Sedit Manual

T.W. Steiner

1995

THIS PROGRAM WAS WRITTEN BY THOMAS W. STEINER.
COPYRIGHT 1992 - 1995, ALL RIGHTS RESERVED.
THERE IS NO WARRANTY OF ANY KIND. USE AT YOUR OWN RISK.
THIS IS NOT FREEWARE. TO REGISTER YOUR COPY SEND $25 TO
T.W. Steiner, 312 - 1230 Haro St.
Vancouver, BC, V6E - 4J9, Canada
e-mail steiner@sfu.ca

1 Introduction

The sedit editor is a fully featured professional text and binary editor. The editor is modeled after the E PC editor. This editor is however a functional superset of E with many useful additional features (except that there is no REXX support). The OS/2 EPM editor is another editor with the same ancestors. Users of E or EPM will quickly feel at home with this editor since basic operation is quite similar.

2 Highlights

Highlights of this editor are as follows:

- Complete OS/2 PM applications with all editor commands available either from drop down menus or by using CTRL and ALT key combinations. Appearance and operation of this editor are highly configurable.
- Edits text files of arbitrary size and line length.
- Edits binary files in either hex or in a special alphanumeric coded form that allows easy identification of strings in a binary file.
- Unlimited undo of previous changes and corresponding step for step redo. Lines can be restored to their previous value even if subsequent edits on other lines have been made that are to be kept.
- Column editing features to allow movement of columns of text. This allows easy change of indentation for example and is extremely useful for manipulating columns of numbers. Columns can be moved, copied, deleted, overlaid and filled.
- Bracket matching of all sorts of brackets and C code comment delimiters.
- Run programs such as dir, grep or compilers from the command line. These run as a separate thread so you can continue to edit while they run. Output from programs started from the command line is piped into another file in the ring of loaded files. Files can be loaded from these read only output files by double clicking on any file names with a trailing number, if any, interpreted as a line number.
- Programming support. Run compilers from the command line and jump to file and line containing the error by double clicking on the file name. Opens on-line programming reference information by double clicking on key words.
- There is limited folding which displays either only the lines starting in the first column to quickly locate the start of functions, only the changed lines or only the lines containing the last search string.
- Changed lines optionally appear in a different colour from the unchanged text.
- A whole ring of files can be loaded up at once allowing painless copying of text between files. One particularly nice feature is the ability to load up a set of files in a ring and rapidly cycle through them using then CTRL N and CTRL P key combinations. Text can be copied between these files by using the block and line mark commands.
- This editor has very few built in hard coded limits. The number of files simultaneously loaded is limited only by available memory. The number of lines in a file or the length of a line is also limited only to the size of available memory or the size of a 4 byte integer whichever is smaller.
- The editor has very powerful search options which allow searching with any of the following modifiers singly or combined. Search up, Search marked block only, Ignore case, loop through all loaded files or interpret as regular expression. For search and replace a dialog allows the following; replace, skip, replace all, quit and undo last replace.

- There are all also some features to facilitate carrying out complex repetitive editing tasks. In particular sequences of key strokes can be saved and assigned to a function key for subsequent reuse. Furthermore, any command including function key macros can be repeated a number of times automatically by entering a command multiplier. Bound macros are saved between editing sessions.
- The editor is completely key configurable. A set of keystrokes can be mapped to an editor function simply by specifying the sequence in the auxiliary key map file. The combination of key mapping and key macros allow the editor to imitate the user interface of other editors or allow the construction of a custom user interface to the taste of the user.

3 Control keys

The editor is extremely easy to use and largely self explanatory. Text is input by straight typing anywhere on the screen. Cursor movement is controlled by the arrow keys and auxiliary keypad keys as per their standard definitions as well as accelerated movement with CTRL combinations. The cursor can also be positioned using the mouse by clicking with the left button at the desired cursor location. The cursor position control keys are as follows:

Cursor Motion Keys

left arrow	left one space
right arrow	right one space
up arrow	up one line
down arrow	down one line
end	end of line
home	beginning of line
PgDn	scroll down one screen
PgUp	scroll up one screen
Center Key	fast motion toggle
CTRL left	left one word
CTRL right	right one word
CTRL up	scroll up five lines
CTRL down	scroll down five lines
CTRL end	end of file
CTRL PgDn	end of file
CTRL home	beginning of file
CTRL PgUp	beginning of file

Editor Control Keys

CTRL A	set place mark
CTRL B	bracket match {, (, [, <, or /*
CTRL C	change case of character under cursor
CTRL D	delete character
DEL	delete character

CTRL E	delete to end of line
CTRL F	find next occurance of search string
CTRL H	destructive backspace
CTRL I	tab
CTRL J	join next line to current line
CTRL K	cut current line at cursor
CTRL L	return to previous set place mark
CTRL M	insert new line (also return key)
CTRL N	edit next file in ring
CTRL 0	switch to other window if screen split
CTRL P	edit previous file in ring
CTRL Q	quit without saving file
CTRL R	redo previous undo
CTRL S	swap current and next character
CTRL T	toggle insert/replace
CTRL U	undo changes to last modified line
CTRL V	view only lines starting in 1st column
CTRL W	write file to disk
CTRL X	delete line
CTRL Y	split edit window horizontally in two
CTRL BACK	delete line
CTRL Z	zap word
CTRL DEL	zap word
ALT A	alternate binary representation
ALT E	execute last macro
ALT J	join next line to current line
ALT K	enter next char literally
ALT P	position curs at previous location
ALT Q	restore a line from undo record
ALT R	re-flow paragraph, stops at blank line
ALT S	split line at cursor
ALT T	teach (end) new macro
ALT V	view only changed lines
ALT W	view lines containing search string
ALT X	enter control character
ALT Z	zero command multiplier
ALT 0-9	enter command multiplier

The following section provides more detailed information for a selection of the above commands that are perhaps not inherently obvious.

- The CTRL A command sets a place mark on the current line in the current file. Any number of place marks can be set. Hitting CTRL L will return the cursor to these place marks in the reverse order that they where set. To remove a place mark hit CTRL A again on the line with the place mark and it will be removed.

- The bracket match command CTRL B moves the cursor to the bracket matching the one under the cursor. If the cursor does not move it means that a matching bracket could not be found or that the cursor is not presently sitting on a bracket. The matchable characters are <, >, (,), {, }, [,] and the C style comment delimiter pairs /* and */.

- CTRL C changes the case of the letter under the cursor. If the case of a whole region is to be changed. Block mark the region (ALT B) and then use the change case option of the block fill command (ALT F).
- CTRL F finds the next occurance of the search string. The first occurance must be found be going to the command line with ESC and entering the desired search string after a '/' as in "/string". The search string can also be set using the search dialog from the menu bar.
- CTRL J, K, M. Unlike most editors hitting return (CTRL M) does not split the current line at the cursor. Instead a line is split at the cursor using CTRL K. This usage may take some getting use to but it is the same as used by the editor E and optionally by the EPM editor. CTRL J is used to join two lines together. Return of CTRL M inserts a new blank line below the current line but the current line is not split. If CTRL K type behaviour of return is desired this can be set up using the key re-mapping facility.
- CTRL N, P. If more than one file is currently in memory then CTRL N switches to the next one in the ring while CTRL P switches to the previous file.
- CTRL Q. Quits the current file which will be removed from memory and all changes will be lost. If changes have been made and the file has not been saved then you will be prompted for confirmation. It is possible to configure the editor to prompt for a confirmation of over writing an old version of the file.
- CTRL U, R. Undo the change to the last edited line. The number of undo levels is set in the configuration dialog and is essentially unlimited. By repeatedly hitting CTRL U all the changes made to a file since it was loaded can be undone. CTRL R re-does a previous undo. The changes to a file can only be undone in the order that they where made except that any line for which a line undo record exists can be restored using ALT Q. Only simple edits that only operate on a single line generate undo records of the type that can be undone using ALT Q. In some instances it may be desirable to turn off the undo record generation for example if column manipulations are to be done on very long columns since in that case the entire file will have to be duplicated may times for the undo records which will adversely affect performance. Undo record generation may be turned off by setting the undo limit to zero in the config dialog of the file menu.
- CTRL V ALT V ALT W. These commands toggle folding on and off. CTRL V causes only lines starting in the first column to be displayed. If the text is organized so that only section headings or sub-routine headings start in the first column then CTRL V can be used to rapidly locate a section in a file. To do this hit CTRL V then move cursor to desired sub-routine heading and hit CTRL V to expand again. ALT V and ALT W work similarly except they show only changed lines and lines containing the last search string respectively.
- CTRL W. Saves the current file to disk using the file name displayed on the command line. The file name can be changed by moving to the command line ESC and using the rename command. The file can also be saved with a different name using the "save as" menu item of the file menu.
- CTRL Y splits the editor window horizontally in to two windows separated by a second command line. Hitting CTRL Y again un splits the screen. Use CTRL O or the left mouse button to change the current window.
- ALT A selects the alternate binary representation. This key is only active for files loaded using the binary switch (-b). The default binary representation is hex with two hex characters per byte and 32 bytes per line. The alternate representation uses the following codes: Printable characters appear with a leading underscore and control characters with a leading $\wedge$. This second representation also has 32 bytes per line and is more useful for identifying text in binary files.
- ALT B,C,D,F,L,M,N,O,U,Y are used with marked regions see section4.

- ALT E executes the last defined macro see section 5.
- ALT K interprets the next character as a literal without re-mapping. This key may be useful for enter characters that have been mapped onto editor commands. This should not be needed with the default configuration since only CTRL and ALT keys are used to control the editor.
- The ALT P command is useful if the cursor was accidentally moved away from the region of interest with a CTRL Home or search for example. In these situations ALT P will restore the cursor to its previous location. This action is similar to using place marks except that unlike place marks these are automatically set by any command that causes the cursor to jump by more than one line. Only one previous location is remembered though.
- ALT T start and end the definition of a new macro see section 5.
- ALT V, ALT W toggles folding on and off. See discussion for CTRL V above.
- ALT X queries the user for the 3 digit hex code of the character to be inserted. ALT X can thus be used to enter characters for which there is no keyboard key such as the characters above 127 in the ASCII table.
- ALT 0-9, Z. These keys are used to enter a command multiplier to be used on the next command. Entering a multiplier has the same effect as entering the following keystroke n times. ALT Z resets the multiplier to zero.

4 Block manipulations

Block mark and line mark are somewhat different. Use line mark for moving, copying or deleting one or more whole lines and block mark for moving, copying, deleting, filling or overlaying columns of text. Note also that a block marked region can also be moved to the left with CTRL ∧ and to the right with CTRL _ but the behaviour is different than if the ALT arrow keys are used. ALT left and ALT right (or CTRL _ and CTRL ∧) are meaningless for line marked regions. Block marked regions can furthermore be filled with a character, a string or incrementing or decrementing numbers. Marked blocks are also used to constrain a search and indicate a target region for column math. ALT N marking is the industry standard marking useful for text but not of much use for programming. Regions can also be marked by dragging the mouse with one of the three buttons down. The type of region mark created depends on the settings in the "config" notebook accessed from the "file" menu.

Block Manipulation Keys

ALT B	mark column block start, end
ALT C	copy marked block
ALT D	delete marked block
ALT F	fill B marked block with char, change case or inc/dec
ALT L	mark LINE(s) start, end
ALT M	move marked block
ALT O	overlay B marked block
ALT U	un mark block
ALT Y	yank back deleted block
CTRL_	shift B marked block right
CTRL∧	shift B marked block left

5 Macros

There is also a macro capability for repetitive, complex editing tasks. The editor is taught a macro by hitting the ALT T combination. After this all subsequent keystrokes are stored as the macro definition until ALT T is hit again. There is a 128 keystroke limit on the length of a macro. A macro can consist of any keystrokes except ALT T and ALT E. At the end of the macro definition the macro can be bound to a function key by hitting the desired function key. Available function keys are F1 - F12 as well as shift F1 - F12. A bound macro is subsequently executed by hitting that function key. The predefined macro for that key will of course be lost. The last learned macro, whether bound or unbound, can be executed by hitting ALT E. If the macro involves a search and another item is not found the rest of the macro is not executed. If macros have been bound to function keys they will be written to the configuration file splot.cfg which will be created in the current directory so that the macros will be preserved in between editing sessions. The editor.cfg file as received contains a number of predefined macros that emulate the E editor usage of the function keys. For example F1 is predefined as help, F2 as save (CTRL W), F3 is predefined as quit (CTRL Q) and F4 is save and quit (CTRL W CTRL Q). These may of course be over written.

A repeat factor for a keystroke can also be entered by prefixing the command with a multiplier which is input using the ALT number keys. The multiplier can be reset to zero during input using ALT Z. This feature also facilitates repetitive editing as an operation can be repeated several times. This is particularly powerful in conjunction with macros.

6 Command Line

Lastly, there are a small number of commands which require typed input on the command line. The command line is reached by hitting ESC. This places the cursor at the beginning of the command line which is on the last row and ordinarily displays some status information. After ESC however text can be input here for the following functions:

Command Line Operations (requires text input so terminate line with a return)

ESC	goto or leave command line
	Change current line
number	goto line number n
+number	down n lines
-number	up n lines

Change current column offset

@number	start at column n
@+number	scroll left n columns
@-number	scroll right n columns

Search and Replace.
find string str (/ actually any punct)
/str[/-bclr] return
replace str1 with str2
c/str1/str2[/-bcl] return

The optional search suffixes are

/-	search up (down is default)
/b	search marked block only
/c	ignore case
/l	loop through all files
/r	interpret as regular expression

For regular expression the following symbols
have special meaning
∧ start of line
$ end of line
. any character
quote next character
* match zero or more times
+ match one or more times

[aei0-9]	match a,e,i and 0 through 9
[∧aei0-9]	match anything but a,e,i and 0 through 9
( )	sub expression
a(ab)*b	matches ab aabb aababb aabababb etc.

Add other files to ring
e name1,name2,...[-r]
The optional edit suffixes are

-r	read only
-b	binary mode

display directory and choose one to edit
    e [*.*]
activate this help file
    ?
insert named file at current location
    m name or merge name
rename current file to name
    r name or rename name
show and modify editor configuration
    cfg or configure
exit without saving changes
    quit
fill marked block with result of column math
    col num1 [+-*/] num2
fill marked block with string
    fill string
sort lines using field number col as key
    sort col
convert decimal num to hex
    hex num
convert hex num to decimal
    dec num
previous command line entered
    up arrow
next command line entered
    down arrow
send command to operating system

	os command

also any unrecognized string

In order to fill a B marked block with a string be sure that the width of the block in characters matches the width of the fill string otherwise the fill word will wrap around to the next line or repeat. For filling a marked block with the result of column math also be sure to define a sufficiently wide box otherwise the results will be truncated.

7 Configuration Files

The startup file editor.cfg preserves the values selected from the configuration menu item between sessions. You can have multiple editor.cfg files in different directories since the current directory is searched before using the default editor.cfg stored in the same directory as the executable. This is useful since a configuration suitable for text (auto wrap on, auto indent off) is not ideal for programming (auto wrap off, auto indent on) thus text specific editor.cfg files may be kept in text directories and programming specific ones in program directories. The editor.cfg file is plain ASCII and can thus be edited with a text editor as an alternative means of changing the settings or the macro definitions. If changes have been made to the configuration the editor will create a new editor.cfg in the current directory. The default editor.cfg is not affected unless the editor was started from the directory where the executable is stored.

The macros are defined in plain text and thus in order to accommodate control and cursor keys as well each macro keystroke is defined by a pair of letters using the following scheme. Starting in the first column of a definition must appear the key word or function key identifying the macro. This is followed by the macro body. Function keys may not appear in macro bodies. Keystroke and are encoded as follows with implied sequences in ASCII order.

Ordinary characters	prepend with _	
Control Keys	prepend with ∧	∧M Return, ∧[esc
Alt Keys	prepend with %	
Cursor keys	prepend with #	#. Del, #0 Ins - #9 PgUp
Ctrl Cursor Keys	prepend with #	#N Del, #P Ins - #Y PgUp
Alt Cursor Keys	prepend with #	#n Del, #p Ins - #y PgUp
Function Keys	prepend with $	$1 - $<
Shift Function Keys	prepend with $	$= - $H

This same key naming scheme is used in the key re-mapping file editkey.map. The first entry in a line of the editkey.map is the un-mapped key name followed by one or more key names describing the new key sequence for that command. For example the line "∧? ∧Y" maps CTRL Y onto the delete line command. The line %A ∧[_A maps "ESC A" onto the ALT A function. The included editkey.map file maps all ALT letter combinations to ESC letter sequences and ESC to ESC ESC. This is used in the UNIX dumb terminal version since these terminal don't have an ALT key. In order to activate key mapping the key_map parameter in the editor.cfg file must be set to 1. If the exclusive_map value is also set to 1 then the editor will only respond to mapped keys. This thus requires that all keys necessary for basic operation be mapped. These parameter values are not accessible from the configuration notebook and must be set by editing the editor.cfg file. Note that the new key mappings will be reflected in the editor help window but not unfortunately in the accelerator tags of the menu items.