User Guide to the CSU Editor Program (csued)

CS-200, Fall 1999 Project

Documentation Updated for Use in CS514 Fall 2000

1 Introduction

The CS200 Editor program is a line editor. A line editor is a program for modifying files one line at a time. A line editor is much simpler than the screen editors (such as VI, Emacs, Pico) that we normally use, and naturally, predated the screen editor. In fact, line editors were designed for input devices, such as TTYs, where one could only type in one line at a time.

This document describes what the editor should do and partially describes the code. Use this document as a requirements specification when testing the editor.

2 Editor Commands

The editor (called `csued`) will allow a user to edit one file at a time. After the editor is invoked on a file, the user can type any of a set of commands for modifying the contents of the file. Just as in a full-screen editor, the user continues to enter commands (thus changing the file) until they are satisfied with the results and then write the file and quit/exit.

This section describes the commands available, both in terms of their appearance and meaning to the user and in terms of how the program interprets the commands. Knowing how the program interprets the commands will tell you exactly what form the commands can take. The last part of this section describes two "advanced" commands, index and keyword, that are a bit different from other editor commands you may know.

2.1 Editor Command Documentation

The following sections document all of the commands in `csued`. Sections 2.2, 2.3 & 2.4 would typically be for user-type documentation available to users in their user's guide, while the latter sections (2.5, etc.) would be for internal programming documentation to be used by system designers or programmers (and would NOT be made available to users).

2.2 Command & Edit File Specifications

There are a few important things to understand about the edit file and editor commands:

1. The name of the edit command itself may be upper or lower case.
2. *At least one blank/space* must separate the edit command from its arguments and the arguments from other arguments. However, blanks/spaces before the edit command or after the last argument are allowed.
3. The file to be edited may contain any number of lines of any length.
4. The editor operates on edit file lines specified by a *current line number* (CLN). The first line in the edit file is 1 (one) and the remaining lines are consecutively numbered from there. With few exceptions, a command operates on lines starting from where CLN is positioned and leaves CLN at the last line operated on. Hence, all references to file lines are *relative* to the CLN!
5. A command that attempts to process more lines than are on a file will process the remaining lines on the file and will leave CLN positioned at the last line in the file. This also implies that "wrapping" from the bottom to the top of the edit file (or vice versa) is disallowed!
6. *Forward/Down* means to higher file line numbers and *Back/Up* means the opposite.

2.3 Table of Commands

In the table below, the CLN column indicates whether CLN is changed by the operation of a particular command. Remember, if a command alters CLN, then CLN is positioned to the last line the command operated on (or the last line of the file if more lines were indicated than existed).

Also note that the default column positions for the W command are 1 & 80.

Command Description	Command Form	CLN	Notes
Invocation	java csued *file*	√	CLN set to 1
Quit (& update file)	Q		
Exit (& NO file update)	X		
Top	T	√	Position CLN to line 1
End	E	√	Position CLN to last line
Next	N *nl*	√	Move forward *nl* lines
Back	B *nl*	√	Move backwards *nl* lines
Where	W		Print/Show CLN value
Count	C		Print/Show # file lines
List lines	L *nl*	√	CLN at last listed
Show lines	S *nl*		CLN remain unchanged
Delete lines	D *nl*	√	
Add/Insert text lines	A	√	Quit w/return on new line
Find string	F *nl str*	√	str delimited
Replace string	R *nl oldstr newstr*	√	oldstr & newstr delimited
Yank (copy) lines	Y *nl*	√	Copy lines to buffer
Yank (delete) lines	Z *nl*	√	Delete lines to buffer
Put buffer lines	P	√	Puts buffer lines after CLN
Index keywords	I		
List keyword lines	K *keyword*		keyword delimited
Order/Sort lines	O *nl*	√	Sorts L-H lines
Margin/Window	M *c1 c2*		Sets column positions, c1 $\leq$ c2
Help information	H {*cmd*}		General help or help on cmd

2.4 Command Explanations

csued — Initiates the editor. If the edit file does not exist then the edit file is created.

Q — Terminates editing and updates the edit file with the editing changes.

X — Terminates editing and does NOT update the edit file. The changes are ignored.

T — Positions CLN to the top of the file (to line 1).

E — Positions CLN to the end/bottom of the file (to last line).

N — Moves CLN number of lines forward/down the file (from CLN).

B — Moves CLN number of lines backwards/up the file (from CLN).

W — Prints "At Edit File Line x" where x is the value of CLN.

C — Prints "Total Edit File Lines: x" where x is number of lines in the edit file.

L — Lists/Prints number of file lines starting at CLN (and leaves CLN at last line printed).

S — Shows/Prints number of file lines starting at CLN (and DOES NOT change CLN!).

D — Deletes the number of file lines starting at CLN and leaves CLN positioned to the line after the last line deleted.

A — Adds lines (typed/entered) to the edit file AFTER the CLN line number. Enter/type a null line (i.e. just hit return on a new line) by itself to terminate entry. CLN is positioned to the last line entered.

F — Finds and prints the line, or lines staring at CLN, that contain one or more occurrences of the string. The first and last characters of the string must be the same to delimit the string and are not considered during string matching. CLN is positioned to the last line a match was attempted.

R — Finds and replaces all occurrences of the old string with the new string starting at CLN. Any line that is changed is printed. The first and last characters of each string must be the same to delimit the string and are not considered during string matching or replacement. Note: If the old string was "xBBx", the new string was "yBy", the line being changed was "BBBBBB", then the updated line would contain "BBB".

Y — Yanks and COPIES the lines starting at CLN to the internal line buffer. Y replaces the previous contents of the internal line buffer with the newly yanked line(s). CLN is positioned to the last line yanked.

Z — Yanks and DELETES the lines starting at CLN to the internal line buffer. Z replaces the previous contents of the internal line buffer with the newly deleted line(s). CLN is positioned to the line after the last line deleted.

P — Puts the entire contents of the internal line buffer after the CLN and leaves CLN positioned to the last line put. The internal line buffer remains unaltered after the put. If the internal line buffer is empty then P has no effect on the edit file.

I — Indexes the keywords at the top of the edit file (i.e. those lines appearing at the top of the file beginning with the character ''. An internal table is created with edit file line numbers of the edit file lines that contain each keyword. Warning: if other commands are used to change the file, then this command should be used again to re-index the file. This command does not affect CLN.

K — Keyword prints the line numbers from the keyword table created by the I(ndex) command for the specified keyword. The first and last characters of the keyword must be the same to delimit the string and are not considered part of the keyword. If I has not yet been issued before K, then K will simply report that the keyword was not found. This command does not affect CLN.

O — Orders/Sorts the lines L-H lexicographically starting at the CLN and leaves CLN positioned to the last line sorted. Note that trailing blanks at the end of lines does affect sorting.

M — Sets column margins/window (default starting values are 1 and 80) for use with the F, R & O commands. That is, these commands will search/sort only within the defined margins/window. This command does not affect CLN.

H — By itself provides a short list of the editor comamnds and with an argument (which must be the letter of a valid command) provides a short description of the command.

2.5 BNF Command Form

The editor must parse the command line after the user types it in and hits the return/enter key. Parsing is the process of taking a stream of characters (i.e., a string) and removing from it the salient symbols/tokens. So if the user types "F 3 xyz", the parser must determine that the user wishes to execute the find string command ("F") with the arguments "3" for the number of lines to search and xyz for the search string (disregarding the required and/or extra blanks in the command string).

We use a notation called *BNF* (for Backus-Naur Form) to describe how to translate an arbitrary string into a set of commands and arguments. We will use a subset of this notation to describe the form of commands for this assignment. To interpret the BNF for the editor commands, you need a bit of information. In particular, the following are the translations for symbols in the BNF:

Symbol	What It Means
<bl>	0 or more blanks
<ws>	1 or more blanks
<d>	any one character, the delimiter
<string>	any sequence of characters NOT CONTAINING <d>
<nl>	integer number containing no more than 7 digits AFTER the leading zeros have been removed (so it can contain many 0's at the front, which will be disregarded)
<cmd>	one character that must be one of the editor commands
{...}	whatever is within the curly braces is OPTIONAL

So for example, if you see " <bl> B <ws> <nl> <bl>", then all of the following are legal strings for that BNF: " B 5" or "B 5" or " B 1"; but other strings, such as "XB 9" or "B 1 0"or "B4", are illegal.

The most complicated of the commands, as described by the BNF in the following table, is replace string, which can be read as: *The command "R" can be preceded by any number of blanks, followed by at least one blank space and then a line number followed by at least one blank space and then a string with a reserved character at its beginning and end followed by at least one blank space and then a string with a reserved character at its beginning and end followed by any number of (optional) blanks.* For example, a legal replace string command is "R 5 /a string/ #the string#".

Given this information about how to "read" BNF, the commands for the editor are described in the following Table. The table lists every command along with its BNF description.

2.6 BNF Table of Commands

Command	BNF Command Form
Invocation	<bl> csued <ws> <string> <bl>
Quit (w/update)	<bl> Q <bl>
Exit (NO update)	<bl> X <bl>
Top	<bl> T <bl>
End	<bl> E <bl>
Next	<bl> N <ws> <nl> <bl>
Back	<bl> B <ws> <nl> <bl>
Where	<bl> W <bl>
Count	<bl> C <bl>
List lines	<bl> L <ws> <nl> <bl>
Show lines	<bl> S <ws> <nl> <bl>
Delete lines	<bl> D <ws> <nl> <bl>
Add lines	<bl> A <bl>
Find string	<bl> F <ws> <nl> <ws> <d> <string> <d> <bl>
Replace string	<bl> R <ws> <nl> <ws> <d> <string> <d> <ws> <d> <string> <d> <bl>
Yank/copy lines	<bl> Y <ws> <nl> <bl>
Yank/delete lines	<bl> Z <ws> <nl> <bl>
Put yanked lines	<bl> P <bl>
Index keywords	<bl> I <bl>
List keyword lines	<bl> K <ws> <d> <string> <d> <bl>
Order/Sort lines	<bl> O <ws> <nl> <bl>
Margin/Window	<bl> M <ws> <nl> <ws> <nl> <bl>
Help information	<bl> H <ws> {<cmd>} <bl>

2.7 Advanced Commands: Index and Keyword

Keywords are considered part of the file. That is, the user may have previously entered keywords on the edit file and/or may do so while editing the edit file. Also and therefore, the user may add or delete keyword lines or modify the keywords themselves while editing.

Each keyword must be on a line by itself and each keyword line must start with an at-sign (i.e. '`@`') in the first character position on the line. Furthermore, all keyword lines MUST be the first lines on the file (i.e. the end of the keyword list is the first line without an at sign in column one).

The keyword specified on a keyword line is considered to start with the first non-blank character after the at-sign and end with the last non-blank character on the line. Hence, a line of " dog " contains the keyword "dog" and a line of " dog & cat " contains the ONE keyword "dog & cat". Obviously, keywords may contain blanks.

When the user issues the I command, the edit file is scanned for all keywords (including the lines that contain the keywords themselves) and an internal table is created. This table contains a list for each keyword. Each list contains the line numbers of those lines that contained one or more occurrences of that keyword.

Obviously, if the user issues the I command and then proceeds to make more editing changes to the edit file, the internal keyword table may no longer be valid. This does not present a programming problem but the user should be aware that the internal table may become stale/obsolete.

The K command is used to cause the line numbers from the internal keyword table to be printed for the keyword specified by the user. If the K command is used before the I command (i.e. no internal keyword table) then K should simply report that the keyword was not found.

2.8 Advanced Commands: Margin (& Find, Replace, Order)

The M/margin command sets column positions for margins (i.e. a window or a vertical file "strip" in the file). Using this command, whose defaults are 1 & 80, affects ONLY the operation of the F/find, R/replace and O/order/sort commands.

Once the window is set, then the F/find command will only find the string within the defined margins/window. Similarly, the R/replace command will only find (for replacement) strings that are within the margin/window even though the replacment may expand past the margin/window. Finally the O/order/sort command will only sort on the characters in each line within the margin/window.

Importantly, note that a string MUST appear FULLY WITHIN the margins/window to be a candidate for a string match (on Find or Replace) or for sorting.

3 Editor Program

One of the most important Classes is the FileBuffer class which is used to provide the actual interface to the edit file and to allow operations on the edit file (i.e. so each edit command is NOT directly accessing the edit file NOR directly accessing the data structure containing the file lines!!).

This Class implements five basic operations: `GetLine`, `PutLine`, `AddLine`, `DeleteLine`, and `NumLines`. The GetLine operation retrieves a particular line from the file buffer, PutLine puts a line into the file line buffer at the specified line number, AddLine adds a new line into the buffer, DeleteLine removes a specified line from the buffer, and NumLines returns the number of lines currently in the file buffer.

As you can see, this Class allows for easy access to file lines, simplifies implementing the edit commands that operate on the file lines, and makes it easy to change the data structure that actually contains the file lines!

Similarly, we use other classes in the same fashion to provide interfaces to the data and operations needed for specific types of things like parsing, file line operations, etc. etc.