Ex changes – Version 2.0 to 3.1

This update describes the new features and changes which have been made in converting from version 2.0 to 3.1 of *ex.* Each change is marked with the first version where it appeared. Versions 2.1 through 2.7 were implemented by Bill Joy; Mark Horton produced versions 2.8, 2.9 and 3.1 and is maintaining the current version.

Update to Ex Reference Manual

Command line options

2.1 Invoking *ex* via

    % ex −l

now sets the *lisp* and *showmatch* options. This is suitable for invocations from within *lisp*(1). If you don't like *showmatch* you can still use "ex −l" to get *lisp* set, just put the command "set noshowmatch" in your *.exrc* file.

3.1 Invoking *ex* with an argument −**w***n* sets the value of the *window* option before starting; this is particularly suitable when invoking *vi,* thus

    % vi −w5 ex2.0-3.1

edits the file with a 5 line initial window.

2.9 The text after a + on the command line is no longer limited to being a line number, but can be any single command. This generality is also available within the editor on *edit* and *next* commands (but no blanks are allowed in such commands.) A very useful form of this option is exemplified by

    % vi +/main more.c

Command addressing

2.9 The address form **%** is short for "1,$".

Commands

2.2 The editor now ignores a ":" in front of commands, so you can say ":wq" even in command mode.

2.8 The *global* command now does something sensible when you say

    g/pat/

printing all lines containing *pat;* before this printed the first line after each line containing *pat.* The trailing / may be omitted here.

3.1 New commands *map* and *unmap* have been added which are used with macros in *visual* mode. These are described below.

3.1 The *next* command now admits an argument of the form "+command" as described above.

3.1 The *substitute* command, given no arguments, now repeats the previous *substitute,* just as "&" does. This is easier to type.

2.8 The substititute command "s/str", omitting the delimiter on the regular expression, now deletes "str"; previously this was an error.

2.9 During pattern searches of a *tag* command, the editor uses *nomagic* mode; previously a funny, undocumented mode of searching was used.

3.1 The editor requires that the tag names in the *tags* file be sorted.

2.3 The command *P* is a synonym for *print.*

2.9 The default starting address for *z* is **.+1**. If *z* is followed by a number, then this number is remembered by setting the *scroll* option.

2.9 A command consisting of only two addresses, e.g. "1,10" now causes all the lines to be printed, rather than just the last line.

Options

2.8 *Autowrite* (which can be abbreviated *aw*) is an on/off option, off by default. If you set this option, then the editor will perform *write* commands if the current file is modified and you give a *next,* ^^ (in *visual), !* or *tag* commands, (and noticeably not before *edit* commands.) Note that there is an equivalent way to do the command with autowrite set without the write in each case: *edit, :e #, shell* and *tag!* do not *autowrite.*

3.1 A new option *edcompatible* causes the presence or absence of *g* and *c* suffixes on *substitute* commands to be remembered, and to be toggled by repeating the suffices. The suffix *r* makes the substitution be as in the ~ command instead of like *&.*

2.8 There is a new *hardtabs* option, which is numeric and defaults to 8. Changing this to, say, 4, tells *ex* that either you system expands tabs to every 4 spaces, or your terminal has hardware tabs set every 4 spaces.

3.1 There is a new boolean option *mapinput* which is described with the macro facility for *visual* below.

2.9 Whether *ex* prompts for commands now depends only on the setting of the *prompt* variable so that you can say "set prompt" inside *script*(1) and get *ex* to prompt.

Environment enquiries

3.1 *Ex* will now execute initial commands from the EXINIT environment variable rather than *.exrc* if it find such a variable.

2.9 *Ex* will read the terminal description from the TERMCAP environment variable if the description there is the one for the TERM in the environment. TERMCAP may still be a pathname (starting with a /; in that case this will be used as the termcap file rather than /etc/termcap, and the terminal description will be sought there.)

Vi Tutorial Update

Change in default option settings.

3.1 The default setting for the *magic* option is now *magic.* Thus the characters

. [* ~

are special in scanning patterns in *vi.* You should

set nomagic

in your *.exrc* if you don't use these regularly. This makes *vi* default like *ex.* In a related change, *beautify* is no longer the default for *vi.*

Line wrap around

2.4 The *w W b B e* and *E* operations in visual now wrap around line boundaries. Thus a sequence of enough **w** commands will get to any word after the current position in the file, and **b**'s will back up to any previous position. Thus these are more like the sentence operations *(* and *).* (You still can't back around line boundaries during inserts however.)

2.3 The */* and *?* searches now find the next or previous instance of the searched for string. Previously, they would not find strings on the current line. Thus you can move to the right on the current line by typing "/pref<ESC>" where "pref" is a prefix of the word you wish to move to, and delete to a following string "str" by doing "d/str<ESC>", whether it is on the same or a succeeding line. (Previously the command "d/pat/" deleted lines through the next line containing "pat". This can be accomplished now by the somewhat unusual command "d/pat/0", which is short for "d/pat/+0". The point is that whole lines are affected if the search patter only specifies a line, and using address arithmetic makes the pattern only specify a line.)

3.1 Arrow keys on terminals that send more than 1 character now work. Home up keys are supported as are the four directions. (Note that the HP 2621 will turn on function key labels, and even then you have to hold shift down to use the arrow keys. To avoid turning on the labels, and to give up the function keys, use terminal type 2621nl instead of 2621.)

Macros

3.1 A parameterless macro facility is included from visual. This facility lets you say that when you type a particular key, you really mean some longer sequence of keys. It is useful when you find yourself typing the same sequence of commands repeatedly.

Briefly, there are two flavors of macros:

a) Put the macro body in a buffer register, say x. Then type @x to invoke it. @ may be followed by another @ to repeat the last macro. This allows macros up to 512 chars.

b) Use the map command from command mode (typically in the *.exrc* file) as follows:

map *lhs rhs*

where *lhs* will be mapped to *rhs.* There are restrictions: *lhs* should be 1-keystroke (either 1 char or 1 function key) since it must be entered within 1 second. The *lhs* can be no longer than 10 chars, the *rhs* no longer than 100. To get space, tab, "|", or newline into *lhs* or *rhs,* escape them with ctrl V. (It may be necessary to escape the ctrl V with ctrl V if the map command is given from visual mode.) Spaces and tabs inside the *rhs* need not be escaped.

For example, to make the Q key write and exit the editor, you can do

:map Q :wq^VCR

which means that whenever you type 'Q', it will be as though you had typed the four characters :wqCR. The control V is needed because without it the return would end the colon command.

For 1 shot macros it is best to put the macro in a buffer register and map a key to '@r', since this will allow the macro to be edited.

Macros can be deleted with

unmap lhs

If the lhs of a macro is "#0" through "#9", this maps the particular function key instead of the 2 char # sequence, if the terminal has function keys. For terminals without function keys, the sequence #*x* means function key *x*, for any digit *x*. As a special case, on terminals without function keys, the #x sequence need not be typed within one second. The character # can be changed by using a macro in the usual way:

map ^V^I #

to use tab, for example. (This won't affect the map command, which still uses #, but just the invocation from visual mode.) The undo command will undo an entire macro call as a unit.

3.1 New commands in visual: ^Y and ^E. These scroll the screen up and down 1 line, respectively. They can be given counts, controlling the number of lines the screen is scrolled. They differ from ^U and ^D in that the cursor stays over the same line in the buffer it was over before rather than staying in the same place on the screen. (^Y on a dumb terminal with a full screen will redraw the screen moving the cursor up a few lines.) If you're looking for mnemonic value in the names, try this: Y is right next to U and E is right next to D.

Miscellaneous

3.1 In visual: '&' is a synonym for ':&<cr>'.

2.2 In input mode in open and visual ^V (like tenex) is now equivalent to ^Q (which is reminiscent of ITS) superquoting the next character.

2.8 The j, k, and l keys now move the cursor down, up, and right, respectively, in visual mode, as they used to do (and always did on some terminals). This is to avoid the creeping of these keys into the map descriptions of terminals and to compensate for the lack of arrow keys on some terminals.

2.5 The $ command now sets the column for future cursor motions to effective infinity. Thus a '$' followed by up/down cursor motions moves at the right margin of each line.

2.9 The way window sizes and scrolling commands are based on the options window and scroll has been rearranged. All command mode scrolling commands (z and ctrl D) are based on scroll: ^D moves scroll lines, z moves scroll*2 lines. Everything in visual (^D, ^U, ^F, ^B, z, window sizes in general) are based on the window option. The defaults are arranged so that everything seems as before, but on hardcopy terminals at 300 baud the default for scroll is 11 instead of 6.