SOFTWARE

PRACTICE & EXPERIENCE

SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 1, 73–81 (1971)

A Design for a Text Editor

S. R. BOURNE
Computer Laboratory, University of Cambridge, England

SUMMARY

This paper describes a program for editing text on the Titan (prototype ICL Atlas II) computer at Cambridge University.

The program is an essential part of the multiple-access system that is in operation on that computer, although it is also used non-interactively through the more conventional job-shop system. The program, EDIT, has also been implemented on an IBM 1130 and an IBM 360/65.

Among the facilities provided by this program are the basic operations of text replacement, insertion and deletion.

Editing operations may be specified either by line number or by reference to a particular context.

The program is a compromise between a low cost, high-speed system and the flexible multi-function editor that is demanded by users from all departments of the University.

KEY WORDS Editing Titan, multiple-access EDIT

THE DESIGN OF AN EDITOR

The differences between an editing program, a macrogenerator and a string manipulation language like SNOBOL[1] are in principle quite small. They all enable the user to specify textual alterations to be made to a document and in some cases their functions overlap (e.g. replace a given string of text by another throughout a document).

The editor described in this paper is designed specifically to simplify program correction and other forms of *ad hoc* text amendment. It is not a device to be used for making systematic or elaborate textual transformations automatically. Text replacement, insertion and deletion are the three main functions provided and these are usually specified individually for each correction to be made. In practice, the editor is also used interactively to scan documents without making any changes.

The editor replaces the established practice of re-punching cards and copying paper-tape. Apart from the convenience provided by an on-line editor, the peripheral handling is reduced and the process is more reliable. Before an amendment can be made it is necessary for the user to specify the precise region of text to be changed. One way to achieve this is by reference to the position of the text in the document (e.g. line numbers), an alternative being to locate the text itself. Two ways in which the position of a piece of text could be identified by line number are:

(1) The number of each line can be computed by the editor as it works its way through the source document. At no time is the line number stored as part of the source document although line numbers could be appended when the contents of the document are printed.

Received 1 September 1970

(2) As a separate operation, the lines of a document can be numbered and these numbers stored with each line of text. When text is edited the numbers would be regarded as part of the source text and re-assignment of line numbers would not automatically follow when an insertion or deletion is made.

Page and chapter numbers may also be introduced and this would be useful for users wishing to edit essentially textual documents such as books or documentation. For editing the character files of programs, however, such a facility does not increase the power of the editor significantly. It may also mean that the editor would have to include pagination facilities and at Cambridge these are already provided by a separate program.[2]

In order to identify a point in the source document by context a search is made for a particular character string and its position is used as a basis for subsequent operations. This method can be more expensive in central processor time than counting lines, although the cost may be reduced if we know that the required string occurs at the start of a line. Another problem that arises when context is used as a means of identification is that the context given may not identify the required point uniquely. There will frequently be more than one occurrence of the same text in the source document. With an interactive on-line editor the risk of errors passing undetected is much reduced since the user can verify the result of a context search before subsequent action is taken. When used off-line errors cannot be corrected as they arise, so that more care is needed if the editor is being used in this way.

The editing program described here can be used interactively from an on-line console or it may be used off-line *via* the job-shop system. It employs both position and context as possible means of identifying the part of the source text to be altered. Commonly both forms of identification are used in conjunction. The editor computes line numbers as the source text is scanned but does not store these numbers as part of the document.

INPUT AND OUTPUT

Information is transferred to and from the editing program *via input* and *output streams* that are administered entirely by the Titan operating system (the *Supervisor*). An input stream may be linked either to a file on disk or to data from punched cards or paper-tape. All data from input peripherals is transferred onto the disk by the Supervisor and at this time the characters are translated into a standard internal code. Subsequently all data is buffered by the Supervisor and is handled by programs either one character or one line at a time.[3] When an output stream is complete, the editor is able to send it to any peripheral device or to a file. Alternatively, it is possible to convert an output stream into an input stream so that it may be used for further processing. Thus it is possible to use an output stream as a buffer for use later either by the editor or by a subsequent process. Another feature of the Supervisor is that an input stream may be read any number of times. At Cambridge, paper-tape is a more common input medium than cards and the tape preparation equipment provides both 'backspace' and 'erase' that appear as symbols on the punched tape. To handle this form of input EDIT provides a cleaning-up operation during which it forms a *line image* that corresponds to the appearance of the printed line. Standard tab settings are used and erased characters are removed in accordance with established conventions. It is also convenient to leave the line in a standard form so that redundant backspaces are removed and overprinted characters are arranged in a unique order. These line image conventions are oriented towards free format input although they could be amended to deal with the fixed format input that is required by some compilers.

Fixed format, that is commonly associated with card input, does present some difficulties when line imaging. In EDIT the onus is on the user to preserve the format of the document, should this be necessary.

Two input streams are presented to the editor: one contains the document to be edited, the source stream, and the other contains the editing commands. Since off-line data is presented to the computer on paper-tape, line imaging can be requested for the source document. This is usually only necessary when it is first read in, and, provided that subsequent changes are line imaged, the operation need never be repeated. The editing commands may also be line imaged. For on-line work Teletypes are used as the input device, and since the operating system provides facilities for making corrections to the line as it is being typed, there is normally no need to line image. However, the user may request line imaging should it be required.

EDITING COMMANDS

Commands to the editor are obeyed interpretively, that is, a command is read from the command stream and is carried out before another command is read. A consequence of this is that commands that refer to line numbers in the document must occur in ascending order of line number. An alternative strategy would be to read in and sort all the editing commands before applying them to the source document. The editing commands are not sorted for reasons of economy at run-time; another system is available that does provide this facility.[4] Thus EDIT maintains a pointer to a line within the document and will fault any references to earlier parts of the document.

The basic editing commands that are available in EDIT are the **Replace**, **Insert** and **Delete** commands. All commands are abbreviated to their first letter to reduce typing and since most commands are mnemonic this has proved in practice to be generally acceptable.

The replace command is typed as follows:

```
R  12  13
⟨lines of
new
material⟩
Z
```

and the effect of this is to replace lines 12 and 13 with the new material. This material may extend over several lines and is terminated by the character Z on a line by itself. The insert command is written:

```
I  168
⟨new
material⟩
Z
```

and will insert new material *before* line 168 of the document.

The delete command

```
D  40  44
```

will delete lines 40 to 44 inclusive from the source document.

Numbering the lines of a document, as it is read in, has the disadvantage that the line numbers in the edited document may be different from those in the original. For short documents a new listing may readily be obtained and then it is convenient to re-edit using

the new line numbers. For longer documents, or for remote users, it may not be convenient to produce a new listing after each use of EDIT. One way of avoiding this difficulty is to file the editing commands so that the current version of the source document is the result of applying the filed editing commands to the original source document.

Successive versions of the edited source document are then obtained by updating the file of editing commands and when a sufficiently large body of editing commands have been accumulated a new source document and a new listing are obtained.

The editor itself may be used to edit the file of editing commands. One minor difficulty that arises is the insertion of the letter Z that is used to terminate an Insert or Replace command. For this reason, a command is provided that allows an alternative letter to be used.

For documents that consist mainly of short lines, the above facilities are probably sufficient, but for long lines of text it is desirable to replace parts of text within the line in order to avoid re-typing a large proportion of the original material. These commands will be described in the next section.

CONTEXT COMMANDS

The context commands available in EDIT operate on a particular line of the document called the *current line*. The user can **Move** to line 94 and make it the current line using the command M 94. Since the teletype carriage is 72 characters wide, each current line is divided into 72 character *part lines*, and the N command will select either the **Next** line or the next part line as the current line. Two further commands are provided that select the current line and these are written +n and −n. The command +n will move forward n lines from the current line while the command −n will delete the next n lines of the document. For effective use of the commands M, + and −, a listing is needed. If no listing is available, then two commands that find the next occurrence of a string of text may be used. The **Locate** command, written L ϕ string of text ϕ, locates the next line that contains the string, while the **Find** command, written F ϕ string of text ϕ, will find the next line of the document that starts with the given string. In the above definitions ϕ denotes a character chosen by the user to delimit the string of text and as such it should not occur in this string. In particular a 'space' may be used as a delimiter; for example if the string to be located is +41 one may type L +41.

When the current line is located using the above commands, the editor will respond by typing the line number followed by the line. If only part of the line has been read, the line number is followed by the letter 'P'. Once a current line has been selected, the command E ϕ string A ϕ string B ϕ will **Exchange** string B with the first occurrence of string A in this line. To delete a string of characters in the line, the user would type E ϕ string ϕ ϕ, and to insert characters at the beginning of the line the user would type E ϕ ϕ string ϕ. It is also possible, using the A and B commands, to insert text **After** or **Before** a string of text in the current line.

A simple example follows:

M40
40.
People on greenhouses may not throw bones
E; on; in; E; b; st; A;nes;.
40.
People in greenhouses may not throw stones.

Note that the last delimiter of these commands is only necessary to separate the commands on the line and may be replaced by a carriage return symbol if no more commands follow it.

It is occasionally necessary to repeat a context command, particularly when trying to locate a line containing text that occurs in earlier lines. Thus if lines 33, 40 and 54 all contain the string **the**, and the line required is the last of these, the following sequence may result:

```
L,text
33.
This is not the text I want
N'
40.
Nor is this the text I require
N'
54.
This is the text I want.
```

The command ' used in this example means repeat the last typed context editing command, and this parameteric reference frequently saves the user considerable typing effort.

The context editing commands operate on the first matching string of text in the current line and so, to ensure the unambiguous localization of text, a facility is available to move the logical start of the current line to any desired position. If the symbol ↑ is typed, the start of the line is moved one character to the right and repeated use of this command allows the start to be repositioned anywhere. Associated with the ↑ command, there are the commands ← and #. The first of these resets the logical start of the line at the beginning and the second deletes the character at the logical start of a line. Hence a typical editing sequence may be as follows:

```
52.
This is text most of which I do not want.
↑↑↑↑↑###↑↑↑↑↑##############↑↑#######  E,.,!,←
52.
This text I want!
```

A series of commands may be presented to the editor on a single line and, when they have all been obeyed, the editor replies by typing the current line.

Verification of the line in this manner can be switched off by selecting **Unverify** mode and this considerably reduces the real time spent in the editor, although it does mean that, for example, the result of a locate may not be the string required.

A **Global** version of the exchange command is provided, denoted by G ϕ string A ϕ string B ϕ. This operates on the whole document starting at the current line and exchanges all occurrences of the first string with the second string. The user may also ask for a block of text to be typed using the T command. When the user has finished editing, he may wish to re-edit the new document and the * command has the effect of setting up the edited document as the new **source** document.

There are a number of other facilities that allow the user to control the processing of whole documents. The most frequently used is the **Print** command that specifies where copies of the edited document are to be sent. Variants of this command allow the user to produce a new listing of a document or arrange for a copy to be directed to any output device. A new source stream or command stream may be selected using the S and C

commands. After the commands have been read from a command stream selected in this manner, the editor returns to the original command stream. Thus C4 will direct EDIT to obey the commands on input stream 4 and return. Stream 4 may then be used again if the commands on this stream are to be obeyed more than once. This mechanism provides a useful subroutine facility.

Blocks of text may be moved from one place in a document to another place later in the document. For example, the following commands move lines 20 to 40 inclusive and insert these at line 100:

```
.....
M20 O2  |  move to line 20 and select output 2
.....
M41 O1  |  move past block of text and re-select normal output stream
.....
H100O2  |  insert output stream 2 before line 100
```

Other editing commands may be typed in between those written down so that the intervening text may be edited while it is being moved.

It has been found that it is convenient to be able to refer symbolically to the current line and to the end of the document. In EDIT the characters . and * are used for this purpose. Thus the command

```
R. *
⟨New material⟩
Z
```

will replace from the current line to the end of the document with new material and respond by typing, for example,

```
*450.
```

for a document containing 449 lines.

Occasionally a user may type a command that causes the editor to flood the teletype with output. Commands such as Type and Global are good examples of this. An interruption mechanism is provided to stop the editor in cases of this nature, and the program then waits for the next command to be typed. For example, having interrupted a Global command, one could continue with verification suppressed by typing U', i.e. set Unverify mode and repeat the last typed context command.

One point that should be noted that arises from the division of lines into parts is that a context, while it is contained in a line, will not be found by EDIT if it is divided between two parts of the same line. The only convenient remedy for this difficulty is to deal with the current line using a circular buffer and the need to keep the core store occupied by EDIT to a minimum has prevented this being implemented.

A system of this nature must include comprehensive diagnostic facilities, but there is an important question of whether one should continue editing when the system is being used off-line; there are some cases when this practice could lead to chaos, but on the other hand there are times when continuing could be extremely useful. The user of EDIT has the option of whether he wants to do this. There is a command that, when used, will cause EDIT to stop if there has been an error whilst editing; otherwise EDIT will continue after most errors. This ability to continue editing after errors is invaluable in certain cases when processing long documents off-line, so that many errors can be detected in one pass.

When this is done, however, the user must take care not to destroy the only copy of a document. A complete list of the editing commands available in EDIT is included in the Appendix.

THE IMPLEMENTATION OF EDIT

The Titan system[5] allows for pure procedures; only one copy of each pure procedure is held in the core store at any time and this is independent of the number of people using the program. The data space for each user is administered by the supervisor. Considerable benefits accrue when an on-line editor is a pure procedure since it will typically have many users and will be present in core store for long periods of time. EDIT is a pure procedure that occupies 700 words (48 bits) of core store and has data space consisting of 64 words of core store and 90 index registers (24 bits) associated with each user. Its speed is about 500 lines per second for line by line editing and about 150 lines per second for a context search (approximately 20 characters per line). The commands have been arranged so that their syntax is easy to decode although the efficiency of this stage is not critical. Once a command has been read in and decoded, the arguments are placed in standard registers and one of a number of routines is entered to carry out the operation requested. These routines may be roughly classified as follows:

(1) Perform an operation on specified lines of the source. For example: R, D, M, H, N, +.
(2) Perform a context operation. For example: A, B, E, G, L, F, #.
(3) Specify data for future operations. For example: K, U, ?, S, C.

The routines of the last group only involve storing small quantities of information and are simple to write. The remaining routines administer basic subroutines that include:

(a) move forward to line n and make this the current line,
(b) delete from the current line to line n − 1 and make line n current,
(c) tidy up when the end of the source is reached and turn the current output stream into the new source,
(d) output a line from the current line buffer,
(e) read a new line of input from the current source stream.

Having carried out the operation requested most routines return to read the next command; if the current line has changed, this will be typed at the user so that he can verify the change.

Since EDIT deals with one (part) line at a time the working space requirements are small. One region is required for the current line and in EDIT this takes 48 of the 64 words available.* EDIT is required to keep the current line when an insertion is being made before it (I and H commands) and so a further region of about 10 words is provided as a buffer. The character strings associated with the context commands are held in the index registers.

DESIRABLE EXTENSIONS

The following commands would have been valuable had space been available to include them, although their inclusion may have made the editor more confusing to use. A command

* The current line buffer therefore allows an extension of the maximum length that is read in (viz. 72 characters) to 96 characters. (Each character is stored in a 24 bit halfword.)

to interchange two strings and an explicit command to delete a string. For strings that are frequently typed, parameteric references are desirable and some more flexible means of locating strings would be useful. For example, the command

S ϕ string A ϕ string B ϕ

could search for a sequence of characters starting with string A and ending with string B, while allowing any number of additional characters to occur between the two strings. If this command were provided, a means of referring to the intermediate string would almost certainly be necessary.

Frequently, when locating a context, the user is uncertain which of two (or more) strings occurs first and, in this case, a command of the form '**locate** string A **or** string B' is useful. The above commands are examples of the use of **and** and **or** connectives between strings and their use implies that two symbols must be reserved to represent 'and' and 'or'. This is outside the scope of EDIT where there are no reserved characters in strings, the delimiters being user defined. Another form of extension that is feasible is to have conditions attached to commands. For example, a **Global** command that only operated between two points would be useful. Thus one could say

G ϕ string A ϕ string B ϕ between a and b

where a and b could be line numbers and this would mean globally exchange string A for string B between lines a and b.

ACKNOWLEDGEMENTS

I should like to thank the members of the Cambridge Supervisor group[6] and users of the Titan system who have suggested improvements to EDIT.

APPENDIX

A complete list of the commands in EDIT:

N	Take the *N*ext (part) line as the current line.
M a	*M*ove to line a of the document.
+n	Copy the next n lines.
−n	Delete the next n lines.
T n	*T*ype the next n lines.
Lϕsϕ	*L*ocate the line containing the string s.
Fϕsϕ	*F*ind the line starting with string s.
Aϕsϕtϕ	*A*fter string s insert string t in the current line.
Bϕsϕtϕ	*B*efore string s insert string t.
Eϕsϕtϕ	*E*rase string s and replace it with string t.
Gϕsϕtϕ	*G*lobally replace string s with string t.
'	Repeat the last typed context command.
I a	*I*nsert new material before line a.
R a b	*R*eplace lines a to b with new material.
D a b	*D*elete lines a to b from the document.
S m	*S*elect document m for the next line to be edited.
C m	Obey the *c*ommands on stream m and return.
O m	Select *o*utput stream m.
O mD	*D*elete output stream m.

P m n	*P*rint input stream m on output stream n.
P m L n	*L*ist input stream m on output stream n.
H a I n	Insert *i*nput stream n before line a.
H a O n	Insert *o*utput stream n before line a.
U	Set unverify mode; reset by next U.
?	Replace error messages by a ?; reset by next ?.
*	Make the edited document into the new source.
W	Finish editing.
Q	Quit: for use in an emergency.
V	Type the current line; this effect is also achieved by a CARRIAGE RETURN but V is used in unverify mode.
X n	Line image input stream n.
!	Move the character pointer along one.
←	Return the character pointer to the start of the line.
#	Erase the first character after the character pointer.
→	Stop editing if there have been errors, otherwise continue.
T letter	Set the terminating letter for the Replace and Insert commands.
K	Treat CARRIAGE RETURN as a normal character and not as an optional delimiter in the context commands; reset by the next K command. (This allows Carriage Return to be included as part of a character string.)
= n	Reset the current line number to n.

REFERENCES

1. D. J. Farber, R. E. Griswold and I. P. Polonsky, 'The SNOBOL 3 Programming Language', *Bell Tech. J.* **65**, 895–944 (1966).
2. J. H. Matthewman and J. C. Matthewman, *The Matthewman Paginating Program*, University Mathematical Laboratory, Cambridge, 1967.
3. D. F. Hartley, B. Landy and R. M. Needham, 'The structure of a multiprogramming supervisor', *Comput. J.* **11**, 247–255 (1968).
4. B. Landy and C. Whitby Strevens, 'TSAS—The time shared supervisor assembly system', *Comput. J.* **12**, 128–131 (1969).
5. S. R. Bourne, *Titan Library Routine Specification—Complete Program EDIT*, University Mathematical Laboratory, Cambridge, 1967.
6. D. F. Hartley (Ed.), *The Cambridge Multiple Access System*, Users' reference manual, University Mathematical Laboratory, Cambridge, 1968.