Ex/Edit Command Summary (Version 2.0)

Ex and *edit* are text editors, used for creating and modifying files of text on the UNIX computer system. *Edit* is a variant of *ex* with features designed to make it less complicated to learn and use. In terms of command syntax and effect the editors are essentially identical, and this command summary applies to both.

The summary is meant as a quick reference for users already acquainted with *edit* or *ex*. Fuller explanations of the editors are available in the documents *Edit: A Tutorial* (a self-teaching introduction) and the *Ex Reference Manual* (the comprehensive reference source for both *edit* and *ex*). Both of these writeups are available in the Computing Services Library.

In the examples included with the summary, commands and text entered by the user are printed in **boldface** to distinguish them from responses printed by the computer.

The Editor Buffer

In order to perform its tasks the editor sets aside a temporary work space, called a *buffer*, separate from the user's permanent file. Before starting to work on an existing file the editor makes a copy of it in the buffer, leaving the original untouched. All editing changes are made to the buffer copy, which must then be written back to the permanent file in order to update the old version. The buffer disappears at the end of the editing session.

Editing: Command and Text Input Modes

During an editing session there are two usual modes of operation: *command* mode and *text input* mode. (This disregards, for the moment, *open* and *visual* modes, discussed below.) In command mode, the editor issues a colon prompt (:) to show that it is ready to accept and execute a command. In text input mode, on the other hand, there is no prompt and the editor merely accepts text to be added to the buffer. Text input mode is initiated by the commands *append*, *insert*, and *change*, and is terminated by typing a period as the first and only character on a line.

Line Numbers and Command Syntax

The editor keeps track of lines of text in the buffer by numbering them consecutively starting with 1 and renumbering as lines are added or deleted. At any given time the editor is positioned at one of these lines; this position is called the *current line*. Generally, commands that change the contents of the buffer print the new current line at the end of their execution.

Most commands can be preceded by one or two line-number addresses which indicate the lines to be affected. If one number is given the command operates on that line only; if two, on an inclusive range of lines. Commands that can take line-number prefixes also assume default prefixes if none are given. The default assumed by each command is designed to make it convenient to use in many instances without any line-number prefix. For the most part, a command used without a prefix operates on the current line, though exceptions to this rule should be noted. The *print* command by itself, for instance, causes one line, the current line, to be printed at the terminal.

The summary shows the number of line addresses that can be prefixed to each command as well as the defaults assumed if they are omitted. For example, *(.,.)* means that up to 2 line-numbers may be given, and that if none is given the command operates on the current line. (In the address prefix notation, ''.'' stands for the current line and ''$'' stands for the last line of the buffer.) If no such notation appears, no line-number prefix may be used.

Some commands take trailing information; only the more important instances of this are mentioned in the summary.

Open and Visual Modes

Besides command and text input modes, *ex* and *edit* provide on some CRT terminals other modes of editing, *open* and *visual*. In these modes the cursor can be moved to individual words or characters in a line. The commands then given are very different from the standard editor commands; most do not appear on the screen when typed. *An Introduction to Display Editing with Vi* provides a full discussion.

Special Characters

Some characters take on special meanings when used in context searches and in patterns given to the *substitute* command. For *edit*, these are ''^'' and ''$'', meaning the beginning and end of a line, respectively. *Ex* has the following additional special characters:

. & * [ ] ~

To use one of the special characters as its simple graphic representation rather than with its special meaning, precede it by a backslash (\). The backslash always has a special meaning.

Name	Abbr	Description	Examples
(.)**append**	**a**	Begins text input mode, adding lines to the buffer after the line specified. Appending continues until "." is typed alone at the beginning of a new line, followed by a carriage return. *0a* places lines at the beginning of the buffer.	:**a** **Three lines of text** **are added to the buffer** **after the current line.** **.** :
(.,.)**change**	**c**	Deletes indicated line(s) and initiates text input mode to replace them with new text which follows. New text is terminated the same way as with *append*.	:**5,6c** **Lines 5 and 6 are** **deleted and replaced by** **these three lines.** **.** :
(.,.)**copy** *addr*	**co**	Places a copy of the specified lines after the line indicated by *addr*. The example places a copy of lines 8 through 12, inclusive, after line 25.	:**8,12co 25** Last line copied is printed :
(.,.)**delete**	**d**	Removes lines from the buffer and prints the current line after the deletion.	:**13,15d** New current line is printed :
edit *file* **edit!** *file*	**e** **e!**	Clears the editor buffer and then copies into it the named *file*, which becomes the current file. This is a way of shifting to a different file without leaving the editor. The editor issues a warning message if this command is used before saving changes made to the file already in the buffer; using the form **e!** overrides this protective mechanism.	:**e ch10** No write since last change :**e! ch10** "ch10" 3 lines, 62 characters :
file *name*	**f**	If followed by a *name*, renames the current file to *name*. If used without *name*, prints the name of the current file.	:**f ch9** "ch9" [Modified] 3 lines ... :**f** "ch9" [Modified] 3 lines ... :
(1,$)**global** (1,$)**global!**	**g** **g!** or **v**	**global/***pattern***/***commands* Searches the entire buffer (unless a smaller range is specified by line-number prefixes) and executes *commands* on every line with an expression matching *pattern*. The second form, abbreviated either **g!** or **v**, executes *commands* on lines that *do not* contain the expression *pattern*.	:**g/nonsense/d** :
(.)**insert**	**i**	Inserts new lines of text immediately before the specified line. Differs from *append* only in that text is placed before, rather than after, the indicated line. In other words, **1i** has the same effect as **0a**.	:**1i** **These lines of text will** **be added prior to line 1.** **.** :
(.,.+1)**join**	**j**	Join lines together, adjusting white space (spaces and tabs) as necessary.	:**2,5j** Resulting line is printed :

Name	Abbr	Description	Examples
(.,.)**list**	**l**	Prints lines in a more unambiguous way than the *print* command does. The end of a line, for example, is marked with a “$”, and tabs printed as “ˆI”.	:**9l** This is line 9$:
(.,.)**move** *addr*	**m**	Moves the specified lines to a position after the line indicated by *addr*.	:**12,15m 25** New current line is printed :
(.,.)**number**	**nu**	Prints each line preceded by its buffer line number.	:**nu** 10 This is line 10 :
(.)**open**	**o**	Too involved to discuss here, but if you enter open mode accidentally, press the ESC key followed by **q** to get back into normal editor command mode. *Edit* is designed to prevent accidental use of the open command.	
preserve	**pre**	Saves a copy of the current buffer contents as though the system had just crashed. This is for use in an emergency when a *write* command has failed and you don’t know how else to save your work.†	:**preserve** File preserved. :
(.,.)**print**	**p**	Prints the text of line(s).	:**+2,+3p** The second and third lines after the current line :
quit **quit!**	**q** **q!**	Ends the editing session. You will receive a warning if you have changed the buffer since last writing its contents to the file. In this event you must either type **w** to write, or type **q!** to exit from the editor without saving your changes.	:**q** No write since last change :**q!** %
(.)**read** *file*	**r**	Places a copy of *file* in the buffer after the specified line. Address 0 is permissible and causes the copy of *file* to be placed at the beginning of the buffer. The *read* command does not erase any text already in the buffer. If no line number is specified, *file* is placed after the current line.	:**0r newfile** "newfile" 5 lines, 86 characters :
recover *file*	**rec**	Retrieves a copy of the editor buffer after a system crash, editor crash, phone line disconnection, or *preserve* command.	
(.,.)**substitute**	**s**	**substitute/***pattern***/***replacement***/** **substitute/***pattern***/***replacement***/gc** Replaces the first occurrence of *pattern* on a line with *replacement*. Including a **g** after the command changes all occurrences of *pattern* on the line. The **c** option allows the user to confirm each substitution before it is made; see the manual for details.	:**3p** Line 3 contains a misstake :**s/misstake/mistake/** Line 3 contains a mistake :

† You should seek assistance from a system administrator as soon as possible after saving a file with the *preserve* command, because the preserved copy of the file is saved in a directory used to store temporary files, and thus, the preserved copy may only be available for a short period of time.

Name	Abbr	Description	Examples
undo	**u**	Reverses the changes made in the buffer by the last buffer-editing command. Note that this example contains a notification about the number of lines affected.	:**1,15d** 15 lines deleted new line number 1 is printed :**u** 15 more lines in file ... old line number 1 is printed :
(1,$)**write** *file* (1,$)**write!** *file*	**w** **w!**	Copies data from the buffer onto a permanent file. If no *file* is named, the current filename is used. The file is automatically created if it does not yet exist. A response containing the number of lines and characters in the file indicates that the write has been completed successfully. The editor's built-in protections against overwriting existing files will in some circumstances inhibit a write. The form **w!** forces the write, confirming that an existing file is to be overwritten.	:**w** "file7" 64 lines, 1122 characters :**w file8** "file8" File exists ... :**w! file8** "file8" 64 lines, 1122 characters :
(.)**z** *count*	**z**	Prints a screen full of text starting with the line indicated; or, if *count* is specified, prints that number of lines. Variants of the *z* command are described in the manual.	
!*command*		Executes the remainder of the line after **!** as a UNIX command. The buffer is unchanged by this, and control is returned to the editor when the execution of *command* is complete.	:**!date** Fri Jun 9 12:15:11 PDT 1978 ! :
control-d		Prints the next *scroll* of text, normally half of a screen. See the manual for details of the *scroll* option.	
(.+1)<cr>		An address alone followed by a carriage return causes the line to be printed. A carriage return by itself prints the line following the current line.	:<cr> the line after the current line :
/*pattern***/**		Searches for the next line in which *pattern* occurs and prints it.	:**/This pattern/** This pattern next occurs here. :
//		Repeats the most recent search.	:**//** This pattern also occurs here. :
?*pattern***?**		Searches in the reverse direction for *pattern*.	
??		Repeats the most recent search, moving in the reverse direction through the buffer.	