   At the beginning of each editing session, the editor puts your keyboard in insert mode. Begin and end insert mode by pressing the **INSERT MODE** key. The bottom of the editor screen reads either `INSERT` or `OVERWRITE`. Insert and overwrite modes affect all keyboard characters.

To Switch between the Current and Alternate Fonts:

1. Press the **FONT** command key.

   When you begin to edit a file, the font you type with (the current font) is Roman, and the alternate font is continuous underline.

To Reset or Display the Current Font:

1. Press the **ENTER** command key.

2. Type one of the following in the ENTER box (or skip this step to reset the current font to Roman):

`w`	for word underline font
`c`	for continuous underline font
`g`	for graphics font
`r`	for Roman font
`?`	for displaying the current and available fonts.

   Use the following guidelines when you alternate fonts:

   - The font changes only the text that is typed after you change the font. To change the font for existing text, you must first change the font and then retype the text.
   - The Roman font is the default font for the editor. It is not underlined.
   - The word underline font underlines each word. It does not underline spaces and punctuation.
   - The continuous underline font underlines everything you type, including spaces and punctuation.
   - The graphics font allows you to enter graphic characters on the display by typing the following keys:

Key	Graphic Character
`s`	Upper left corner box
`t`	Top side middle
`d`	Upper right corner box
`w`	Left side middle
`q`	Horizontal line
`a`	Vertical bar
`z`	Intersection
`e`	Right side middle
`f`	Lower left corner box
`r`	Bottom side middle
`g`	Lower right corner box.

   Note: Use the **PRINT** command key to print graphic characters. (How the graphic characters print depends on your printer.)

3. Press the **FONT** command key.
4. Type the text for the new font.

To Type a Control Character:

1. Press the **QUOTE** command key.
2. Type the character that corresponds to the desired ASCII control character.

   Control characters are nonprinting characters. Some control characters perform formatting functions in text files. You can use the **QUOTE** key to insert one control character at a particular location in text. For example, an ASCII control character is FF (form feed). To enter a form feed character, press the **QUOTE** key and then type `l`.

Note: To print files formatted with control characters, go to the system prompt and use the AIX **qprt** command. The **PRINT** command key does not recognize the control characters you enter with the **QUOTE** key.

To Insert One or More Lines:

1. Move the cursor to the line directly below where you want one or more new lines to appear.
2. To insert one line, skip this step and step 3.

   Press the **ENTER** command key.
3. Type in the ENTER box the number of lines you want to insert.
4. Press the **INSERT LINE** command key.

   Use the **INSERT LINE** key to insert one or more blank lines in the middle of text. You can use blank lines to enter new text or to indicate a paragraph.

   In the following example, we want a blank line inserted between the end of one paragraph and the beginning of the next paragraph:

```
this beginning will serve as a helpful guide to
our initial system introduction.

     Please feel free to call if you have any
questions or problems.
```

   Move the cursor anywhere on the line that starts `Please feel free`, and press the **INSERT LINE** key. The result is:

```
this beginning will serve as a helpful guide to
our initial system introduction.

     Please feel free to call if you have any
questions or problems.
```

   In the following example, we want to insert two lines for a signature:

```
Sincerely yours,
Lila Beardorf
Vice President, Ecological Affairs
```

   Move the cursor to the line `Lila Beardorf`, press the **ENTER** key, type `2` in the ENTER box, and press the **INSERT LINE** key. The result is:

```
Sincerely yours,
Lila Beardorf


Vice President, Ecological Affairs
```

To Split a Line:

1. Move the cursor under the first character you want to move to the new line.
2. Press the **ENTER** command key.
3. Press the **INSERT LINE** command key.

   When you split lines, part of the text remains on the original line, and the rest of the text moves to a new line. For example, to split the following line of text:

```
Jack and Jill went up the hill To fetch a pail of water.
```

   move the cursor under the `T` (of the word `To`), press the **ENTER** key, and (after the ENTER box appears) press the **INSERT LINE** key. The editor splits the text as follows:

```
Jack and Jill went up the hill
To fetch a pail of water.
```

How to Search for and Replace Text with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.
2. Edit an existing file or insert text.

Procedures

To Specify and Search for a Character String:

1. Press the **ENTER** command key.
2. Type in the ENTER box the character string you want to find.

   Character strings are words or text you specify. When the editor finds the character string, the editor scrolls the window to display the character string. (The cursor will be on the first character of the character string.)
3. To search down the file for the character string, press the **SEARCH DOWN** command key.

   To search up the file for the character string, press the **SEARCH UP** command key.

   Use the following guidelines when you search for text:

   - When the editor does not find any more occurrences of the character string in the direction you are searching, it displays a message. You can press the **CANCEL** command key to remove the message and then press the **SEARCH UP** or **SEARCH DOWN** key to search in the opposite direction.
   - The editor checks for uppercase and lowercase letters. For example, if you type `March`, the editor does not locate `march` or `MARCH`.
   - The editor checks for blanks. For example, if you enter `he` (with a blank space before `he`), the editor locates `here` but not `the`.

   To stop a search before it completes processing, press the **BREAK** command key. The editor displays a message like `Stopped`. If the search completes before you press the **BREAK** key, the operation is not undone.

   Note: Some keyboards have **Break** engraved on the top of a key. This key is not necessarily the **BREAK** command key.
4. Continue to press the **SEARCH DOWN** or **SEARCH UP** key to search for other occurrences of the character string.

To Search for Another Occurrence of a Word:

1. Move the cursor to the first character of the word to be used as the search character string.
2. Press the **ENTER** command key.
3. To move down to the next occurrence of the character string, press the **SEARCH DOWN** command key.

   To move up to the previous occurrence of the character string, press the **SEARCH UP** command key.

   When you search for an existing word, the search character string begins at the character the cursor is on and ends at the character in front of the next blank. For example, if the cursor is on the `r` in the word `bright` and you press the **ENTER** key and

then the **SEARCH DOWN** key, the editor looks for the next occurrence of the character string `right`.

To Replace (or Delete) Text:

1. Use the **SEARCH DOWN** or **SEARCH UP** command key to move the cursor to the text you want to replace.
2. Press the **ENTER** command key.
3. To delete the text, skip this step.

   To replace the text, type the new text in the ENTER box.
4. Press the **REPLACE** command key.
5. Repeat steps 1 through 4 until you replace all of the text.

   For example, to search for `blossom` and replace it with `bloom` in the following text:

   ```
   Blue Sage is one of the most widely distributed
   Salvia species.  In Texas it starts to blossom in
   April and continues to blossom much of the summer.
   ```

   a. Press the **ENTER** key, type `blossom` in the ENTER box, and press the **SEARCH DOWN** key until you locate the first `blossom` you want to replace.

   b. Press the **ENTER** key, type `bloom` in the ENTER box, and press the **REPLACE** key. The word `blossom` is in the search buffer, the word `bloom` is in the replacement buffer, and the text looks like this:

   ```
   Blue Sage is one of the most widely distributed
   Salvia species.  In Texas it starts to bloom in
   April and continues to blossom much of the summer.
   ```

   c. Press the **SEARCH DOWN** key to find the next `blossom` and press the **REPLACE** key. The text looks like this:

   ```
   Blue Sage is one of the most widely distributed
   Salvia species.  In Texas it starts to bloom in
   April and continues to bloom much of the summer.
   ```

   To delete the search text, do not type any text in the ENTER box before you press the **REPLACE** key.

   Note: If you want to replace a text string throughout a file, you can use the **rpl** filter command.

How to Format Text with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.
2. Edit an existing file or insert text.

Procedures

To Set or Remove Tab Stops:

1. Move the cursor to the column where you want to set or remove a tab stop.

   The `t`'s above the editing window mark the position of the current tab stops.

2. To set a tab stop, skip this step.

   To remove a tab stop, press the **ENTER** command key.

3. Press the **SET TAB** command key.

Note: The tab stops you set and remove are for the current editing session only. When you exit the file and then return to that file, the original tab stops are in effect. (When the editor saves a file, it may change multiple blank spaces at the beginning of a line to tab characters.)

To Change the Left or Right Margin:

1. Move the cursor to the column when you want your left or right margin.

   The `l` and `r` above the editing window mark the current positions of the margins.

2. To change the left margin, skip this step.

   To change the right margin, press the **ENTER** command key.

   If you have several lines of text to format or indent, change the left margin. If you want to indent only one line, it is easier to use the space bar or the **TAB** command key. To return the left margin to its original position, move the cursor to the first column in the window and press the **MARGIN** key.

   The editor automatically moves text that does not fit within the margins to the next line. This function is called word wrap. The right margin controls the word wrap function and the format function. If you do not want the word wrap function, change the right margin to the same column as the left margin. You can always return to the word wrap function by moving the cursor and resetting the right margin. You can change the right margin to any column between 1 and 200. If you want lines that are longer than the editor window, press the **RIGHT** command key to move the window beyond the edge of the current window and then change the right margin to the column you want.

3. Press the **MARGIN** command key.

   The new margin settings apply only to:

   - Text you enter after you change the margins.
   - Paragraphs you format with the **FORMAT** command key.
   - Lines you center with the **CENTER** command key.

Note: The margins you change are for the current editing session only. When you exit the editor and then restart the editor and return to that file, the original margins are in effect. (The text you typed with the changed margins does not change.)

To Change Both Margins:

1. Move the cursor to the column where you want your left margin.
2. Press the **ENTER** command key.
3. Move the cursor to the column where you want your right margin.
4. Press the **MARGIN** command key.

To Format a Paragraph:

1. Move the cursor to the first line of the paragraph you want to format.

   A paragraph is any text that is separated from other text with a blank line.

2. If necessary, reset the margins.
3. Press the **FORMAT** command key.

   The text fills in to fit the current margin settings from the cursor position to the end of the paragraph. (The indentation of the first line of the paragraph does not change.) This is convenient if you changed the margins or made changes and now have ragged or broken lines. For example, if your text looks like the following and your right margin is in column 50:

```
     Users' meetings provide a forum for
system operators to learn
new techniques and share common problems with
other people in their company.
```

   you can press the **FORMAT** key to format the text as follows:

```
     Users' meetings provide a forum for system
operators to learn new techniques and share common
problems with other people in their company.
```

   For some text you may get unexpected results when you press the **FORMAT** key. For example, if this is how your text looks:

```
M. Akers    998–9984
J. Brown    967–9987
G. Erwin    998–9982
M. Harper   998–9988
L. Lamar    998–9981
```

   and you press the **FORMAT** key, the result is:

```
  M. Akers    998–9984  J. Brown    967–9987  G.
Erwin    998–9982  M. Harper  998–9988  L. Lamar
998–9981
```

   The editor places the original text in the delete buffer. If you press the **FORMAT** key and do not like the results, you can press the **RESTORE** command key to return the paragraph to its original format. (The original text is put down at the cursor position, and the formatted text is at the end of the restored text.)

   You can also format text with the **fformat** command, **fill** or **ffill** command, and **just** or **fjust** command.

To Center One or More Lines between Margins:

1. Move the cursor to the first (or only) line you want centered.
2. To center one line, skip this step and step 3.

   To center more than one line, press the **ENTER** command key.
3. Type in the ENTER box the number of lines to center.
4. Press the **CENTER** command key.

   If you do not set margins for a document, the editor centers text within the window.

To Center a Block of Text:

1. Place the cursor on the first or last line of the block of text.
2. Press the **ENTER** command key.
3. Move the cursor up or down to mark the lines.
4. Press the **CENTER** command key.

How to Control Windows and Refresh the Screen with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Create a Window:

1. Move the cursor to the new window position.

   Use these guidelines when you create additional windows:

   - Create horizontal windows by placing the cursor anywhere in the window (except the top line) before pressing the **WINDOW** command key.
   - Create vertical windows by placing the cursor on the top line of a window before pressing the **WINDOW** key.
   - Be careful when your exit any file if you are using multiple windows. If you exit any of the windows by entering `q` in the ENTER box, changes made during the current editing session are erased for each text file in each window, and the editor returns you to the operating system prompt.
   - The information above and below the window changes to reflect the status of the window that contains the cursor. For example, the full path name contains the file name in the cursor window.

2. To create a window containing the current file, skip this step and step 3.

   Press the **ENTER** command key.

3. Type the name of a file in the ENTER box.

   The new window will contain the specified file.

4. Press the **WINDOW** command key.

   You can use additional windows to:

   - View and edit two or more files at the same time.
   - View and edit the same file in two or more file locations.
   - Use the same pick–up stack buffers for each of the files. For example, using the block commands, you can move or copy text from one file to another file.
   - Use the delete buffer to delete text from one file and restore it in another file.
   - Use the search buffers and replace buffers to search and replace the same text in more than one file.

To Remove All Windows except the Window Containing the Cursor:

1. Move the cursor to the window you want to keep.
2. Press the **ENTER** command key.
3. Press the **WINDOW** command key.

To Move to Another Window:

1. To move to the next window, skip this step.

   To move to the previous window, press the **ENTER** command key.

   The top and bottom lines of the editor screen display the status of the window with the cursor. This means the file name and the line number of the cursor window are always on the bottom line. "Next window" and "previous window" refer to the sequence in which you created the windows. You can also use the cursor movement keys to move from window to window.

2. Press the **NEXT WINDOW** command key.
3. The cursor moves to the other window.

To Alternate Files in a Window:

1. To exchange the current file with the alternate file, skip this step and step 2.

   To specify a new file in the editor window, press the **ENTER** command key.

   When you specify a new file, the current file becomes the alternate file. You can switch between these two files by pressing the **USE** key. The editor keeps the cursor position of both files. If you specify the current file as the alternate file, you can switch between two places in the same file without losing your cursor position. You can continue to specify more current files, but you can only alternate between the last two files you specify.

   Note: If you try to alternate multiple windows, the editor only alternates the file where the cursor is located. (The window for the other file is gone when you return.)

2. Type the name of a file in the ENTER box.

   The editor window will change to the specified file.

   To edit a file in a different directory, type the full path name in the ENTER box, instead of just the file name.

3. Press the **USE** command key.
4. The editor window changes to the other file.

   You can use the same buffers for each file you alternate in the window. This means that you can:

   - Delete text in one file and restore it in the alternate file.
   - Pick up text or a copy of text in one file and put it down in the alternate file.
   - Search and replace text in one file and then search and replace the same text in the alternate file.

   When you exit the editing session, you can press the **EXIT** command key to save all of the changes in all of the text files, or you can press the **ENTER** command key and type `q` and press the **EXIT** key to save none of the changes in all of the text files.

To Clear and Redraw the Screen:

1. Press the **REFRESH** command key.

   If a system message appears on the screen or if the display becomes garbled for some reason, pressing the **REFRESH** key clears the screen and redraws the display as it was before the message or problem occurred. It does not erase or delete any text in the window.

How to Control Editor Boxes with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Create an ENTER Box to Enter an Argument:

1. Press the **ENTER** command key.

   Note: Some keyboards have **Enter** engraved on the top of a key. This key is not necessarily the **ENTER** command key.

   The editor creates an ENTER box. The **ENTER** command key toggles the ENTER box on and off. (If you press the **ENTER** key more than once, the ENTER box appears and disappears.)

2. Enter an argument.

   Many of the editor commands allow you to type a parameter in the ENTER box. Arguments consist of numbers, letters, or words that expand or change the way the commands work.

   For example, if you press the **INSERT LINE** command key, the editor inserts a blank line above the cursor. If you want to insert 10 blank lines, you can press the **INSERT LINE** key 10 times. Or you can insert 10 blank lines at once by pressing the **ENTER** key, typing `10` in the ENTER box, and pressing the **INSERT LINE** key.

3. Press the appropriate command key.

To Repeat the Last Argument:

1. Press the **LAST ARG** command key.

   Use the **LAST ARG** key to reenter the last argument entered in the ENTER box (instead of pressing the **ENTER** key and retyping the argument). This command is convenient if you press the wrong command key after you type a long argument in the ENTER box.

   You can also use the **LAST ARG** key to reenter the same argument for a different command.

2. The ENTER box with the last argument pops up on the window.

3. Press the appropriate command key.

To Remove ENTER, Menu, and Message Boxes:

1. Press the **CANCEL** command key.

   Use the **CANCEL** key to remove an editor box from the window so you can continue to edit. For example, the box may be a message. (Messages can be caused by pressing the wrong key. You cannot continue to edit until you remove the message.)

There are several kinds of editor boxes:

- The ENTER box, which is created when you press the **ENTER** command key.
- Menu boxes, which contain lists of items from which you can make a selection.
- Message boxes, which contain contain comments, error messages, or instructions from the system.

Information boxes are another type of pop–up box. They contain processing notes, and they disappear when processing completes.

How to Run AIX Commands and Filter Commands from the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Run AIX Commands:

1. Press the **MENU** command key.
2. The editor displays the New Task Menu.

   The New Task Menu lets you run AIX commands without leaving the editor.
3. Move the cursor to one of the following options:

   - `Execute AIX shell commands`
   - `Run a shell command in a box`

   and press the **EXECUTE** command key.

   Note: Always select `Execute AIX shell commands` if:

   - The command requires a response.
   - You want to use the AIX command to make changes in the file you are editing.
   - You want to save the file you are editing before running the AIX command.

   If you select `Execute AIX shell commands`, the editor displays a message like `Saving file` and displays a shell prompt. Enter an AIX command or commands at the shell prompt. Press the **CTRL** and **D** keys to resume editing.

   If you select `Run a shell command in a box`, the editor displays the Shell command box. Type an AIX command in the box and press the **EXECUTE** command key. The editor displays the message `Executing` and, when the command completes, displays the results of the AIX command in a box. (If the command has no output, the editor displays the message `No output`.

   Note: You can also run an AIX command in a box by pressing the **ENTER** command key, typing the AIX command in the ENTER box, and pressing the **MENU** command key. The editor displays a message like `Executing` and displays the results in a box.

To Run a Filter Command:

1. Press the **ENTER** command key.
2. Type the filter command in the ENTER box.

   Filters are commands that change files. For example, the **rpl** command is the INed filter command to replace text in a file. Some other useful AIX filter commands are the **nl** command, **sort** command, and **tr** command. You can also use many other AIX commands as filters.
3. Press the **DO** command key.

   The editor uses the following conventions when it runs a filter command:

   - Only the text from the cursor line to the end of the paragraph is affected unless you specify otherwise. The end of a paragraph is a blank line or a line beginning with a . (period).
   - You can specify text beyond the current paragraph as follows:
     - Type the number of paragraphs to change, type a blank, and then type the filter command.
     - Type the number of lines to change followed by an `l` (lowercase ell), type a blank, and then type the filter command.
   - The latest filter command you enter is effective until you enter a new filter command or exit the editor. To rerun the latest filter command, press the **DO** key. To display (but not run) the latest filter command, press the **ENTER** key and then the **DO** key.
   - Filter commands can contain several parameters. Separate parameters with a blank. If a parameter contains a blank or a special character, use ' (single quotation marks) or " (double quotation marks) to enclose the parameter.
   - If you press the **DO** key and run an AIX command that is not a filter command, the output of the command replaces the text from the cursor to the end of the current paragraph. For example, you can use the AIX **cat** command to replace text with text from another file. (If you want to add the text, but not replace the current paragraph, you can run the **cat** command at a blank line.)
   - If the filter command produces no output, the paragraph is removed.

   To stop a filter command before it completes processing, press the **BREAK** command key. The editor displays a message like `Filter stopped`. If the command completes before you press the **BREAK** key, the operation is not undone.

   Note: Some keyboards have **Break** engraved on the top of a key. This key is not necessarily the **BREAK** command key.

To Perform a Simple Sort or Sort Numerically:

1. Put the cursor on the first line you want to sort.
2. Press the **ENTER** command key.
3. To perform a simple sort, type `sort` in the ENTER box.

   To sort numerically, type `sort -n` in the ENTER box.

   (The **sort** command has other flags you can use to sort lines).
4. Press the **DO** command key.
5. The editor displays the sorted lines in ascending or arithmetic order.

To Replace Text from the Current Line to the End of the Paragraph, by Paragraphs, or by Lines:

1. Press the **ENTER** command key.
2. To replace text from the current line to the end of the paragraph, type in the ENTER box: `rpl` *OldText NewText* (where *OldText* is the text you want to replace and *NewText* is the new text).

   To replace text by paragraphs or by lines, type in the ENTER box: *Number* `rpl` *OldText NewText* (where *Number* is either the number of paragraphs or the number of lines you want to search, *OldText* is the text you want to replace, and *NewText* is the new text).

   Text can consist of either one word or a phrase. For example, you can replace the word `priority` with the phrase `order of importance`. Use the following guidelines when you type text for the **rpl** command:

   - Capitalization is important. For example, typing `mark` does not change the words `Mark` or `MARK`. The **rpl** command contains information on how to search for uppercase or lowercase letters and other pattern matching symbols.
   - Blanks are important. For example, if you replace `the` with `a` (with no blanks before and after `the`), words such as `them` and `other` become `am` and `oar`.
   - Separate single words with a blank. For example, to change `ski` to `sky`, type: `rpl ski sky`
   - Enclose phrases with quotation marks. To change `sky` to `blue sky`, type: `rpl sky "blue sky"`
   - Delete text by entering two double quotation marks for the new text. To delete `blue sky`, type: `rpl "blue sky" ""`
   - Replace text throughout a file by going to the top of the file and using at least the number of lines in the file to search and replace. If the file contains 550 lines, replace `blue sky` with `gray sky` by typing: `551 rpl "blue sky" "gray sky"`
   - Only phrases that are on the same line are changed. If `blue` is the last word on one line and `sky` is the first word on the next line, the phrase `blue sky` is not changed.
3. Press the **DO** command key.

How to Manipulate Files and Directories with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Create a File:

1. Access the File manager screen and press the **RETURN** command key to move the cursor to the beginning of the first blank line of the `File` field.
2. Type a name (*FileName*) for your new file.

   Use the following guidelines when your name a file:

   - Give each file a unique name.
   - Choose names that are easy to remember. For example, the file name `memo_form` may mean more to you than `form_12`.
   - Make file names 1 to 10 characters long. (If necessary, file names may be as long as 255 characters. To enter a file name longer than 14 characters, use the **RIGHT** command key.)

- Use letters, numbers, underscores, and other characters that do not have a special meaning to the operating system. In general you should avoid using: * (asterisk), \ (reverse slash), (space), – (minus sign), & (ampersand), ; (semicolon), ? (question mark), ((left parenthesis),) (right parenthesis), [(left bracket),] (right bracket), { (left brace), } (right brace), > (greater than sign), < (less than sign), ' (quote), " (double quote), ‘ (grave accent), ^ (circumflex accent), | (pipe symbol), and @ (at sign).
- Use capitalization and characters carefully. Each time you want to use a file, you must type the file name exactly as you created it. These would all be unique file names: `chapter1`, `CHAPTER1`, `Chapter1`, and `chapter_1`.

3. Press the **TAB** command key to move the cursor to the `Description` field and type a brief description of the file.

   The description is to help you locate files quickly and easily from the File Manager screen. The editor does not require a description of the file.

4. Press the **ZOOM IN** command key.

   The Create File menu appears. The name of the file you are creating is at the end of the first line of this menu. If you typed the wrong file name or want to change this file request, press the **CANCEL** command key and return to step 2.

5. Press the **RETURN** command key to move the cursor to the type of file you want to create.

   Note: Unless you have a special need for a structured file, select `Create a text file (without history)`. (ASCII files are text files.)

6. Press the **EXECUTE** command key.

   The editor displays an editor window for the new file.

7. Edit the file.

8. Press the **ZOOM OUT** command key to return to the File Manager screen.

Note: To make sure the files you edit are saved, use the **SAVE** command key or end the editing session with the **EXIT** command key.

To Access a File:

1. Access the File manager screen.
2. Move the cursor to the file you want to edit.
3. Press the **ZOOM IN** command key.
4. Edit the file.
5. Press the **ZOOM OUT** command key to return to the File Manager screen.

To Print a File:

1. Be sure your printer is turned on and ready to print.

   Follow the instructions that came with your printer.

2. Access the file you want to print.

   To print with the Print menu, the file must be in the editor window.

3. Press the **PRINT** command key.

   The editor displays a Print menu.

4. Move the cursor to one of the following options on the menu and press the **EXECUTE** command key:

   - `Print on default printer`
     The editor displays a message like `Printing on the default printer` and prints the file on the default printer. This option prints the file name and the page number at the top of each page unless your print profile has been changed.

   - `Print (ask for options)`
     The editor displays a Print options box. Type the print options (flags) you want in the box and press the **ENTER** command key. The editor prints the file on the line printer using the options you specified.

     For example, to print a file that is wider than 80 columns, enter the flag `–cdp` in the Print options box. The `–cdp` flag is the **piobe** command flag to condense the print. (When you finish printing the file, you should turn the condensed print off by entering the `–cocdp` flag.) For information about the other available print options, see the **qprt** command and **piobe** command.

   - `Print to file (overwrite)`
     To print to a file means to copy (write) the contents of the file you are editing to another file. To overwrite a file means that the file you are printing replaces any text in the file you are printing to.

     The editor displays a message like `File for printing`. To write the file into the default file shown in parentheses, press the **EXECUTE** command key. To write the file into a specific file, type the file name or full path name and then press the **EXECUTE** key. If the file does not exist, the editor displays a prompt. Press the **EXECUTE** key to create the file. While it prints the file, the editor displays a message like `Printing to file`.

   - `Print to file (append)`
     To print to a file means to copy (write) the contents of the file you are editing to another file. To append a file means to write a file to the end of a second file without overwriting the second file.

     The editor displays a message like `File for printing`. To write the file into the default file shown in parentheses, press the **EXECUTE** command key. To write the file into a specific file, type the file name or full path name and then press the **EXECUTE** key. (You can print more than one file into the file; each file is added to the end of the previous file.) If the file does not exist, the editor displays a prompt. Press the **EXECUTE** key to create the file. While it prints the file, the editor displays a message like `Printing to file`.

To Save Changes to a File:

1. To save to the file you are editing, skip this step and step 2.

   To save to a different file, press the **ENTER** command key.

2. Type in the ENTER box the name of the file in which you want to store the copy of the file you are editing.

   If you specify a new file name in the ENTER box, the editor creates the new file for you.

   Note: If you copy a file into an existing file, the file overwrites the existing file. Overwrite means the text in the file you save replaces the text that is in the file.

3. Press the **SAVE** command key.

4. The editor saves the changes and does not exit the file you are editing.

Each time you press the **SAVE** key, the changes you made are written to a file. As the changes are written, the editor also makes a copy of the file without the current changes and stores it in a **.bak** file, which is the name of the original file with **.bak** at the end. (If no changes were made to the file, the file is not rewritten, and the **.bak** file remains at the previous level.)

Note: The **.bak** files are only for text files.

While it saves the files, the editor displays pop–up messages like `Saving file` or `Copying from` and `Copying to`.

To Copy a File:

1. Access the File manager screen and move to the directory containing the file you want to copy.
2. Move the cursor to the line of the file you want to copy.
3. Press the **PICK COPY** command key.

   The cursor moves to the next line.
4. To copy the file to a different directory, press the **ENTER** command key, type the path name of the other directory, and press the **USE** command key. (If you want to copy the file to the same directory, skip this step.)
5. Press the **PUT DOWN** command key.

   The editor displays a message.
6. Type a name for the new file and press the **EXECUTE** command key. (If you do not want to copy the file, press the **CANCEL** command key instead.)

   The editor creates the copy of the file you picked up. If you want to change the description of the file, press the **TAB** command key to move to the `Description` field, press the **ENTER** command key and then the **DELETE CHAR** command key, and type the new description.

To Delete a File:

1. Access the File manager screen and move to the directory containing the file you want to delete.
2. Move the cursor to line of the file you want to delete.
3. Press the **DELETE LINE** command key.

   When you delete a file, the editor puts the deleted file in the delete buffer. You can press the **RESTORE** command key to restore the file in your directory.

   Note: Deleted files and text are stored in the delete buffer only for the current editing session. However, even though you exited the editor, you can still recover a deleted file.

To Move from One Directory to Another Directory

1. Access the File manager screen.

   The directory level you are located in is called your current directory. The File Manager screen displays each file in the current directory. The descriptions are typed in by the user (not created by the File Manager).

   Note: If you know the full path name of a directory, you can move to that directory by pressing the **ENTER** command key, typing the full path name in the ENTER box, and pressing the **USE** key.

2. To go to a higher directory level, press the **ZOOM OUT** command key.

   To go to a lower level, move the cursor to the line of the lower level directory and press the **ZOOM IN** command key.

To Move to Your Home Directory:

1. Press the **MENU** command key to display the New Task Menu.
2. Move the cursor to the option `Show home directory` and press the **EXECUTE** command key.

   The directory that ends with your user name is called your home directory ($HOME).
3. The editor returns you to the File Manager screen for your home directory.

To Create a Directory:

1. Access the File manager screen.
2. Press the **INSERT LINE** command key.
3. Type the new directory name in the `File` field.
4. Press the **TAB** command key.
5. Type a description for the new directory in the `Description` field.
6. Press the **ZOOM IN** command key.

   The Create File menu appears.
7. Move the cursor to `Create a directory` and press the **EXECUTE** command key.

   When you create a new directory, the editor displays a blank File Manager screen for the new directory. You can use this screen to create new files and directories. To exit this directory and return to the previous File Manager screen, press the **ZOOM OUT** command key.

To Rename a File or Directory:

1. Access the File manager screen and move to the directory containing the file or directory you want to rename.
2. Move the cursor to the line of the file or directory you want to rename.
3. Press the **INSERT MODE** command key to change from insert mode to overwrite mode.
4. Type the new name over the top of the old name.

   When you change a file or directory name, you can use most of the editing command keys. For example, if the new file name is shorter than the old file name, you can press the **DELETE CHAR** command key to delete the remaining letters of the old file name.

To Move a File to Another Directory:

1. Access the File manager screen and move to the directory containing the file you want to move.
2. Move the cursor to the line of the file you want to move.
3. Press the **PICK UP** command key.
4. Move to the File Manager screen of the directory in which you want to place the file.
5. Move the cursor to a line.
6. Press the **PUT DOWN** command key.

When you move a file to another directory, the file is deleted from the original directory and stored in the directory you specify.

To Copy a File into Another Directory:

1. Access the File manager screen and move to the directory containing the file you want to copy.
2. Move the cursor to the line of the file you want to copy.
3. Press the **PICK COPY** command key.
4. Move to the File Manager screen of the directory in which you want to copy the file.
5. Move the cursor to a line.
6. Press the **PUT DOWN** command key.

   When you copy a file to another directory, the file is left in the original directory, and an exact copy of it is stored in the directory you specify.

To Delete a Directory:

1. Access the File manager screen that contains the directory you want to delete.
2. Move the cursor to the line of the directory you want to delete.
3. Press the **DELETE LINE** command key.

   When you delete a directory, the editor removes the directory and the entire directory structure (all of the files and directories it contains). The editor puts the deleted directory in the delete buffer. You can press the **RESTORE** command key to restore the directory and any files it contained.

   Note: Deleted directories and files are stored in the delete buffer only for the current editing session. However, even though you exited the editor, you can still recover a deleted directory.

To Display a Hidden File or Details of a File or Directory:

1. Access the File manager screen.
2. Press the **LOCAL MENU** command key.

   The Local Menu displays.

   Note: When you become familiar with the Local Menu options, you can select the options without going to the Local Menu. To select an option, press the appropriate Local Menu option key, **(1)** through **(5)**.

3. Move the cursor to one of the following options and press the **EXECUTE** command key:

   - `(1) Display "visible" files`
     Displays the File Manager screen for the current directory. To display or change which files are hidden in your File Manager screens, create a File Manager profile.
   - `(2) Display all files`
     Displays the File Manager screen containing all files in the directory, including hidden files. Hidden files are usually created and used internally by the system; most hidden files begin with a . (period). To display or change which files are hidden in your File Manager screens, create a File Manager profile.
   - `(3) Return to normal directory display`
     Returns to the File Manager screen from the Detailed File Status screen.
   - `(4) Show details about files`

Displays the Detailed File Status screen showing some detailed file information for all the files in the directory.

- `(5) Show more details about this file`
  Displays the Detailed File Status Information screen for a specific file.

To Recover a File or Directory:

1. Access the File manager screen of your home directory.
2. Press the **LOCAL MENU** command key.
3. Move the cursor to option `(2) Display all files` and press the **EXECUTE** command key.
4. Move the cursor to the line containing **.putdir** and press the **ZOOM IN** command key.

   When you delete a file or directory, the editor saves it in the **.putdir** directory. The editor prefixes the files in the **.putdir** directory with a number and a . (period). If you delete many files or directories, you should delete the contents of the **.putdir** directory periodically.
5. Move the cursor to the line containing the file or directory you want to restore and press the **PICK COPY** command key.

   Do not use the **PICK UP** or **DELETE LINE** command key unless you intend to remove the file or directory from **.putdir** and make the file or directory unrecoverable.
6. Press the **ZOOM OUT** command key to return to the File Manager screen and press the **PUT DOWN** command key to restore the file or directory.

To Change File or Directory Permissions:

1. Access the File manager screen containing the file or directory.

   If you are the owner of a file or directory, you can change the protection (permissions) at any time. Before you change permissions, you should understand what the three types of permissions (read, write, and execute) allow you to do. You should also understand the three groups of users (owner, group, and others).
2. Press the **LOCAL MENU** command key.
3. Move the cursor to option `(4) Show details about files` and press the **EXECUTE** command key.

   The Detailed File Status screen displays information for the entire directory. The fields in this screen contain the following information:

   `File Name` The name of the file or directory.
   `T` The type of the entry. An `r` indicates a regular text or structured file, a `d` indicates a directory, a `c` indicates a character device, and a `b` indicates a block device.
   `Owner` The owner of the file or directory.
   `Size` The size of the file or directory in bytes.
   `Modification Date` The date and time the file or directory was last modified.
   `Permissions` 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 either at this level or at the detailed file level.

   To change the permissions at the detailed file level, move to the appropriate directory or file, press the **LOCAL MENU** command key, move the cursor to option `(5) Show more`

`details about this file`, and press the **EXECUTE** command key. The Detailed File Status Information screen displays information for the file.

Move the cursor to the `Permissions` field. You can change the permissions by changing (if necessary) to overwrite mode and typing the new permissions in the appropriate field.

4. To exit the Detailed File Status or Detailed File Status Information screen, do one of the following:
   - Press the **LOCAL MENU** command key, move the cursor to `(3) Return to normal directory display`, and press the **EXECUTE** command key.
   - Press the **ZOOM OUT** command key to return to the previous level.

How to Use Profiles with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Create the Editor Profile File:

1. Press the **MENU** command key to display the New Task Menu.
2. Move the cursor to `Edit your editor profile`.

   Before you can customize the editor, you must create the **editorprf** file.
3. Press the **EXECUTE** command key.

   If they do not exist, the editor creates the directory **$HOME/profiles** and the structured file **editorprf** for you. The editor then displays the Editor Profile File screen, the options on which allow you to:

   - Change the defaults in your New Task Menu.
   - Change your Help Menu.
   - Specify files for the editor to watch.
   - Specify editor search paths.
   - Change your keyboard to a different language group.

   You can do the following when the Editor Profile File screen is displayed:

   - Move the cursor to the option you want to edit and press the **ZOOM IN** command key.
   - Return to the file you were editing by pressing the **USE** command key.
   - Exit the editor by pressing the **EXIT** command key.

To Change the New Task Menu

1. Select `MENU Options` from the Editor Profile File screen.
2. The MENU Options screen is displayed.

   When you change or add options in this screen, you change your New Task Menu. (The New Task Menu is the menu that appears when you press the **MENU** command key.) You make changes by modifying the existing options or adding new options. For example, you could add an option to compile the current program file. Or, if you use a certain AIX command frequently, you can add it as an option.

3. To change an existing menu option, move the cursor to the option.

   To add a menu option, move the cursor to a blank line.

4. Press the **ZOOM IN** command key to display the Details of MENU Option screen.

5. Make the changes or the new entries you want:

   - Type the option as you want it displayed in the New Task Menu in the `Description shown in menu` field.
   - Type one of the following in the `Type` field to specify how you want the option to work:

     `form` Changes to the named form
     `file` Changes to the named file
     `helper` Changes to the new helper
     `screen` Clears the screen and runs the program
     `popbox` Runs a program and displays the results in a pop–up box.

     If you select `screen` or `popbox`, AIX processes the file or program. This means that you can enter | (pipe symbols) or >, >>, < (redirection symbols).

   - Type the command, file name, or helper in the `Name of file or program` field. You can enter shell and editor variables in addition to normal shell variables and AIX file names. Indicate a shell variable by typing a $ (dollar sign), followed by the variable name. Underscore messages that you want to appear as a user prompt. In addition to any other shell variables you set, you can use these names:

     `$ALTFILE` The name of the alternate file
     `$FORM` The name of the current form
     `$FILE` The name of the file you are editing
     `$HOME` The name of your home directory
     `$HELPER` The name of the current helper
     `$SYS` The editor directory where the helpers, forms, and help files are installed.

- Type any character in the appropriate box if you want the function:
  - `Sync and reopen file`. A character in this box causes the editor to save the file, run the menu option, and reopen the file. Use this box for any New Task Menu option that can modify a file you are editing.
  - `Save all files`. A character in this box causes the editor to save all text files and run the program. Use this box for any option that can read a file you are editing.

  Note: If there is not enough room in a field, press the **RIGHT** command key to scroll the field. You can press the **LEFT** command key to scroll back.

6. Press the **SAVE** command key.

   The changes you make are effective the next time you press the **MENU** command key.

7. Do one of the following to exit this screen or the MENU options screen:

   - Press the **ZOOM OUT** command key to return to the previous screen.
   - Press the **USE** command key to return to your editing file.
   - Press the **EXIT** command key to return to the system prompt.

To Change the Help Menu:

1. Select `HELP Options` from the Editor Profile File screen.

2. The HELP Options screen is displayed.

   When you change or add options in this screen, you change your Help Menu. You make changes by modifying the existing options or adding new options. For example, you could add your own help information to the Help Menu or create your own help menu.

3. To change an existing menu option, move the cursor to the option.

   To add a menu option, move the cursor to a blank line.

4. Make the changes or the new entries you want:

   - Type the option as you want it displayed in the Help Menu in the `Description shown in menu` field.
   - Type one of the following (normally, `file`) in the `Type` field to specify how you want the option to work:

`form`	Changes to the named form
`file`	Changes to the named file
`helper`	Changes to the new helper
`screen`	Clears the screen and runs the program
`popbox`	Runs a program and displays the results in a pop–up box.

   - Type the file name (or path name) in the `Name of file or program` field. You can enter shell and editor variables in addition to normal shell variables and AIX file names. Indicate a shell variable by typing a $ (dollar sign), followed by the variable name. In addition to any other shell variables you set, you can use these names:

`$ALTFILE`	The name of the alternate file
`$FORM`	The name of the current form
`$FILE`	The name of the file you are editing
`$HOME`	The name of your home directory
`$HELPER`	The name of the current helper
`$SYS`	The editor directory where the helpers, forms, and help files are installed.

5. Press the **SAVE** command key.

   The changes you make are effective the next time you view the Help Menu.

6. Create a file that corresponds to the file name in this menu and put the help information into the file.

7. Do one of the following to exit this screen:

   - Press the **ZOOM OUT** command key to return to the MENU options screen.
   - Press the **USE** command key to return to your editing file.
   - Press the **EXIT** command key to return to the system prompt.

To Enter Files for the Editor to Watch:

1. Select `Files the Editor Should Watch` from the Editor Profile File screen.
2. The Files the Editor Should Watch screen is displayed.

   The editor can watch files to see if they have changed. It does this by checking the files when you start the editor or refresh the screen. For example, if you have a mail file, the editor can check whether you have any new messages. Use this screen to enter any file that you want the editor to watch and display a message when the file changes.

3. To change an existing menu option, move the cursor to the option.

   To add a menu option, move the cursor to a blank line.

4. Make the changes or the new entries you want:

   - Type a message in the `Message to display when the file changes` field. This is the message you receive when the editor notifies you of the file change.
   - Type the file name (or path name) in the `Name of file` field. You can enter shell variables in this field.

5. Press the **SAVE** command key.

   The changes you make are effective immediately.

6. Do one of the following to exit this screen:

   - Press the **ZOOM OUT** command key to return to the MENU options screen.
   - Press the **USE** command key to return to your editing file.
   - Press the **EXIT** command key to return to the system prompt.

To Change the Editor Search Paths:

1. Select `Editor Search Paths` from the Editor Profile File screen.
2. The Editor Search Paths screen is displayed.

   You can use the Editor Search Paths screen to specify the search paths for the editor to use to find forms, helpers, messages, and scripts.

3. Move the cursor to the appropriate box.
4. Type the names of the directories, one per line, that you want the editor to look in. (Type the directories in the sequence you want the editor to search.)
5. If necessary, put the forms, helpers, messages, or scripts in the directories you specified.
6. Press the **SAVE** command key.

   The changes you make are effective after you exit the editor.

7. Do one of the following to exit this screen:

   - Press the **ZOOM OUT** command key to return to the MENU options screen.
   - Press the **USE** command key to return to your editing file.
   - Press the **EXIT** command key to return to the system prompt.

To Create a Print Profile:

1. Press the **ENTER** command key, type `/usr/lpp/msg/$LANG` in the ENTER box, and press the **USE** command key to access the File Manager screen for the **profiles** directory.
2. Move the cursor to the **printprf** file and press the **PICK COPY** command key.
3. Press the **MENU** command key, move the cursor to the `Show Your Profiles Directory` option, and press the **ENTER** command key. If the $HOME/profiles directory does not exist, the editor creates it. The $HOME/profiles directory is then displayed.
4. Press the **PUT DOWN** command key.
5. Press the **ZOOM IN** command key.
6. The Print Options screen is displayed.

   When you change this screen, you change the options in your Print Menu.

7. To change an existing menu option, move the cursor to the option.

   To add a menu option, move the cursor to a blank line.

8. Press the **ZOOM IN** command key to display the Details of Print Option screen.
9. Make the changes or the new entries you want:

   - Type the option as you want it displayed in the Print Menu in the `Description shown in menu` field.

- Type the print output instructions in the `Command` field. You can enter these types of commands:
  - AIX commands
  - | (pipe symbol)
  - > (redirect and overwrite symbol)
  - >> (redirect and append symbol)

  If you enter a | (pipe symbol) in the first column, the editor pipes the printed output through the specified program. If you enter > or >> (redirect symbols), the editor redirects the output to the specified file. If you enter an AIX command, the editor executes the command. If you leave the `Command` field blank, the Print Helper prompts for an AIX command. If you do not specify a file or program name after the pipe or redirect symbol, the Print Helper prompts for the missing information.

You can enter the following shell and editor variables in addition to normal shell variables and AIX file names:

`$ALTFILE`	The name of the alternate file
`$FORM`	The name of the current form
`$FILE`	The name of the file you are editing
`$HOME`	The name of your home directory
`$HELPER`	The name of the current helper
`$PRTCMD`	The expanded command (used only in the `Description for popbox`)
`$PRTFILE`	The name of the temporary print output file
`$SYS`	The editor directory where the helpers, forms, and help files are installed.

- Type the message you want the editor to display while the print command is executing in the `Description for popbox` field.
- Type any character in the appropriate box if you want the function:
  - `Save all ASCII files`. A character in this box causes the editor to save all of the open text files before printing the file.
  - `Clear screen and run command`. A character in this box causes the editor to clear the screen before executing the print command.
  - `Display all output of command`. A character in this box causes the output of the print command to display in a pop–up box.

10. Press the **SAVE** command key.

    The changes you make are effective the next time you use the Print Menu.

11. Do one of the following to exit this screen:

    - Press the **ZOOM OUT** command key to return to the previous screen.
    - Press the **USE** command key to return to the **profiles** file.
    - Press the **EXIT** command key to return to the system prompt.

To Create a File Manager Profile:

1. Press the **ENTER** command key, type `/usr/lpp/msg/$LANG` in the ENTER box, and press the **USE** command key to access the File Manager screen for the **profiles** directory.
2. Move the cursor to the **indexprf** file and press the **PICK COPY** command key.
3. Press the **MENU** command key, move the cursor to the `Show Your Profiles Directory` option, and press the **ENTER** command key. If the $HOME/profiles directory does not exist, the editor creates it. The $HOME/profiles directory is then displayed.
4. Press the **PUT DOWN** command key.
5. Press the **ZOOM IN** command key.
6. The Directory Helper Options screen is displayed.

   When you change this screen, you change the defaults in your File Manager.
7. To change an existing default, move the cursor to the appropriate field and modify it:
   - The first field lets you specify whether the editor updates (synchronizes) the directory listing if you create a file with the **USE** command key. The default for this is `x` for yes. No character in this field means that you must manually list files created with the **USE** command key. You can manually list these files by selecting the Local Menu option `(1) Display "visible" files`.
   - The second field specifies the directory for storing deleted files and directories.
   - The third field specifies the files to be hidden. The default hidden files are any files that begin with a `.` (period) or end in `.bak`, `.index`, or `.old`. The * (asterisk is a pattern–matching symbol that means any character or characters.
8. Press the **SAVE** command key.
9. Do one of the following to exit this screen:
   - Press the **ZOOM OUT** command key to return to the previous screen.
   - Press the **USE** command key to return to the **profiles** file.
   - Press the **EXIT** command key to return to the system prompt.

How to Use the History Display with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Access the History of File Screen:

1. Press the **MENU** command key.
2. The New Task Menu is displayed.

   You can use this menu to display the history of a structured file without leaving the editor.
3. Move the cursor to `Show history of current file`.

   If this option is not on your menu, you can change the New Task Menu to add it.

4. Press the **EXECUTE** command key.
5. The History of File screen is displayed.

   This screen 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 that were inserted, deleted, or changed. You can use this screen to display any of the history of this file.
6. To exit this screen, do one of the following:
   - Press the **ZOOM OUT** command key to return to an editing session.
   - Press the **EXIT** command key to end the editing session and return to the system prompt

To Access the History of a Structured File:

1. Access the History of File screen for the structured file.
2. Move the cursor to the version of the file you want to display and press the **ZOOM IN** command key.

   The editor displays the version of the file you selected.
3. Press the **LOCAL MENU** command key.

   Note: When you become familiar with the Local Menu options, you can select the options without going to the History Display Options menu. To select an option, press the appropriate Local Menu option key, **(1)** through **(5)**.
4. The History Display Options menu is displayed.
5. Move the cursor to one of the following options and press the **EXECUTE** command key:
   - `(1) show time of this version of the file`. Displays the last modification time of the current version in a pop–up box.
   - `(2) show next time`. Displays the next version of the file.
   - `(3) show previous time`. Displays the previous version of the file.
   - `(4) redisplay history`. Displays the History of File screen.
   - `(5) save current version of file`. Allows you to save this version of the file.
6. To remove the History Display Options menu, press the **CANCEL** command key.

To Save a Version of a Structured File:

1. Access the History of File screen for the structured file.
2. Move the cursor to the version of the file you want to display and press the **ZOOM IN** command key.
3. Press the **LOCAL MENU** command key.
4. Move the cursor to `(5) save current version of file` and press the **EXECUTE** command key.
5. The editor displays a message like `Enter file name`.

   When you save a version of a structured file, you must give the version a new name or make the version you are saving the current version.
6. To save this version of the file and retain the current version, type a new file name and press the **EXECUTE** command key.

   The new file name consists of one version. It has no history until changes are made to the new file. The current version retains all of the history.

Note: If you press the **EXECUTE** command key without typing a new file name, the current version of the file is put into a **.bak** file. The version of the file you are viewing becomes the current version, and all of the history of the file is removed.

7. Press the **ZOOM OUT** command key to return to the History of File screen.

To Remove the History of Structured Files:

1. Press the **MENU** command key.
2. The New Task Menu is displayed.

   You can use this menu to remove **... (dots)** files and remove the history from all of your structured files. Dots files are files the editor creates when you edit text files.

3. Move the cursor to `Housekeep`.
4. Press the **EXECUTE** command key.

   The messages `Executing "Housekeep"` and `No output from "Housekeep"` appear.

   Note: If you create many structured files, this procedure can take a long time to run. You may want to use the **at** command to schedule this procedure for a time when you are not using the system.

   To remove dots files without removing the history, use the **del** command.

   To remove the history from only one file, use the **rmhist** command.

INed Editor ASCII Control Characters

Name	Hex Value	Control Key	Graphic Character
NUL	00	@	
SOH	01	A	Vertical bar
STX	02	B	
ETX	03	C	
EOT	04	D	Upper right corner box
ENQ	05	E	Right side middle
ACK	06	F	Lower left corner box
BEL	07	G	Lower right corner box
BS	08	H	
HT	09	I	
LF	0A	J	
VT	0B	K	
FF	0C	L	
CR	0D	M	
SO	0E	N	
SI	0F	O	
DLE	10	P	
DC1	11	Q	Horizontal line

Name	Hex Value	Control Key	Graphic Character
DC2	12	R	Bottom side middle
DC3	13	S	Upper left corner box
DC4	14	T	Top side middle
NAK	15	U	
SYN	16	V	
ETB	17	W	Left side middle
CAN	18	X	
EM	19	Y	
SUB	1A	Z	Intersection
ESC	1B	[	
FS	1C	\	
RS	1E	^	
US	1F	_	

How to End an Editing Session with the INed Editor

Prerequisite Tasks or Conditions

1. Start the INed editor.

Procedures

To Save File Changes, Store Updated Files, and Exit:

1. Press the **EXIT** command key.

   Usually when you create or revise files, you want to store the latest versions.

2. The editor stores the files and returns to the system prompt.

   After the files are stored, the editing session ends, and the previous screen appears. (If you entered the editor from the system prompt, the system prompt displays.)

   The next time you use the files, you will see the updated versions, exactly as you stored them.

   If no changes were made to the files, the files are not stored.

   Note: To save space when the editor stores a text file, it may replace multiple blanks at the beginning of lines with ASCII tab characters. Some programs cannot process files that contain tab characters. Before you use these programs with INed text files, you should run the **untab** command.

To Ignore Editing Changes, Restore Original Files, and Exit:

1. Press the **ENTER** command key.

   Sometimes when you create or revise a file you may want to start over and not store anything you just typed.

Warning: When you use this command, the editor does not save any of the changes you made to text files during this editing session. However, any changes already saved with the **SAVE** command key are retained.

2. Type q in the ENTER box.
3. Press the **EXIT** command key.
4. The editor does not save the editing changes and returns to the system prompt.

   The next time you use the files, you will see the original versions exactly as they appeared when you began your last editing session or last saved with the **SAVE** command key.

   If you are editing structured files, the changes are saved.

To Cancel the Editor if No Other Method Works:

1. Press the **QUIT** command key.
2. The editor attempts to save all files and returns to the system prompt.

INed Files

Purpose

These are the files used by the INed program.

Description

The directory **/usr/lib/INed** contains a number of files and subdirectories used internally by the INed program. The directory **/usr/lpp/msg/$LANG** contains files of translatable text. This directory also contains other files that are not used by INed.

In the following file names, **$LANG** is the value of the **LANG** environment variable, which indicates the natural language currently being used.

bin	A directory containing programs called by the editor to perform various functions. Do not run these programs from the command line.
FATAL.LOG	A log of the error messages the editor records when it encounters a system problem.
helpers	A directory containing programs called by the editor 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 on the INed local menus.
forms	A directory containing forms used by the INed program. Files ending in **.x** or named **x** use the form **x.ofm**. The forms are binary files used directly by the editor in generating displays for structured files.
help/keys.map	A file that is displayed when the **HELP** command key is pressed and the keymap option is chosen.
termcap	A directory containing the files used by the editor to read input from the terminals and write output to the terminals. The **def.trm** file is the readable structured file, and the **terms.bin** file is the compressed version.
/usr/lpp/msg/$LANG	A directory containing help message files and other files containing translated text used by INed. This directory also contains other files that are not used by INed.

Implementation Specifics

These files are part of INed Editor Facilities.

These files are not available for Japanese Language Support.

File

/usr/lib/INed	The directory of files and subdirectories used by the INed program.
/usr/lpp/msg/$LANG	Contains files of translatable text.

Editing with the vi Editor

The **vi** editor is the full–screen, full–function member of a family of editors that also includes:

- The **vedit** editor, a version of the **vi** editor for novices
- The **view** editor, a version of the **vi** editor for reading only
- The **ex** editor, a full–function line editor
- The **edit** editor, a version of the **ex** editor for novices.

To edit a new or existing file, you enter the **vi** command at the prompt on the command line. The file is placed into the edit buffer and displayed on your screen as lines of text. The cursor indicates your current line and character position within the file. You move the cursor or scroll the screen to change position in the file.

Modes

The **vi** editor has three modes of operation:

- Command mode, in which keystrokes are interpreted as **vi** subcommands to be carried out immediately without being displayed
- Text input mode, in which keystrokes are interpreted as text to be displayed as it is added to the file
- Last line mode, in which keystrokes are interpreted as line editor subcommands to be carried out when entered at the bottom of the screen.

You start editing in command mode. The **a**, **A**, **i**, **I**, **o**, **O**, **c**, **C**, **s**, **S**, and **R** subcommands switch you to text input mode, and the **Esc** key switches you back. Subcommands that start with **:** (colon), **/** (slash), **?** (question mark), and **!** (exclamation point) switch you to last line mode, and the **Enter** key switches you back.

Subcommands

A **vi** subcommand allows you to tell the editor how often to perform what operation on how many of which objects.

With one or more subcommands, you can:

- Position the cursor or control the screen
- Search or mark text
- Add, change, or delete text
- Undo or repeat changes or deletions
- Move or copy text
- Run AIX commands or line editor subcommands
- Manipulate files
- Quit the **vi** editor

Subcommands operate on the following objects:

- Characters, character strings, words, lines, sentences, paragraphs, and sections
- Text between the cursor and the match to a literal string or regular expression from a search
- Text between the cursor and the character or line with a mark (**a** through **z**)
- Text in a numbered (**1** through **9**) or named (**a** through **z**) buffer.

Options and Macros

You can tailor the **vi** editor somewhat by:

- Setting options, such as whether line numbers are displayed or which shell is used to run AIX commands
- Defining macros, such as mapping a key to a sequence of subcommands or abbreviating a character string.

You can set options and define macros within the **vi** editor each time you use it, or you can set and define them in a file that is run every time you use the editor.

Related Information

The following items are discussed in this chapter:

"How to Set Editor Options with the vi Editor" contains procedures for listing and changing the settings of options.

"How to Define and Use Macros with the vi Editor" contains procedures for defining and using buffer macros, mapping and unmapping keys, and defining and removing abbreviations.

"How to Start the vi Editor" contains a procedure for starting the editor.

"How to Position the Cursor with the vi Editor" contains procedures for moving the cursor left or right, up or down, forward or backward, and to a position on the screen or within the file.

"How to Control the Screen with the vi Editor" contains procedures for scrolling the screen, positioning a line on the screen, and cleaning up the screen.

"How to Add Text with the vi Editor" contains a procedure for appending or inserting text.

"How to Search Text with the vi Editor" contains a procedure for searching text for a literal string or regular expression.

"How to Mark Text with the vi Editor" contains procedures for marking a character or line and for moving or referring to a mark.

"How to Change Text with the vi Editor" contains procedures for changing characters, strings, lines, and other text objects and for breaking, joining, and shifting lines.

"How to Delete Text with the vi Editor" contains procedures for deleting characters and other text objects.

"How to Undo or Repeat Changes or Deletions with the vi Editor" contains procedures for undoing, retrieving, or repeating changes or deletions.

"How to Move or Copy Text with the vi Editor" contains procedures for moving or copying lines and other text objects.

"How to Run AIX Commands from the vi Editor" contains procedures for running one or more AIX commands, adding output from a command, and inputting to a command and replacing with output from the command.

"How to Run Line Editor Subcommands with the vi Editor" contains a procedure for running **ex** subcommands.

"How to Manipulate Files with the vi Editor" contains procedures for reading, writing, and editing files.

"How to Quit the vi Editor" contains procedures for writing and quitting and for quitting with or without writing.

For reference information supporting the following commands, see AIX Commands Reference for IBM RISC System/6000.

The **vi** command contains reference information (including syntax, description, flags, and subcommands) for the **vi** editor.

The **vedit** command contains reference information for the **vedit** editor.

The **view** command contains reference information for the **view** editor.

The **ex** command contains reference information for the **ex** editor.

The **edit** command contains reference information for the **edit** editor.

How to Start the vi Editor

Prerequisite Tasks or Conditions

1. Start AIX.
2. Be at the prompt on the command line.

Procedure

1. Type vi.
2. To start with the cursor on the first line of the file, omit this step and step 3.

   Press the spacebar and type + (plus).
3. Type:

Number	to start with the cursor on the line indicated by *Number*
/Pattern	to start with the cursor on the line indicated by */Pattern* (slash followed by *Pattern*), where *Pattern* is the pattern for a search

   or skip this step to start with the cursor on the last line.
4. To edit a new file with no name, omit this step and step 5.

   Press the spacebar and type the name of an existing or new file to edit.
5. To edit a list of files, repeat step 4 for each file in the list (or skip this step to edit one file).
6. Press the **Enter** key to start editing with the vi editor (in command mode).

How to Position the Cursor with the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Move Left or Right (within a Line) or Up or Down (within a Column):

1. Type the number (for example, 5) of positions to move (or skip this step to move one position).

2. Type:

h or the cursor left key or the **Backspace** key	to move left
l or the cursor right key or the spacebar	to move right
k or the cursor up key	to move up
j or the cursor down key	to move down.

To Move Left or Right (within a Line) to a Particular Column:

1. Type the number (for example, `10`) of the column (or skip this step to move to the first or last column or the first nonblank character).
2. Type:

|	(logical OR) to move to the column indicated in step 1
0	to move to the first column
$	(dollar sign) to move to the last column
^	(circumflex accent) to move to the first nonblank character.

To Move Left or Right (within a Line) to a Particular Character:

1. Type the number (for example, `2`) of occurrences to move to (or skip this step to move to the first occurrence).
2. Type:

f	to move right to the character
F	to move left to the character
t	to move right to the position before the character
T	to move left to the position after the character.

3. Type the character (for example, `w`).

   To repeat the move, type:

;	(semicolon) to move in the same direction
,	(colon) to move in the opposite direction.

To Move Forward or Backward by Line:

1. Type the number (for example, `10`) of lines to move (or skip this step to move one line).
2. Press the **Enter** key to move forward or type:

+	(plus sign) to move forward
–	(minus sign) to move backward.

To Move Forward or Backward by Word, Sentence, Paragraph, or Section:

1. Type the number (for example, `5`) of units to move (or skip this step to move one unit).
2. Type:

w	to move forward by words (with punctuation marks separate) to the beginning of a word
W	to move forward by words (with punctuation marks included) to the beginning of a word
e	to move forward by words (with punctuation marks separate) to the end of a word
E	to move forward by words (with punctuation marks included) to the end of a word
)	(right parenthesis) to move forward by sentences (delimited by two blank characters) to the beginning of a sentence
}	(right brace) to move forward by paragraphs (delimited by blank lines) to the beginning of a paragraph

]] (two right brackets) to move forward by sections to the beginning of a section

b to move backward by words (with punctuation marks separate) to the beginning of a word

B to move backward by words (with punctuation marks included) to the beginning of a word

((left parenthesis) to move backward by sentences (delimited by two blank characters) to the beginning of a sentence

{ (left brace) to move backward by paragraphs (delimited by blank lines) to the beginning of a paragraph

[[(two left brackets) to move backward by sections to the beginning of a section.

To Move to the Top, Middle, or Bottom of the Screen:

1. Type the number (for example, 2) of lines below the top or above the bottom to move to (or skip this step to move to the top, middle, or bottom line).
2. Type:

H to move to the top of the screen

M to move to the middle of the screen

L to move to the bottom of the screen.

To Move to a Particular Line:

1. To find out the number of the current line, press the **Ctrl–G** keys.
2. Type the number (for example, `150`) of the line to move to (or skip this step to move to the last line).
3. Type **G**.

---

vi Editor Options

vi Option, abbreviation	Description
autoindent, ai	Indents automatically in text input mode to the indention on the previous line by using the spacing between tab stops specified by the **shiftwidth** option. The default is **noai**. To back the cursor up to the previous tab stop, press the **Ctrl–D** keys. This option is not in effect for global commands.
autoprint, ap	Prints the current line after any command that changes the editing buffer. The default is **ap**. This option applies only to the last command in a sequence of commands on a single line and is not in effect for global commands.
autowrite, aw	Writes the editing buffer to the file automatically before the **:n** subcommand, **:ta** subcommand, **Ctrl–A** subcommand, and the **!** subcommand if the editing buffer changed since the last **write** subcommand. The default is **noaw**.
beautifying text, bf	Prevents the user from entering control characters (except for tab, new–line, and scans form feed) in the editing buffer during text entry. The default is **nobf**. This option applies to command input.
closepunct, cp=	Handles a list of closing punctuation, especially when wrapping text (see the **wraptype** option). Precedes multi–character punctuation with the number of characters, for example: `cp=3..;)}`. The **vi** command does not split

	closing punctuation when wrapping. To set the default value, run kanji–compatible vi and enter **:set cp**.
directory, dir=	Displays the directory that contains the editing buffer. The default is **dir = /tmp**.
edcompatible, ed	Retains global (**g**) and confirm (**c**) subcommand suffixes during multiple substitutions and causes the read (**r**) suffix to work like the **r** subcommand. The default is **noed**.
hardtabs, ht=	Tells the vi editor the distance between the hardware tab stops on your display screen. The default is **ht=8**.
ignorecase, ic	Ignores distinction between uppercase and lowercase while searching for regular expressions. The default is **noic**.
lisp, lisp	Removes the special meaning of **( )**, **{ }**, **[[**, and **]]** and enables the **=** (formatted print) operator for s–expressions, so you can edit LISP programs. The default is **nolisp**.
list, list	Displays text with tabs and the end of lines marked. Tabs are displayed as `^I` and the end of lines as `$`. The default is **nolist**.
magic, magic	Treats the characters **.** (period), **[**, and ***** as special characters when searching for a pattern. In off mode, only the **( )** and **$** retain special meanings. However, you can evoke special meaning in other characters by preceding them with a \ (reverse slash). The default is **magic**.
modeline, modeline	Runs an editor command line if found in the first five or the last five lines of the file. An editor command line can be anywhere in a line. For the editor to recognize a command line, the line must contain a space or a tab followed by the string `ex:` or `vi:`. The command line is ended by a second **:** (colon). The editor tries to interpret any data between the first and second colon as editor commands. The default is **nomodeline**.
number, nu	Displays lines prefixed with their line numbers. The default is **nonu**.
optimize, opt	Speeds up the operation of terminals that lack cursor addressing. The default is **noopt**.
paragraphs, para=	Defines vi macro names that start paragraphs. The default is **para=IPLPPPQPP LIppIpipnpbp**. Single–letter **nroff** macros, such as the **.P** macro, must include the space as a quoted character if respecifying a paragraph.
partialchar, pc=	Appears in the last display column where a double–wide character would not be displayed completely. The default character is **–** (dash).
readonly, ro	Sets permanent read–only mode. The default is **noreadonly**.
redraw, redraw	Simulates a smart workstation on a dumb workstation. The default is **nore**.
report, re=	Sets the number of times you can repeat a command before a message is displayed. For subcommands that produce many messages, such as global subcommands, the messages are displayed when the command sequence completes. The default is **report=5**.
scroll, scr=	Sets the number of lines to be scrolled when the user scrolls up or down. The default is 1/2 window size rounded down.
sections, sect=	Defines vi macro names that start sections. The default is **sect=NHSHHH HUuhsh+c**. Single–letter **nroff** macros, such as the **.P** macro, must include the space as a quoted character if respecifying a paragraph.

shell, sh=	Defines the shell for the ! subcommand or the :! subcommand. The default is the login shell.
shiftwidth, sw=	Sets the distance for the software tab stops used by the **autoindent** option, the shift commands (> and <), and the text input commands (**Ctrl–D** and **Ctrl–T**). The default is **sw=8**.
showmatch, sm	Shows the matching open parenthesis (or open bracket { as you type the close parenthesis) or close bracket }. The default is **nosm**.
showmode, smd	Displays a message to indicate when the editor is in input mode. The default is **nosmd**.
slowopen, slow	Postpones updating the display screen during inserts. The default is **noslow**.
tabstops, ts=	Sets the distance between tab stops in a displayed file. The default is **ts=8**.
term, term=	Sets the type of work station you are using. The default is **term=$TERM** where **$TERM** is the value of the **TERM** shell variable.
terse, terse	Allows the vi editor to display the short form of messages. The default is **noterse**.
timeout, to	Sets a time limit of two seconds on entry of characters. This limit allows the characters in a macro to be entered and processed as separate characters when **timeout** is set. To resume use of the macro, set **notimeout**. The default is **to**.
warn, warn	Displays a warning message before the ! subcommand executes a shell command if it is the first time you issued a shell command after changes were made in the editing buffer but not written to a file. The default is **warn**.
window, wi=	Sets the number of lines displayed in one window of text. The default is dependent on the baud rate at which you are operating: 600 baud or less, 8 lines; 1200 baud, 16 lines; higher speeds, full screen minus 1 line.
wrapmargin, wm=	Sets the margin for automatic word wrapping from one line to the next. The default is **wm=0**. A value of 0 turns off word wrapping.
wrapscan, ws	Allows string searches to wrap from the end of the editing buffer to the beginning. The default is **ws**.
wraptype, wt=	Determines how a line splits for word wrapping. The default is **wt=general**. The default causes the vi editor to insert a new line after a white space. Any characters following the insert point begin the new line. White space immediately before the inserted line is deleted. When Japanese Language Support is installed and **wraptype=general**, the vi editor inserts a new line to wrap one character. One character forming closing punctuation is allowed past the right margin.
writeany, wa	Turns off the checks usually made before a **write** subcommand. The default is **nowa**.

How to Set Editor Options with the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To List the Settings of Options:

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **set**.
3. Press the spacebar and type **all** to list the settings of all options (or skip this step to list only the nondefault settings).
4. Press the **Enter** key to list the settings.
5. Press the **Enter** key to return (from last line mode) to command mode.

To Change the Setting of an Option:

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **set** and press the spacebar.
3. Type:

Option	to set an option on, where *Option* is the name of the option
no*Option*	to set an option off, where *Option* is the name of the option
Option=*Value*	to change the value of an option, where *Option* is the name of the option and *Value* is the new value.

4. Press the **Enter** key to change the setting and return (from last line mode) to command mode.

Setting vi Editor Options

The vi editor allows you to customize vi options so that you can use the editor for a specific task. You can set options temporarily so they are in effect for only the current editing session, or you can set options permanently so they are automatically in effect for every editing session.

Setting vi Options Temporarily

Use the **:set** subcommand from the command line to set or change an option temporarily. You set some options to a string or a number value; other options you simply turn on or off. To change an option set to a value, enter a subcommand of the form **:set** *Option=Value.* To toggle an option set to on or off, enter a subcommand of the form **:set** *Option* to set it on or **:set no***Option* to set it off. To view the current setting of all the options, enter `:set all`. You can also abbreviate options with the **:set** subcommand. See vi options for a listing of the options and their abbreviations.

Setting vi Options Permanently

You can set vi options permanently by setting the **EXINIT** environment variable in the **.profile** file or by putting the **set** subcommand into a **.exrc** file. Each time you start the editor, it reads the **.profile** file first and then the **.exrc** file.

You set some options to a string or a number value; other options you simply turn on or off. To change an option by setting the **EXINIT** environment variable, see the description of environment variables. To change an option set to a value in a **.exrc** file, put a subcommand of the form **set** *Option=Value* into the **.exrc** file. To toggle an option that can be set to on or off, put in a subcommand of the form **set** *Option* to set it on or **set no***Option* to set it off. You can abbreviate options. See vi options for a listing of the options and their abbreviations.

Here is an example of a **.exrc** file:

```
set ai aw
set wm=5
```

To view the current setting of all the options, enter `:set all` while in the **vi** command mode.

vi General Subcommand Syntax

[*Named_Buffer***] [***Operator***] [***Number***]** ***Object***

Note: Surrounding square brackets indicate optional items.

[*Named_Buffer***]** Specifies a temporary text storage area.

[*Operator***]** Specifies the subcommand or action; instructs the vi editor.

[*Number***]** Is a whole decimal value that specifies either the extent of the action or a line address. The vi editor interprets this number in one of the following ways:

- Go to line *Number*:

  ```
  5G
  10Z
  ```

- Go to column *Number*:

  ```
  25|
  ```

- Scroll *Number* of lines:

  ```
  10Ctrl–D
  10Ctrl–U
  ```

Object Specifies what to act on. This can be a text object (a character, word, sentence, paragraph, section, character string) or a text position (a line, position in the current line, screen position).

Editing a File with the vi Editor

The subcommands for editing allow you to perform the following tasks:

Adding Text to a File

Changing Text While in Input Mode

Changing Text from Command Mode

Copying and Moving Text

Restoring and Repeating Changes

Saving Changes to a File.

vi Subcommands for Editing a File

Adding Text to a File—Text Input Mode

The following subcommands are entered in command mode and bring the vi editor into text input mode to allow you to add text to your file. If you need information about the format of vi subcommands, see general subcommand syntax.

a*Text*	Inserts *Text* after the cursor. End text input mode by pressing the **Esc** key.
A*Text*	Adds *Text* to the end of the line. End text input mode by pressing the **Esc** key.
i*Text*	Inserts *Text* before the cursor. End text input mode by pressing the **Esc** key.
I*Text*	Inserts *Text* before the first nonblank character in the line. End text input mode by pressing the **Esc** key.
o	Adds an empty line below the current line. End text input mode by pressing the **Esc** key.
O	Adds an empty line above the current line. End text input mode by pressing the **Esc** key.

Changing Text While in Input Mode

Use the following subcommands only while in text input mode. They have different meanings in command mode. If you need information about the format of vi subcommands, see general subcommand syntax.

Ctrl–H	Erases the last character.
Ctrl–W	Erases the last small word.
\	Quotes the erase and kill characters.
Esc	Ends insertion and returns to command state.
Ctrl–?	Interrupts and ends insert or **Ctrl–D**.
Ctrl–D	Goes back to previous **autoindent** stop.
^ Ctrl–D	Ends **autoindent** for this line only.
0Ctrl–D	Moves cursor back to left margin.
Ctrl–V	Enters a nonprinting character.

Changing Text from Command Mode

Use the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

C	Changes the rest of the line (same as **c$**).
cc	Changes a line.

cw	Changes a word.
cw*Text*	Changes a word to *Text*.
D	Deletes the rest of the line (same as **d$**).
dd	Deletes a line.
dw	Deletes a word.
J	Joins lines.
r*x*	Replaces the current character with character specified by *x*.
R*Text*	Overwrites characters with *Text*.
s	Substitutes characters (same as **cl**).
S	Substitutes lines (same as **cc**).
u	Undoes the previous change.
x	Deletes a character at the cursor.
X	Deletes a character before the cursor (same as **dh**).
<<	Shifts one line to the left.
<L	Shifts all lines from the cursor to the end of the screen to the left.
>>	Shifts one line to the right.
>L	Shifts all lines from the cursor to the end of the screen to the right.
~	Changes letter at the cursor to the opposite case.
!	Indents for LISP.

Copying and Moving Text

Use the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

p	Puts back text from the undo buffer after the cursor.
P	Puts back text from the undo buffer before the cursor.
"*x***p**	Puts back text from the buffer *x*.
"*x***d**	Deletes text into the buffer *x*.
y	Places the object that follows (for example, **w** for word) into the undo buffer.
"*x***y**	Places the object that follows into the *x* buffer, where *x* is any letter.
Y	Places the line in the undo buffer.

Restoring and Repeating Changes

Use the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

u	Undo the last change.
U	Restores the current line if the cursor has not left the line since the last change.

.	Repeats the last change or increments the "*n***p** command. **Note:** This subcommand is not meant for use with a macro. Enter @@ to repeat a macro.
"*n* **p**	Retrieves the *n*th last delete of a complete line or block of lines.

Saving Changes to a File

Use the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

:w	Writes the edit buffer contents to the original file. If you are using this subcommand within the ex editor, you do not need to type the : (colon).
:w *File*	Writes the edit buffer contents to the named *File*. If you are using this subcommand within the ex editor, you do not need to type the : (colon).
:w! *File*	Overwrites *File* with the edit buffer contents. If you are using this subcommand within the ex editor, you do not need to type the : (colon).

How to Add Text with the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Position the cursor.
3. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedure

1. Type the number of times (for example, 5) to add the text on the current line (or skip this step to add the text once or to add the text on a new line).
2. Type:

a	to append after the cursor
A	to append at the end of the line
o	to open a new line after the current line
i	to insert before the cursor
I	to insert at the beginning of the line
O	to open a new line before the current line.

3. Type the text.

   To make corrections while typing, press:

The **Ctrl–H** keys or the **Backspace** key	to correct the last character
The **Ctrl–W** key	to correct the last word.

4. Press the **Esc** key to return (from text input mode) to command mode.

How to Change Text with the vi Editor

Prerequisite Tasks or Conditions

1. Edit an existing file or add text.
2. Mark text (if a mark is needed for the deletion).
3. Position the cursor.
4. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Change Characters, Strings, or Lines:

1. Type the number (for example, 5) of characters, strings, or lines to change (or skip this step to change one character, string, or line).
2. Type:

r	to replace by character
R	to replace by character or string
s	to substitute by character or string
S	to substitute by line.

3. Type the changed text.

   To make corrections while typing, press:

The **Ctrl–H** keys or the **Backspace** key	to correct the last character
The **Ctrl–W** keys	to correct the last word.

4. Press the **Esc** key to return (from text input mode) to command mode (or skip this step if you typed **r** in step 2).

To Change Text Objects:

1. Type the number (for example, 5) of text objects to change (or skip this step to change one text object).
2. Type **c**.
3. Type:

c	to change text by line
NumberPositon	to delete the text defined by the current cursor position and the cursor position indicated by *NumberPosition*, where *Number* is a number greater than 1 (or is omitted) and *Position* is **h**, **l**, **k**, **j**, **|** (logical OR), **0**, **$** (dollar sign), **^** (circumflex accent), **f**, **F**, **t**, **T**, **w**, **W**, **e**, **E**, **)** (right parenthesis), **}** (right brace), **]]** (two right brackets), **b**, **B**, **(** (left parenthesis), **{** (left brace), **[[** (two left brackets), **H**, **M**, **L**, or **G**
'*Letter*	to delete the text defined by the current cursor position and the character indicated by **'***Letter* (grave accent followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark
'*Letter*	to delete the text defined by the current cursor position and the line indicated by **'***Letter* (apostrophe followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark
/*Pattern***Enter**	to delete the text defined by the current cursor position and the string indicated by **/***Pattern***Enter** (slash followed by *Pattern* followed by the **Enter** key), where *Pattern* is the pattern for a search.

4. Type the changed text.

   To make corrections while typing, press:

The **Ctrl–H** keys or the **Backspace** key	to correct the last character
The **Ctrl–W** keys	to correct the last word.

5. Press the **Esc** key to return (from text input mode) to command mode.

To Change a Character between Uppercase and Lowercase:

1. Type ~(tilde).

To Break a Line:

1. Type **i**.
2. Press the **Enter** key.
3. Press the **Esc** key to return (from text input mode) to command mode.

To Join Lines:

1. Type the number (for example, 5) of lines to join (or skip this step to join two lines).
2. Type **J**.

To Shift Lines:

1. Type the number (for example, 5) of lines to shift (or skip this step to shift one line).
2. Type:

>>	(two greater than signs) to shift right
<<	(two less than signs) to shift left.

How to Delete Text with the vi Editor

Prerequisite Tasks or Conditions

1. Edit an existing file or add text.
2. Mark text (if a mark is needed for the deletion).
3. Position the cursor.
4. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Delete Characters:

1. Type the number (for example, 5) of characters to delete from the current line (or skip this step to delete one character).
2. Type:

x	to delete one or more characters at the cursor
X	to delete one or more characters before the cursor.

To Delete Text Objects:

1. Type the number (for example, 5) of text objects to delete (or skip this step to delete one text object).
2. Type **d**.
3. Type:

d	to delete text by line
NumberPositon	to delete the text defined by the current cursor position and the cursor position indicated by *NumberPosition*, where *Number* is a number greater than 1 (or is omitted) and *Position* is **h**, **l**, **k**, **j**, **|** (logical OR), **0**, **$** (dollar sign), **^** (circumflex accent), **f**, **F**, **t**, **T**, **w**, **W**, **e**, **E**, **)** (right parenthesis), **}** (right brace), **]]** (two right brackets), **b**, **B**, **(** (left parenthesis), **{** (left brace), **[[** (two left brackets), **H**, **M**, **L**, or **G**
'*Letter*	to delete the text defined by the current cursor position and the character indicated by **'***Letter* (grave accent followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark
'*Letter*	to delete the text defined by the current cursor position and the line indicated by **'***Letter* (apostrophe followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark
/*Pattern***Enter**	to delete the text defined by the current cursor position and the string indicated by **/***Pattern***Enter** (slash followed by *Pattern* followed by the **Enter** key), where *Pattern* is the pattern for a search.

How to Undo or Repeat Changes or Deletions with the vi Editor

Prerequisite Tasks or Conditions

1. Change text or delete text.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Undo or Retrieve Changes or Deletions:

1. Type:

u	to undo the last change or deletion
U	to undo the last changes or deletions on the current line
"*Number***p**	to retrieve the previous change or deletion indicated by **"***Number* (double quotation mark followed by *Number*), where *Number* is the number (**1** through **9**) of a buffer.

To Repeat Changes or Deletions:

1. Type **.** (period).

How to Mark Text with the vi Editor

Prerequisite Tasks or Conditions

1. Edit an existing file or add text.
2. Position the cursor.
3. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Mark a Character or Line:

1. Type **m***Letter*, where *Letter* is the letter (**a** through **z**) for the mark.

To Move (or Refer) to a Mark:

1. Type:

'Letter	(grave accent followed by *Letter*) to move (or refer) to a marked character
'Letter	(apostrophe followed by *Letter*) to move (or refer) to the beginning of a marked line

   where *Letter* is the letter (**a** through **z**) for the mark.

How to Move or Copy Text with the vi Editor

Prerequisite Tasks or Conditions

1. Edit an existing file or add text.
2. Mark text (if a mark is needed for the move or copy).
3. Position the cursor.
4. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Move Text:

1. Type:

"LowercaseLetter	to replace the contents of *"LowercaseLetter* (double quotation mark followed by *LowercaseLetter*), where *LowercaseLetter* is the name (**a** through **z**) of a buffer
"UppercaseLetter	to append to the contents of *"UppercaseLetter* (double quotation mark followed by *UppercaseLetter*), where *UppercaseLetter* is the name (**A** through **Z**) of a buffer

   or skip this step to use an unnamed buffer.

2. Type the number (for example, 5) of text objects to delete (or skip this step to delete one text object).
3. Type **d**.
4. Type:

d	to delete text by line
NumberPositon	to delete the text defined by the current cursor position and the cursor position indicated by *NumberPosition*, where *Number* is a

number greater than 1 (or is omitted) and *Position* is **h**, **l**, **k**, **j**, **|** (logical OR), **0**, **$** (dollar sign), **^** (circumflex accent), **f**, **F**, **t**, **T**, **w**, **W**, **e**, **E**, **)** (right parenthesis), **}** (right brace), **]]** (two right brackets), **b**, **B**, **(** (left parenthesis), **{** (left brace), **[[** (two left brackets), **H**, **M**, **L**, or **G**

'*Letter*	to delete the text defined by the current cursor position and the character indicated by **'***Letter* (grave accent followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark
'*Letter*	to delete the text defined by the current cursor position and the line indicated by **'***Letter* (apostrophe followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark
/*Pattern***Enter**	to delete the text defined by the current cursor position and the string indicated by **/***Pattern***Enter** (slash followed by *Pattern* followed by the **Enter** key), where *Pattern* is the pattern for a search.

5. Reposition the cursor (if necessary).
6. Type **"***Letter* (double quotation mark followed by *Letter*), where *Letter* is the name (**a** through **z** or **A** through **Z**) of the buffer from step 1, or skip this step if step 1 was skipped.
7. Type:

p	to put the text below the current line
P	to put the text above the current line.

To Copy Text:

1. Type:

"*LowercaseLetter*	to replace the contents of **"***LowercaseLetter* (double quotation mark followed by *LowercaseLetter*), where *LowercaseLetter* is the name (**a** through **z**) of a buffer
"*UppercaseLetter*	to append to the contents of **"***UppercaseLetter* (double quotation mark followed by *UppercaseLetter*), where *UppercaseLetter* is the name (**A** through **Z**) of a buffer

   or skip this step to use an unnamed buffer.
2. Type the number (for example, 5) of text objects to yank (or skip this step to yank one text object).
3. Type **y**.
4. Type:

y	to yank text by line
NumberPositon	to delete the text defined by the current cursor position and the cursor position indicated by *NumberPosition*, where *Number* is a number greater than 1 (or is omitted) and *Position* is **h**, **l**, **k**, **j**, **|** (logical OR), **0**, **$** (dollar sign), **^** (circumflex accent), **f**, **F**, **t**, **T**, **w**, **W**, **e**, **E**, **)** (right parenthesis), **}** (right brace), **]]** (two right brackets), **b**, **B**, **(** (left parenthesis), **{** (left brace), **[[** (two left brackets), **H**, **M**, **L**, or **G**
'*Letter*	to delete the text defined by the current cursor position and the character indicated by **'***Letter* (grave accent followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark
'*Letter*	to delete the text defined by the current cursor position and the line indicated by **'***Letter* (apostrophe followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark

/*Pattern*Enter — to delete the text defined by the current cursor position and the string indicated by /*Pattern*Enter (slash followed by *Pattern* followed by the **Enter** key), where *Pattern* is the pattern for a search.

5. Reposition the cursor (if necessary).
6. Type "*Letter* (double quotation mark followed by *Letter*), where *Letter* is the name (**a** through **z** or **A** through **Z**) of the buffer from step 1, or skip this step if step 1 was skipped.
7. Type:

   p — to put the text below the current line
   P — to put the text above the current line.

How to Search Text with the vi Editor

Prerequisite Tasks or Conditions

1. Edit an existing file or add text.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedure

1. Type:

   / — (slash) to search forward
   ? — (question mark) to search backward.

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type *Pattern*, where *Pattern* is a character string that is the pattern for the search.

   The pattern can be either a literal string or a regular expression that is made up of literal characters and the special characters ^ (circumflex accent), $ (dollar sign), . (period), [ ] (left and right brackets), – (minus sign), * (asterisk), \< (slash followed by less than sign), \> (slash followed by greater than sign), and \ (reverse slash).

3. To position the cursor anywhere on the screen at the sought pattern, omit this step and step 4.

   Type / (slash) or **?** (question mark), whichever was typed in step 1.

4. Type:

   +*Number* — (plus sign followed by *Number*) to move a *Number* of lines after the sought pattern
   –*Number* — (minus sign followed by *Number*) to move a *Number* of lines before the sought pattern
   z — to position the sought pattern at the top of the screen
   z. — (**z** followed by period) to position the sought pattern at the middle of the screen
   z– — (**z** followed by minus sign) to position the sought pattern at the bottom of the screen.

5. Press the **Enter** key to search the text and return (from last line mode) to command mode.

   To repeat the search, type:

   n — to search in the same direction
   N — to search in the opposite direction.

Moving within a File with the vi Editor

The subcommands for moving within a file allow you to move in the following ways:

Moving within a Line

Moving within a Line by Character Positioning

Moving to Words

Moving by Line Positioning

Moving to Sentences, Paragraphs, or Sections

Moving by Redrawing the Screen

Paging and Scrolling

Searching for Patterns

Marking a Specific Location in a File and Returning.

vi Subcommands for Moving within a File

Moving within a Line

Enter the following subcommands in command mode. You can cancel an incomplete command by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

(Left arrow) or **h** or **Ctrl–H**	Moves the cursor one character to the left.
(Down arrow) or **j** or **Ctrl–J** or **Ctrl–N**	Moves the cursor down one line (but it remains in the same column).
(Up arrow) or **k** or **Ctrl–P**	Moves the cursor up one line (but it remains in the same column).
(Right arrow) or **l**	Moves the cursor one character to the right.

Moving within a Line by Character Positioning

Enter the following subcommands in command mode. You cancel an incomplete command by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

^	Moves the cursor to the first nonblank character.
0	Moves the cursor to the beginning of the line.
$	Moves the cursor to the end of the line.
f*x*	Moves the cursor to the next *x* character.
F*x*	Moves the cursor to the last *x* character.
t*x*	Moves the cursor to one column before the next *x* character.
T*x*	Moves the cursor to one column after the last *x* character.
;	Repeats the last **f**, **F**, **t**, or **T** subcommand.

,	Repeats the last **f**, **F**, t, or **T** subcommand in the opposite direction.
Number|	Moves the cursor to the specified column.

Moving to Words

Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see general subcommand syntax.

w	Moves the cursor to the next small word.
b	Moves the cursor to the previous small word.
e	Moves the cursor to the end of a small word.
W	Moves the cursor to the next big word.
B	Moves the cursor to the previous big word.
E	Moves the cursor to the end of a big word.

Moving by Line Positioning

Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see general subcommand syntax.

H	Moves the cursor to the top line on the screen.
L	Moves the cursor to the last line on the screen.
M	Moves the cursor to the middle line on the screen.
+	Moves the cursor to the next line at its first nonblank character.
–	Moves the cursor to the previous line at its first nonblank character.
Enter	Moves the cursor to the next line at its first nonblank character.

Moving to Sentences, Paragraphs, or Sections

Entering the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

(	Places the cursor at the beginning of the previous sentence (or the previous s–expression if you are in LISP mode).
)	Places the cursor at the beginning of the next sentence (or the next s–expression if you are in LISP mode).
{	Places the cursor at the beginning of the previous paragraph (or at the next list if you are in LISP mode).
}	Places the cursor at the beginning of the next paragraph, at the next section if you are in C mode, or at the next list if you are in LISP mode.
]]	Places the cursor at next section (or function if you are in LISP mode).
[[	Places the cursor at previous section (or function if you are in LISP mode).

Moving by Redrawing the Screen

Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

z	Redraws the screen with the current line at the top of the screen.
z–	Redraws the screen with the current line at the bottom of the screen.
z.	Redraws the screen with the current line at the center of the screen.
/*Pattern*/**z–**	Redraws the screen with the line containing *Pattern* at the bottom.

Paging and Scrolling

Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

Ctrl–U	Scrolls up one half screen.
Ctrl–D	Scrolls down one half screen.
Ctrl–F	Scrolls forward one screen.
Ctrl–B	Scrolls backward one screen.
Ctrl–E	Scrolls the window down one line.
Ctrl–Y	Scrolls the window up one line.
z+	Pages up.
z^	Pages down.

Searching for Patterns

Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

[*Number***]G**	Places the cursor at line number *Number* or at the last line if *Number* is not specified.
/*Pattern*	Places the cursor at the next line containing *Pattern*.
?*Pattern*	Places the cursor at the next previous line containing *Pattern*.
n	Repeats last search for *Pattern* in the same direction.
N	Repeats last search for *Pattern* in the opposite direction.
/*Pattern*/+*Number*	Places the cursor the specified *Number* of lines after the line matching *Pattern*.
?*Pattern***?–***Number*	Places the cursor the specified *Number* of lines before the line matching *Pattern*.
%	Finds the parenthesis or brace that matches the one at the current cursor position.

Marking a Specific Location in a File and Returning

Enter the following subcommands in command mode. You can cancel an incomplete subcommand by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

`	Moves the cursor to the previous location of the current line.
'	Moves the cursor to the beginning of the line containing the previous location of the current line.
m*x*	Marks the current position with the letter specified by *x*.
`*x*	Moves the cursor to the mark specified by *x*.
'*x*	Moves the cursor to the beginning of the line containing the mark specified by *x*.

vi Subcommands for Adjusting the Screen

Enter the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the **Esc** key.If you need information about the format of vi subcommands, see general subcommand syntax.

Ctrl–L	Clears and redraws the screen.
Ctrl–R	Redraws the screen and eliminates blank lines marked with @.
z*Number*	Makes the window the specified *Number* of lines long.

Manipulating Files with the vi Editor

The subcommands for manipulating files allow you to do the following tasks:

Saving Changes to a File

Finding Out File Information

Editing a Second File

Editing a List of Files.

vi Subcommands for Manipulating Files

Editing a Second File

Enter the following subcommands in command mode. An incomplete subcommand can be canceled by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

:e *File*	Edits the specified file. If you are using this subcommand from the ex editor, you do not need to type the : (colon).
:e!	Re–edits the current file and discards all changes.
:e + *File*	Edits the specified file starting at the end.
:e + *Number*	Edits the specified file starting at the specified line number.
:e #	Edits the alternate file. The alternate file is usually the previous current file name. However, if changes are pending on the current file when a new file

is called, the new file becomes the alternate file. This subcommand is the same as the **Ctrl–A** subcommand.

:r *File* — Reads the file into the editing buffer by adding new lines below the current line. If you are using this subcommand from the ex editor, you do not need to type the : (colon).

:r !*Command* — Runs the specified AIX command and places its output into the file by adding new lines below the current cursor position.

:ta *Tag* — Edits a file containing *Tag* at the location of *Tag*. To use this command, you must first create a database of function names and their locations using the **ctags** command. If you are using this subcommand from the ex editor, you do not need to type the : (colon).

Ctrl–A — Edits the alternate file. The alternate file is usually the previous current file name. However, if changes are pending on the current file when a new file is called, the new file becomes the alternate file. This subcommand is the same as the **:e #** subcommand.

Editing a List of Files

Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see general subcommand syntax.

:n — Edits the next file in the list entered on the command line. If you are using this subcommand from the ex editor, you do not need to type the : (colon).

:n *Files* — Specifies a new list of files to edit. If you are using this subcommand from the ex editor, you do not need to type the : (colon).

Finding Out File Information

Enter the following subcommand in command mode. If you need information about the format of vi subcommands, see general subcommand syntax.

Ctrl–G — Shows the current file name, the current line number, the number of lines in the file, and the percentage of the way through the file where the cursor is located.

How to Manipulate Files with the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Position the cursor.
3. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Read a File:

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type the number (for example, `10`) of the line to insert the file after (or skip this step to insert the file after the current line).
3. Type **r**.
4. Press the spacebar and type the name of the file to read.
5. Press the **Enter** key to read the file and return (from last line mode) to command mode.

To Write to a File:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.
2. To write all lines to the file, omit this step and steps 3 and 4.

   Type the number (for example, `5`) of the first line to write.
3. Type **,** (comma).
4. Type the number (for example, `10`) of the last line to write.
5. Type **w**.
6. To write to the original file, omit this step and step 7.

   Type **!** (exclamation point) to overwrite an existing file (or skip this step to write to a new file).
7. Press the spacebar and type the name of the file to write to.
8. Press the **Enter** key to write to the file and return (from last line mode) to command mode.

To Edit a File:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.
2. Type **e**.
3. Type **!** (exclamation point) to edit a file without writing the changed current file (or skip this step if the current file is unchanged or was written after the last change).
4. Press the spacebar and type the name of the file to edit.
5. Press the **Enter** key to edit the file and return (from last line mode) to command mode.

   To edit the previous file again, type **#** (pound sign) in place of steps 3 through 5.

To Edit the Next File from a List of Files:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.
2. Type **n**.
3. Type **!** (exclamation point) to edit the next file without writing the changed current file (or skip this step if the current file is unchanged or was written after the last change).
4. Press the spacebar and type a new list of files to edit (or skip this step to edit the next file from the current list).
5. Press the **Enter** key to edit the next file and return (from last line mode) to command mode.

How to Control the Screen with the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Scroll the Screen:

1. Type the number (for example, 2) of units to scroll (or skip this step to scroll one unit).
2. Press:

The **Ctrl–E** keys	to scroll down by line
The **Ctrl–D** keys	to scroll down by half screen
The **Ctrl–F** keys	to scroll forward by full screen
The **Ctrl–Y** keys	to scroll up by line
The **Ctrl–U** keys	to scroll up by half screen
The **Ctrl–B** keys	to scroll backward by full screen.

To Position a Particular Line:

1. Type the number (for example, 5) of the line (or skip this step to position the current line).
2. Type **z**.
3. To change the number of lines on a screen, type the new number (for example, 10) of lines (or skip this step to leave the number of lines unchanged).
4. Press the **Enter** key to position the line at the top of the screen or type:

.	(period) to position the line in the middle of the screen
–	(minus sign) to position the line at the bottom of the screen.

To Clean Up the Screen:

1. Press:

The **Ctrl–R** keys	to remove blank lines with @ (at sign) at the beginning
The **Ctrl–L** keys	to eliminate extraneous lines from outside messages or transmission errors.

How to Quit the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Write and Quit:

Use either procedure A or procedure B.

Procedure A

1. Type **ZZ** to write the current file and quit the vi editor.

Procedure B

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **wq**.

3. Press the **Enter** key to write the current file and quit the vi editor.

To Quit without Writing:

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **q**.

3. Type **!** (exclamation point) to quit without writing the changed current file (or skip this step if the current file is unchanged or was written after the last change).

4. Press the **Enter** key to quit the vi editor.

Subcommands for Interrupting and Ending the vi Editor

Enter the following subcommands in command mode. If you need information about the format of vi subcommands, see general subcommand syntax.

Q	Enters the ex editor in command state.
ZZ	Exits the vi editor, saving changes.
:q	Quits the vi editor. If you have changed the contents of the editing buffer, the vi editor displays a warning message and does not quit. If you are using this subcommand from the ex editor, you do not need to type the : (colon).
:q!	Quits the vi editor, discarding the editing buffer. If you are using this subcommand from the ex editor, you do not need to type the : (colon).
Esc	Ends text input or ends an incomplete subcommand.
Ctrl–?	Interrupts a subcommand.

Defining Macros with the vi Editor

If you use a subcommand or sequence of subcommands frequently, you can create a macro that issues the subcommand or sequence. To create a macro, enter the sequence of subcommands into a buffer named with a letter of the alphabet. When used, **a** through **z** overlay the contents of the buffer; **A** through **Z** append text to the previous contents of the buffer, allowing the building of a macro piece by piece.

To invoke the macro, enter @*x*, where *x* is the letter name of the buffer. Enter @@ to repeat the last macro you invoked.

The character set used by your system is defined by the collation table. This table affects the performance of vi macros. See vi character sets for more information.

vi Character Sets

The collation table defines the alphanumeric set used by your system. This table affects the performance of vi macros and subcommands. If you intend to use non–ASCII extended characters with vi macros, it may be necessary to revise this table using the **ctab** command.

The vi editor uses the collation table to distinguish between a *small word* and a *big word.* A small word is bounded by alphabetic characters or numbers as defined in the collation table. For example, `isn't` is two small words. The `'` (apostrophe) is not a number or an alphabetic character, and it bounds both the small word `t` and the small word `isn`. A big word is bounded by blanks, tabs, or new–line indicators. For example, `stop` is a big word.

Mapping Keys with the vi Editor

The vi editor allows you to temporarily map a keystroke to a subcommand or a sequence of subcommands so that the mapping is in effect for only the current editing session or to permanently map a keystroke to a subcommand or a sequence of subcommands so that the mapping is in effect automatically for every editing session.

Mapping Keys Temporarily with the vi Editor

You can use the **:map** subcommand from the command line to map a keystroke to a subcommand or a sequence of subcommands. To map a key, enter **:map** *Key Subcommand,* where *Key* is the key you want to assign to a subcommand or sequence of subcommands, and *Subcommand* is the subcommand or sequence of subcommands. For example, to map the @ key to delete lines, enter:

```
:map @ dd
```

In this example, `@` is the key to which the subcommand is assigned, and `dd` is the subcommand.

In the next example, a subcommand sequence is mapped to a key:

```
:map * {>}
```

The `*` is the key to which the subcommand sequence is assigned, and `{>}` is the subcommand sequence. The `{` moves the cursor to the beginning of the paragraph, and the `>` indents the paragraph to the next shiftwidth.

Enter the **:map** or **:map!** subcommand to display a list of the current key mappings in command mode or text input mode, respectively. To remove a key map, enter **:unmap** *String* or **:unmap!** *String*, where *String* is the string used after the **:map** subcommand to set the key and subcommand sequence. For example, to remove key map for the previous example, enter:

```
:unmap * {>}
```

If function keys are defined for your terminal, you can include them in the **:map** or **:unmap** subcommand by pressing the **Ctrl–V** keys before pressing the desired key. It is useful to define keys seldom used in editing with another key or function key (**F0 – F12**). For example, the **Shift** key, the **Ctrl** key, or the **Alt** key can be defined in this way.

The **:ab** (abbreviation) subcommand is similar to the **:map** subcommand. For example, if you set the letter s equal to Sam with a **:map** subcommand and then enter the following sentence:

```
s ate apples.
```

it is displayed as

```
Sam ate appleSam.
```

The **:map** subcommand does not recognize the difference between the macro `s` and the text `s`. With some restrictions, the **:ab** subcommand does distinguish between text and a macro. Setting the macro in the previous example with the **:ab** subcommand,

```
:ab s Sam
```

and typing the same sentence would result in the correct sentence, `Sam ate apples`. The **:ab** subcommand only recognizes a macro when it is preceded by a blank space or tab character. If the following were entered,

```
Sam swims
```

the **:ab** subcommand would translate the sentence as:

```
Sam Samwims
```

The abbreviated item can occupy more than one line. However, you must use the **ex** editor to remove an abbreviation that contains a **Ctrl–M** (enter sequence). After entering the **ex** editor, enter **:unab** *Abbreviation*, and return to the vi editor. Remove abbreviations without the **Ctrl–M** by using **:ab** *Abbreviation*. After removing or changing abbreviations created with the **:ab** subcommand, enter **:ab** to list all currently defined abbreviations.

If you use an IBM 3161 ASCII Display Station, IBM 3163 ASCII Display or IBM 3101 ASCII Display Station, the default key mapping of the vi editor can cause you to lose data. To see the default mapping, issue a **:map** subcommand. Specific problems arise with the **Esc–J** or **Shift–J** sequence. These sequences delete any information from the current position of the cursor to the end of the file. To avoid problems, change this sequence using a **.exrc** file.

Mapping Keys Permanently with the vi Editor

You can map keys permanently by putting the **map** subcommand into a **.exrc** file. Each time you start the editor, it reads this file. To map a key, enter **map** *Key Subcommand,* where *Key* is the key you want to assign to a subcommand or sequence of subcommands, and *Subcommand* is the subcommand or sequence of subcommands. For example, to map the @ key to delete lines, put the following subcommand into a **.exrc** file:

```
map @ dd
```

In this example, `@` is the key to which the subcommand is assigned, and `dd` is the subcommand.

In the next example, a subcommand sequence is mapped to a key:

```
map * {>}
```

The `*` is the key to which the subcommand sequence is assigned and `{>}` is the subcommand sequence. The `{` moves the cursor to the beginning of the paragraph and the `>` indents the paragraph to the next shiftwidth.

To remove a key map, remove the appropriate subcommand from the **.exrc** file.

Enter the **:map** or **:map!** subcommand to display a list of the current key mappings in command mode or text input mode, respectively.

If function keys are defined for your terminal, you can include them in the **map** subcommand by pressing the **Ctrl–V** keys before pressing the desired key. It is useful to define keys seldom used in editing with another key or function key (**F0** – **F12**). For example, the **Shift** key, the **Ctrl** key, or the **Alt** key can be defined in this way.

How to Run AIX Commands from the vi Editor

Prerequisite Tasks or Conditions

1. Position the cursor.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Run One AIX Command:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.
2. Type **!** (exclamation point).
3. Type an AIX command.
4. Press the **Enter** key to run the command and display its output.
5. Press the **Enter** key to return (from last line mode) to command mode.

   To run the same command again, type **!** (exclamation point) in place of step 3.

To Run Several AIX Commands:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.
2. Type **sh** and press the **Enter** key.

   A prompt appears.
3. Type an AIX command.
4. Press the **Enter** key to run the command and display its output.
5. To run other commands, repeat steps 3 and 4.
6. Press the **Ctrl–D** keys to return (from last line mode) to command mode.

To Add Output from an AIX Command:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.
2. Type the number (for example, `10`) of the line to add the output after (or skip this step to add the output after the current line).
3. Type **r**.
4. Type **!** (exclamation point).
5. Type an AIX command.
6. Press the **Enter** key to run the command, add its output, and return (from last line mode) to command mode.

To Input Lines to an AIX Command and Replace Them with Output from the Command:

Use either procedure A or procedure B.

Procedure A

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.
2. Type the number (for example, 5) of the first line to input to the command.
3. Type , (comma).
4. Type the number (for example, 10) of the last line to input to the command.
5. Type ! (exclamation point).
6. Type an AIX command.
7. Press the **Enter** key to run the command, replace the lines with its output, and return (from last line mode) to command mode.

Procedure B

1. Unless ! (exclamation point) is typed in step 3, omit this step.

   Type the number (for example, 5) of lines to input to the command (or skip this step to input only the current line).
2. Type ! (exclamation point).
3. Type:

!	(exclamation point) to input the lines indicated in step 1
NumberPositon	to input the lines defined by the current line and the line indicated by *NumberPosition*, where *Number* is a number greater than 1 (or is omitted) and *Position* is) (right parenthesis), } (right brace),]] (two right brackets), ((left parenthesis), { (left brace), [[(two left brackets), **H**, **M**, **L**, or **G**
'*Letter*	to input the lines defined by the current line and the line indicated by '*Letter* (apostrophe followed by *Letter*), where *Letter* is the letter (**a** through **z**) for a mark.

   The cursor moves to the bottom of the screen to indicate last line mode.
4. Type an AIX command.
5. Press the **Enter** key to run the command, replace the lines with its output, and return (from last line mode) to command mode.

Entering Shell Commands within the vi Editor

The following subcommands allow you to run an AIX command within the vi editor. Enter these subcommands in command mode. An incomplete subcommand can be canceled by pressing the **Esc** key. If you need information about the format of vi subcommands, see general subcommand syntax.

:sh Enters the shell to allow you to run more than one AIX command. You can return to the editor by pressing the **Ctrl–D** keys. If you are using this subcommand within the ex editor, you do not need to type the : (colon).

:!*Command* Runs the specified AIX command and then returns to the editor. If you are using this subcommand within the ex editor, you do not need to type the : (colon).

:!! Repeats the last :!*Command* subcommand.

Number!!*Command*
Executes specified AIX command and replaces the lines specified by *Number* with the output of *Command*. If *Number* is not specified, the default value is 1. If *Command* expects standard input, the specified lines are used as input.

!*Object Command*
Executes the specified AIX command and replaces *Object* with the output of *Command*. If *Command* expects standard input, the specified *Object* is used as input.

How to Run Line Editor Subcommands with the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedure

1. Type **Q**.

   The cursor moves to the bottom of the screen, and a prompt appears.
2. Enter one or more **ex** subcommands (for example, **s** or **g**).
3. Enter `vi` to return (from last line mode) to command mode.

How to Define and Use Macros with the vi Editor

Prerequisite Tasks or Conditions

1. Start the vi editor.
2. Be in command mode. (If you are not sure, press the **Esc** key.)

Procedures

To Define a Buffer Macro:

1. Type **o**.
2. Type one or more vi subcommands.
3. Press the **Esc** key.
4. Type "*Letter* (double quotation mark followed by *Letter*), where *Letter* is the name (**a** through **z**) of a buffer.
5. Type **dd**.

To Use a Buffer Macro:

1. Type @*Letter* (at sign followed by *Letter*), where *Letter* is the name (**a** through **z**) of the buffer containing the macro.

   To use the same macro again, type @@ (two at signs).

To List Maps:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **map**.
3. Press the **Enter** key to list the maps.
4. Press the **Enter** key to return (from last line mode) to command mode.

To Map a Key:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **map**.
3. Press the spacebar and press the key being mapped.
4. Press the spacebar and type one or more vi subcommands.
5. Press the **Enter** key to map the key and return (from last line mode) to command mode.

To Unmap a Key:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **unmap**.
3. Press the spacebar and press the key being unmapped.
4. Press the **Enter** key to unmap the key and return (from last line mode) to command mode.

To List Abbreviations:

1. Type **:** (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **ab**.
3. Press the **Enter** key to list the abbreviations.
4. Press the **Enter** key to return (from last line mode) to command mode.

To Define an Abbreviation:

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **ab**.
3. Press the spacebar and type the character string being defined as an abbreviation.
4. Press the spacebar and type the character string being abbreviated.
5. Press the **Enter** key to define the abbreviation and return (from last line mode) to command mode.

To Remove an Abbreviation:

1. Type : (colon).

   The cursor moves to the bottom of the screen to indicate last line mode.

2. Type **unab**.
3. Press the spacebar and type the character string being removed as an abbreviation.
4. Press the **Enter** key to remove the abbreviation and return (from last line mode) to command mode.

Addressing Lines in a File with the ex Editor

There are three types of ex addresses: line number addresses, addresses relative to the current line, and pattern addresses. The current line is the point of reference in the buffer.

Following are guidelines for constructing addresses:

- . (period) addresses the current line. This is the default for most ex subcommands and does not need to be specified.
- $ (dollar sign) addresses the last line of the buffer.
- *Number* addresses the specified line number of the buffer.
- '*x* addresses the line marked with a lowercase ASCII letter, represented by *x*, by the **ma** subcommand.
- % addresses the first through the last lines in the buffer.
- */pattern/* (a pattern enclosed in slashes; a pattern can be a fixed character string or a regular expression) addresses the next line containing a matching string. The search begins with the line after the current line and stops when it finds a match for the pattern. If necessary, the search moves to the end of the buffer, wraps around to the beginning of the buffer, and continues until it either finds a match or returns to the current line.
- ?*pattern*? (a pattern enclosed in question marks; a pattern can be a fixed character string or a regular expression) addresses the previous line that contains a match for the pattern. The ?*pattern*? construct, line */pattern/*, can search the entire buffer, but it does so in the opposite direction.
- An address followed by +*number* or –*number* (a plus sign or a minus sign followed by a decimal number) specifies an address plus or minus the indicated number of lines. (The + sign is optional.)
- An address that begins with + or – specifies a line relative to the current line. For example, –5 is the equivalent of .–5 (five lines above the current line).
- An address that ends with – or + specifies the line immediately before (–) or immediately after (+) the addressed line. Used alone, the – character addresses the line immediately after the current line; however, the + character is optional. The + and – characters have a cumulative effect; for example, the address ++ addresses the line two lines below the current line.

Subcommands that do not accept addresses regard the presence of an address as an error. Subcommands that do accept addresses can use either given or default addresses. When given more addresses than it accepts, a subcommand uses the last (rightmost) one(s).

In most cases, commas (,) separate addresses (for example `2,8`). The first address must be numerically smaller than the second.

Regular Expressions in the ex Editor

A regular expression is a string constructed of special pattern matching characters. Using a regular expression to locate text in a file gives you more flexibility than trying to locate a fixed character string.

Pattern Matching in the ex Editor

The **ex** command supports the following pattern matching characters that you can use as regular expressions to construct pattern strings:

^	Finds a match only at the beginning of a line.
$	Finds a match only at the end of a line.
.	Matches any character.
\<	Finds a match at the beginning of a word.
\>	Finds a match at the end of a word.
[*string***]**	Finds a match for any single character in *string*.
[^*string***]**	Finds a match for any single character not in *string*.
[*x–y***]**	Finds a match for any character between the characters specified by x and y.
*	Finds a match for zero or more repetitions of the preceding character.

List of ex Subcommands

Subcommand	Description
a	Appends text. If you are using the ex editor in visual mode, see the **a** subcommand for the vi editor.
ab	Creates an abbreviation.
args	Displays the argument list.
c	Changes lines of text. If you are using the ex editor in visual mode, see the **c** subcommand for the vi editor.
co	Copies lines of text.
d	Deletes lines of text.
e	Edits another file.
f	Prints the file status.
g	Makes global changes.
i	Inserts text. If you are using the ex editor in visual mode, see the **i** subcommand for the vi editor.
j	Joins lines.
l	Displays lines.
m	Moves text.
ma	Marks lines of text.
map	Defines macros.
n	Edits next file in a list of files.
nu	Displays lines and line numbers.
p	Displays lines.

preserve	Preserves the buffer after a system crash.
pu	Puts back deleted or yanked lines.
q	Ends and exits the editor.
r	Reads another file into the buffer.
recover	Recovers a file.
rewind	Rewinds the parameter list.
s	Substitutes text.
set	Displays options whose values have been changed.
sh	Creates a new shell.
so	Reads and executes a source file.
ta	Edits a file containing *Tag*.
u	Undoes a change.
unab	Removes an abbreviation.
unmap	Removes the defined macro.
vi	Enters visual mode.
w	Saves changes.
x	Ends the editor.
ya	Copies lines.
z	Displays a window of text.
!*AIX–command*	Processes an AIX command.
<	Shifts lines left.
>	Shifts lines right.
Enter	Displays next line.
&	Repeats last substitute command.
Ctrl–D	Scrolls through the buffer.

ex Subcommands

Adding Text

In the following subcommands, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address*]**a**
text
.

If you do not specify an *address*, the **a** subcommand appends the text you type after the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **a** subcommand appends text after the addressed line. If you specify address 0, the **a** subcommand places the text at the beginning of the buffer.

Type your text, pressing Enter at the end of each line. When you have entered all your text, type a . (period) alone at the start of a line. This ends text input mode and returns to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer. The **a** subcommand differs from the **i** subcommand in the placement of text.

[*address*]**i**
text
.

If you do not specify an *address*, the **i** subcommand inserts text before the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **i** subcommand inserts text before the addressed line.

Type your text, pressing Enter at the end of each line. When you have entered all your text, type a . (period) alone at the start of a line. This ends text input mode and returns to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer. The **i** subcommand differs from the **a** subcommand in the placement of text.

Changing Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1,address2*]**c**
text
.

If you do not specify an *address*, the **c** subcommand replaces the current line with the text you type. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **c** subcommand replaces the addressed line (a single address) or a range of lines (two addresses separated by a comma).

Type your text, pressing Enter at the end of each line. When you have entered all your text, type a . (period) alone at the start of a line. This ends text input mode and returns to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer. The last input line becomes the current line.

Copying Text

In the following subcommand, *address1* and *address2* are optional. You must specify *address3*. If you specify an *address*, you do not type the brackets.

[*address1,address2*]**co** *address3*

If you do not specify an *address*, the **co** subcommand moves the current line after the line specified by *address3*. If you specify an *address*, the **co** subcommand moves the addressed line (a single address) or a range of lines (two addresses separated by a comma). You may need to find out the current line or specify *address1* and/or *address2* if you are not in the correct position in the buffer. See addressing lines in a file for more information. The last line copied becomes the current line. You can use the **1,$p** subcommand to display the entire contents of the buffer.

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1,address2*]**ya** [*buffer*]

If you do not specify an *address*, the **ya** subcommand copies the current line; otherwise, it copies the addressed line (a single address) or a range of lines (two addresses separated by a comma) into a *buffer* (specified by a single alpha character name **a** – **z**).You may need to find out the current line

or specify an address if you are not in the correct position in the buffer. You can use the **pu** subcommand to put these lines into another file.

Deleting Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1*,*address2*]**d** [*buffer*]

If you do not specify an *address*, the **d** subcommand deletes the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **d** subcommand deletes the addressed line (a single address) or a range of lines (two addresses separated by a comma). If you specify a *buffer* by giving a letter from a to z, the editor saves the addressed lines in that buffer or, if the letter is uppercase, appends the lines to that buffer. You can put the deleted lines back into the buffer by using the **pu** subcommand.

Displaying Text and Finding Out the Current Line

In the following subcommands, *address* is optional. If you specify an *address*, you do not type the brackets.

(*address1*,*address2*)**l**

If you do not specify an *address*, the **l** subcommand displays the current line. If you specify an *address*, the **l** subcommand displays the addressed line (a single address) or a range of lines (two addresses separated by a comma). See addressing lines in a file for more information. The **l** subcommand wraps long lines and, unlike the **p** subcommand, represents nonprinting characters, either with mnemonic overstrikes or in hexadecimal notation.

[*address1*,*address2*]**nu**

Displays the addressed line or lines preceded by its buffer line number. If you do not specify an *address*, the **nu** subcommand displays the current line and number. If you specify an *address*, the **nu** subcommand displays the addressed line (a single address) or a range of lines (two addresses separated by a comma). See addressing lines in a file for more information. The last line displayed becomes the current line.

[*address1*,*address2*]**p**

If you do not specify an *address*, the **p** subcommand displays the current line. If you specify an *address*, the **p** subcommand displays the addressed line (a single address) or a range of lines (two addresses separated by a comma). See addressing lines in a file for more information. The last line displayed becomes the current line.

[*address*]**z** If an *address* is not specified, displays a screen of text beginning with the current line. If an *address* is specified, displays a screen of text beginning with the addressed line. See addressing lines in a file for more information.

Displaying and Rewinding the Argument List

args Displays the argument list you specified when you started the ex editor (the file or files that you specified).

rewind Rewinds the argument list to the first argument (*File*) you specified when you started the ex editor.

Displaying the Current File Name and Status

f Displays the current file name along with the following information about it:

- Whether it has been modified since the last **w** subcommand
- What the current line is
- How many lines are in the buffer
- What percentage of the way through the buffer the current line is.

Displaying Editor Options

set Displays the editor options that have been changed. The **set all** subcommand displays all the editor options. You can change editor options within the vi editor.

Ending and Exiting the ex Editor

x Saves any changes made to the buffer and ends the ex editor. The **x** subcommand is a quick way to end the editor as it does not require you to save the changes with the **w** subcommand and then end the editor with the **q** subcommand.

Entering Visual and Open Mode

In the following subcommands, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address*]**vi** Enters visual mode at the addressed line. If no *address* is specified, enters visual mode with the current line at the top of the screen. You may want to find out the current line or specify an address if you are not in the correct position in the buffer. You can change the placement of the addressed line by using the **vi.** subcommand to put the line in the middle of the screen or the **vi–** to put the line at the bottom of the screen. To return to **ex** command mode, enter the **Q** subcommand.

[*address*]**o** Enters open mode (a one–window line of text) at the addressed line. If no *address* is specified, enters open mode at the current line. You may want to find out the current line or specify an address if you are not in the correct position in the buffer. To return to **ex** command mode, enter the **Q** subcommand.

Executing Subcommands from a Source File

so Reads and executes subcommands from the specified *File*. Source files can be nested, allowing one file to call another; however, there is no return mechanism.

Joining Lines

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1*,*address2*]**j**

The **j** subcommand joins contiguous lines of text. If you do not specify an *address*, the **j** subcommand joins the current line with the following line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer.

Making Global Changes

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1*,*address2*]**g**/*pattern*/*subcommand–list*

Marks each of the addressed lines that match the *pattern*. A *pattern* can be a fixed character string or a regular expression. Then the editor carries out the specified subcommands on each marked line. If you do not specify an *address*, the **g** subcommand works on the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **g** subcommand works on the addressed line (a single address) or a range of lines (two addresses separated by a comma).

A single *subcommand* or the first *subcommand* in a subcommand list appears on same line as the **g** subcommand. The remaining subcommands must appear on separate lines, where each line (except the last) ends with a \ (reverse slash). The default subcommand is the **p** subcommand. The list can include the **a** subcommand, **i** subcommand, and **c** subcommand, and their associated input. In this case, if the ending period comes on the last line of the command list, you can omit it. The **u** subcommand and the **g** subcommand itself, however, may not appear in the command list.

Marking Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the parenthesis.

[*address*]**ma** *x* If an *address* is not specified, marks the current line with *x* (a lowercase case letter that you specify); otherwise, marks the addressed line. See addressing lines in a file for more information. The marked line can be addressed by using the '*x* subcommand, where *x* is the mark you specified.

Moving Text

In the following subcommand, *address1* and *address2* are optional. You must specify *address3*. If you specify an *address*, you do not type the brackets.

[*address1*,*address2*]**m** *address3*

If you do not specify an *address*, the **m** subcommand moves the current line after the line specified by *address3*. If you specify an *address*, the **m** subcommand moves the addressed line (a single address) or a range of lines (two addresses separated by a comma). You may need to find out the current line or specify an address if you are not in the correct position in the buffer. The first moved line becomes the current line.

Moving through the Buffer

Enter Pressing the **Enter** key, moves the current line forward through the buffer one line at a time.

Retrieving Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address*]**pu** [*buffer*]

If you do not specify an *address*, the **pu** subcommand retrieves the contents of the specified *buffer* and places it after the current line. You may need to

find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **pu** subcommand retrieves the contents of the specified *buffer* and places it after the addressed line. If you do not specify a *buffer*, the editor restores the last deleted or yanked line. Thus, you can use this subcommand together with the **d** subcommand to move lines within a file or with the **ya** subcommand to duplicate lines between files.

Saving Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1,address2*]**w**

If no *address* is specified, the **w** subcommand writes the entire contents of the buffer to the file. Otherwise, the **w** subcommand writes the specified line (a single address) or a range of lines (two addresses separated by a comma). See addressing lines in a file for more information.

Scrolling through Text

Ctrl–D

Pressing the **Ctrl–D** keys (END OF FILE) scrolls through the file, a half screen of text at a time. The editor displays a message when you are at the end of the file.

Shifting Lines

In the following subcommands, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1,address2*]**<**

If no *address* is specified, shifts the current line to the left; otherwise, shifts the addressed line (a single address) or a range of lines (two addresses separated by a comma). You may need to find out the current line or specify an address if you are not in the correct position in the buffer. The amount the lines are shifted is determined by the **shiftwidth** option.

[*address1,address2*]**>**

If no *address* is specified, shifts the current line to the right; otherwise, shifts the addressed line (a single address) or a range of lines (two addresses separated by a comma). You may need to find out the current line or specify an address if you are not in the correct position in the buffer. The amount the lines are shifted is determined by the **shiftwidth** option.

Substituting Text

In the following subcommands, *address* is optional. If you specify an *address*, you do not type the brackets.

[*address1,address2*]**s**/*pattern*/*replacement*/
[*address1,address2*]**s**/*pattern*/*replacement*/**g**

Replaces on each addressed line the first instance of *pattern* with the text specified. A pattern can be a fixed character string or a regular expression. If you add the **g** subcommand, the **s** subcommand replaces all instances of *pattern* on each addressed line. If you do not specify an *address*, the **s** subcommand works on the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **s** subcommand works on the

addressed line (a single address) or a range of lines (two addresses separated by a comma).

[*address1*,*address2*]**&**
: If no *address* is specified, repeats the previous **s** subcommand on the current line; otherwise, repeats the previous **s** subcommand on the addressed line (a single address) or range of lines (two addresses separated by a comma). You may need to find out the current line or specify an address if you are not in the correct position in the buffer.

Editing a File with the edit Editor

edit Subcommand Syntax

The general format of an edit subcommand is as follows:

[*address*] subcommand [*parameters*] [*count*]

Note: If you supply *address, parameters,* or *count*, you do not type the brackets. The brackets only mean that this information is optional.

If you do not specify an *address*, the **edit** command works on the current line. If you add a numeric *count* to most subcommands, the **edit** command works on the specified number of lines.

The value for *address* can be a line number or a *pattern* to be matched or, in some cases, a range of line numbers or *pattern*s. See addressing lines in a file for more information about addressing.

edit Subcommands for Editing a File

The subcommands for editing a file allow you to do the following tasks:

Adding Text

Changing Text

Deleting Text

Displaying Text and Finding Out the Current Line

Making Global Changes

Moving or Copying Text

Saving Text

Substituting Text

Undoing a Change.

Adding Text

In the following subcommands, *address* is optional. If you specify an *address*, you do not type the brackets. You can use the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address*]**append (a)**
text
.

If you do not specify an *address*, the **a** subcommand appends the text you type after the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **a** subcommand appends text after the addressed line. If you specify address 0, the **a** subcommand places the text at the beginning of the buffer.

Type your text, pressing Enter at the end of each line. When you have entered all your text, type a . (period) alone at the start of a line. This ends text input mode and returns to

command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer. The **a** subcommand differs from the **i** subcommand in the placement of text.

[*address*]**insert** (**i**)
text
.

If you do not specify an *address*, the **i** subcommand inserts text before the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **i** subcommand inserts text before the addressed line.

Type your text, pressing Enter at the end of each line. When you have entered all your text, type a . (period) alone at the start of a line. This ends text input mode and returns to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer. The **i** subcommand differs from the **a** subcommand in the placement of text.

Changing Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets. You can use the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address1*,*address2*] **change** (**c**)
text
.

If you do not specify an *address*, the **c** subcommand replaces the current line with the text you type. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **c** subcommand replaces the addressed line (a single address) or a range of lines (two addresses separated by a comma).

Type your text, pressing Enter at the end of each line. When you have entered all your text, type a . (period) alone at the start of a line. This ends text input mode and returns to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer. The last input line becomes the current line.

Deleting Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets. You can use the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address1*,*address2*]**delete**(**d**) [*buffer*]

If you do not specify an *address*, the **d** subcommand deletes the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **d** subcommand deletes the addressed line (a single address) or a range of lines (two addresses separated by a comma). The line following the last deleted line becomes the current line.

If you specify a *buffer* by giving a letter from a to z, the edit editor saves the addressed lines in that buffer or, if the letter is uppercase, appends the lines to that buffer. You can put the deleted lines back into the buffer by using the **pu** subcommand.

Displaying Text and Finding Out the Current Line

In the following subcommands, *address* is optional. If you specify an *address*, you do not type the brackets. You can use either the full subcommand or its abbreviation, which is in

parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address1*,*address2*]**number** (**nu**)
: Displays the addressed line or lines preceded by its buffer line number. If you do not specify an *address*, the **nu** subcommand displays the current line and number. If you specify an *address*, the **nu** subcommand displays the addressed line (a single address) or a range of lines (two addresses separated by a comma). See addressing lines in a file for more information. The last line displayed becomes the current line.

[*address1*,*address2*]**print** (**p**)
: Displays the addressed line or lines. If you do not specify an *address*, the **p** subcommand displays the current line. If you specify an *address*, the **p** subcommand displays the addressed line (a single address) or a range of lines (two addresses separated by a comma). See addressing lines in a file for more information. The last line displayed becomes the current line.

[*address*]**=**
: Displays the line number of the addressed line. If you do not specify an *address*, displays the line number of the current line. See addressing lines in a file for more information.

[*address*]**z**
: If an *address* is not specified, displays a screen of text beginning with the current line. If an *address* is specified, displays a screen of text beginning with the addressed line. See addressing lines in a file for more information.

[*address*]**z–**
: If an *address* is not specified, displays a screen of text with the current line at the bottom. If an *address* is specified, displays a screen of text with the addressed line at the bottom. See addressing lines in a file for more information.

[*address*]**z.**
: If an *address* is not specified, displays a screen of text with the current line in the middle. If an *address* is specified, displays a screen of text with the addressed line in the middle. See addressing lines in a file for more information.

Making Global Changes

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets. You can use either the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address1*,*address2*]**global**/*pattern*/*subcommand–list* (**g**)
: Marks each of the addressed lines that match the *pattern*. Then the edit editor carries out the specified subcommands on each marked line. If you do not specify an *address*, the **g** subcommand works on the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **g** subcommand works on the addressed line (a single address) or a range of lines (two addresses separated by a comma).

  A single *subcommand* or the first *subcommand* in a subcommand list appears on same line as the **global** subcommand. The remaining subcommands must appear on separate lines, where each line (except the last) ends with a \ (backslash). The default subcommand is the **print** subcommand.

  The list can include the **append** subcommand, **insert** subcommand, and **change** subcommand, and their associated input. In this case, if the ending

period comes on the last line of the command list, you can omit it. The **undo** subcommand and the **global** subcommand itself, however, may not appear in the subcommand list.

Moving or Copying Text

In the following subcommand, *address1* and *address2 are* optional. If you specify an *address*, you do not type the brackets. You must specify *address3*. You can use either the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address1*,*address2*]**move** *address3* (**m**)

If you do not specify an *address*, the **m** subcommand moves the current line after the line specified by *address3*. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **m** subcommand moves the addressed line (a single address) or a range of lines (two addresses separated by a comma). The first of the moved lines becomes the current line.

[*address1*,*address2*]**yank** [*buffer*] (**ya**)

Copies the specified line or lines into *buffer* (specified by a single alpha character name **a** – **z**). You can use the **pu** command to put these lines into another file.

[*address*]**put** [*buffer*] (**pu**)

If you do not specify an *address*, the **pu** subcommand retrieves the contents of the specified *buffer* and places it after the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **pu** subcommand retrieves the contents of the specified *buffer* and places it after the addressed line. If you do not specify a *buffer*, the **pu** subcommand restores the last deleted or yanked text. Thus, you can use this subcommand together with the **delete** subcommand to move lines within a file or with the **yank** subcommand to duplicate lines between files.

Saving Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets. You can use either the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address1*,*address2*]**write** [*file*] (**w**)

If you do not specify an *address*, the **w** subcommand writes the entire contents of the buffer to the file specified by *file*. If you specify an *address*, the **w** subcommand writes the addressed line (a single address) or a range of lines (two addresses separated by a comma) to the *file* specified. See addressing lines in a file if you need information about addressing. The editor displays the number of lines and characters that it writes. If you do not specify a *file*, the editor uses the current file name. If *file* does not exist, the editor creates it.

Substituting Text

In the following subcommand, *address* is optional. If you specify an *address*, you do not type the brackets. You can use either the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

[*address1*,*address2*]**substitute**/*pattern*/*replacement*/ (**s**)
[*address1*,*address2*]**substitute**/*pattern*/*replacement*/**g**

Replaces on each addressed line the first instance of *pattern* with the text specified. If you add the **g** subcommand, the **s** subcommand replaces all instances of *pattern* on each addressed line. If you do not specify an *address*, the **s** subcommand works on the current line. You may need to find out the current line or specify an address if you are not in the correct position in the buffer. If you specify an *address*, the **s** subcommand works on the addressed line (a single address) or a range of lines (two addresses separated by a comma).

Undoing a Change

You can use either the full subcommand or its abbreviation, which is in parentheses. See edit subcommand syntax if you want information about the format of edit subcommands.

undo (**u**) Reverses the changes made in the buffer by the last buffer editing subcommand. Note that **global** subcommands are considered a single subcommand to an **undo** subcommand. You cannot undo a **write** subcommand or an **edit** command.

Manipulating Files with the edit Editor

The edit subcommands for manipulating files allow you to perform the following tasks:

Editing Additional Files

Displaying the Current File Name and Status

Changing the Name of the Current File

Saving a File after a System Crash.

edit Subcommands for Manipulating Files

In the following subcommands, you can use the full subcommand or its abbreviation, which is in parentheses.

Editing Additional Files

edit *File* (**e**) Begins an editing session on a new *File*. The editor first checks to see if the buffer has been modified (edited) since the last **write** subcommand. If it has, the editor command issues a warning and cancels the **:e** subcommand. Otherwise, it deletes the complete contents of the editor buffer, makes the named file the current file, and displays the new file name. After insuring that this file can be edited, the editor reads the file into its buffer. If the editor reads the file without error, it displays the number of lines and characters that it read. The last line read becomes the new current line.

next (**n**) Copies the next *File* in the command line argument list to the buffer for editing.

Displaying the Current File Name and Status

In the following subcommand, you can use the full subcommand or its abbreviation, which is in parentheses.

file (f) Displays the current file name along with the following information about it:

- Whether it has been modified since the last **write** subcommand.
- What the current line is.
- How many lines are in the buffer.
- What percentage of the way through the buffer the current line is.

Changing the Name of the Current File

file *File* Changes the name of the current file to the *File* specified. The **edit** command considers this file not edited.

Saving a File after a System Crash

preserve Saves the current editor buffer as though the system had just crashed. Use this command when a **write** subcommand has resulted in an error, and you do not know how to save your work. Use the **recover** subcommand to recover the file.

recover *File* Recovers the *File* specified from the system save area. Use this after a system crash, or after a **preserve** subcommand.

Ending and Exiting the edit Editor

You can use the full subcommand or its abbreviation, which is in parentheses.

quit (q) Ends the editing session after using a **write** subcommand. If you have modified the buffer and have not written the changes, the editor displays a warning message and does not end the editing session.

quit! (q!) Ends the editing session discarding the changes made to the buffer since the last **write** subcommand.

Addressing Lines in a File with the edit Editor

There are three types of edit addresses:

- Line number addresses

  The simplest way to address a line within a file is to use its line number. You can also address the current line by its symbolic name, . (period), and the last line by its symbolic name, $ (dollar sign). This is useful when addressing a range of lines. For example, `.,$print` displays the current line through the last line of the buffer.

- Relative position addresses

  An address that begins with *–Number* or *+Number* addresses a line the *Number* of lines specified before or after the current line, respectively. For example, `+8` addresses 8 lines past the current line. You can also address a line relative to the last line by using $ (the symbolic name for the last line). For example, `$–5` addresses the fifth line from the last.

- Pattern addresses

  You can address a line by searching through the buffer for a particular *pattern*. To search forward, use */pattern/*. To search backward, use *?pattern?*. The **edit** command searches forward or backward and stops at the first line containing the match for *pattern*. If necessary the search wraps around past the end or beginning of the buffer until it either finds a match or returns to the current line.

The following characters have special meanings in these search patterns:

^	Matches the beginning of a line.
$	Matches the end of a line.

Thus, you can use */^pattern/* to search for patterns at the beginning of a line, and */pattern$/* to search for patterns at the end of the line.

To specify a range of lines, separate two line numbers or *pattern*s with a comma or a semicolon (for example, `1,5` or `1;5`). In a range, the second address must refer to a line that follows the first addressed line in the range.

Index

Symbols

A

E

H

I

K

V

Reader's Comment Form

AIX Editing Concepts and Procedures for RISC System/6000

SC23–2212–00

Please use this form only to identify publication errors or to request changes in publications. Your comments assist us in improving our publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page	Comments

Please contact your IBM representative or your IBM–approved remarketer to request additional publications.

Please print

Date _____

Your Name _____

Company Name _____

Mailing Address _____

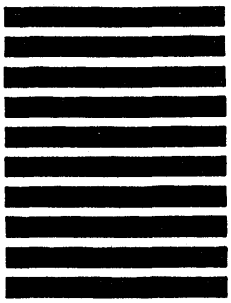
_____

_____

Phone No. ( ) _____
Area Code

No postage necessary if mailed in the U.S.A

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 997, Building 997
11400 Burnet Rd.
Austin, Texas 78758–3493

Fold

Fold

Cut or Fold Along Line

Fold and Tape

Please Do Not Staple

Fold and Tape

International Business Machines
Corporation
11400 Burnet Road
Austin, Texas 78758–3493

Printed in the
United States of America

SC23-2212-00

SC23-2212-00