The CHEF Editor

by

M. A. Maclean
and
J. E. L. Peck

Technical Manual 80-1

1980 October 15

Department of Computer Science
The University of British Columbia
Vancouver, British Columbia, V6T1W5

A user's Manual
(Second Printing)

The CHEF editor

M.A.Maclean
University of Canterbury

J.E.L.Peck
University of British Columbia

CHEF is a portable text editor written in BCPL. It is a descendant of the UNIX system editor ED, but has a significantly different command structure and a number of new features.

The primary field of application for CHEF is computer source program editing. However there are some simple word-processing facilities in the form of text justification and centering which make it usable for document and letter preparation. In fact, this document was prepared using CHEF.

The editor is 'safe' in the sense that the entire file is copied into an internal workspace before any editing takes place. The original file is undisturbed until you explicitly request that it be updated.

How to use CHEF

The editor is started by entering the system command CHEF under RDOS or UNIX or $RUN CS:CHEF under MTS. When loaded the editor displays

```
Enter H for help (Q for quit)
>
```

and is then ready for the first command, entered on the same line as the prompt character '>' and completed by depressing the RETURN or ENTER key. Thereafter the editor displays '>' whenever it is ready for a new command.

On-line documentation is available. To obtain a general introduction enter 'H'. To obtain details of a particular operator or special character, enter 'H' followed by the character.

How to edit a file

To edit the file XXX use the command

```
>EF XXX
```

which reads the file into the editor's workspace and closes the disk file. Under MTS or UNIX this may also be done when the editor is started as in $RUN CS:CHEF PAR=xxx or CHEF -xxx. To

write the file out again, use the command

```
>WF .                    (the dot stands for the 'current file')
```

which will write the contents of the workspace to the file whose name was used in the last E command. To write it to a different file YYY use the command

```
>WF YYY                                                          .
```

Once a file has been read into the workspace, you can examine any line of it or print any range of lines. To examine line 56, for example, enter

```
>56
```

and to see lines 50 to 60, enter

```
>50,60
```

Often you might not know the exact line number. To print a line that contains the pattern 'abc', use the command

```
>/abc/
```

and to print the range of lines starting with one containing 'abc' and finishing with one containing 'def', enter

```
>/abc/,/def/                                                     .
```

If you want to insert a new line after line 3, enter

```
>3I
```

followed by the desired line, followed by a 'dot stop' line, i.e., a line consisting of a '.' only. When you insert a line in this way, of course, all succeeding line numbers are increased. To delete line 7 use the command '7D'.

To replace part of line 10, use the R operator:-

```
>10R/abc/def/;P
```

which replaces the first occurrence of 'abc' in line 10 with the string 'def'. The ';P' at the end causes the revised line to be printed for verification. This demonstrates two things: first, that CHEF commands can be concatenated using ';' as a separator and, second, that CHEF remembers the 'current line' and uses it as the location for the P (print) command.

Note that commands begin with a location followed by an operator. The full syntax of commands will be discussed below. The above was a quick summary of the simplest basic commands and a more complete description can be found in the command summary or in the CHEF on-line documentation.

Text storage in CHEF

A file being edited is stored in a workspace which can be of arbitrary size (the maximum varies with the implementation) and within this workspace the lines of the file are numbered from 1 to some 'last line'.

After each editing operation CHEF remembers the 'current line'. This can often be used to simplify the following command. The position of the the current line after each command is specified in the command summary.

Workspaces may be stacked using the N (new) operator: two or more workspaces may coexist, although only the most recent is available for editing. When the most recently created workspace is finished with, the previous one may be restored again with the Q (quit) operator.

As well as the workspace, there are 27 one-line buffers called 'controls' which are known by the letters A to Z and the symbol +. These can be used to store text or strings of CHEF commands or parts of commands and they may be edited in exactly the same manner as the lines of the workspace. Text can be transferred in either direction between the workspace and the controls and there is a macro substitution facility to enable the content of a control to be included anywhere in a CHEF command line. The controls are independent of the workspace and do not disappear when the current workspace is stacked and superseded by a new one.

If you give CHEF a single non concatenated command that alters the workspace, the content of the special control + is executed immediately afterwards. This provides an automatic verification facility. The control + initially contains a V (view) command, but you may change this to any other command string, including the null string.

The term 'location' is used to mean the line or lines where the action of a CHEF command is to occur. A location may be:-

i) a line of the current workspace,

ii) a 'range', i.e., a group of contiguous lines of the current workspace,

iii) a control.

Some CHEF commands admit all three possibilities as locations, some are more restrictive. Consult the command summary for details.

Workspace line and range specification

A line is specified by a 'finder', an arithmetic combination (using + and - only) of 'terms' such as the following:-

10	line number 10,
.	the current line,
$	the last line,
/abc/	(a 'search pattern') the first line containing the pattern 'abc' found by a forward search from the current line (the search 'wraps around' from the last line to the first if necessary),
\abc\	the first line containing the pattern 'abc' found by a backward search from the current line (With EBCDIC machines use '|' for '\'),
'A	(a 'tagger') the first line, found by a forward search, that has been tagged with the character 'A',
"A	the first line found by a backward search that has been tagged with the character 'A',
~/abc/, ~\abc\, ~'A and ~"A,	the negated forms of the four previous types, which require the absence of a match.

Some example of finders are:-

```
10   .+1   $-5   /abc/+3   'X-1                    .
```

A 'range' is specified with two finders separated by a comma or a colon. Thus, '1,5' specifies the range of lines 1 to 5 inclusive. The difference between the comma and the colon is only significant when a search has to be carried out to evaluate the finders. If the comma is used, both finders are evaluated from the point of view of the current line. If the colon is used, the current line is set to the value specified by the finder preceding the colon before the next is evaluated. The following are examples of ranges:-

```
.-5,.+5   \BEGIN\,/END/     /abc/+3:/def/
```

A range may be specified with three or more finders separated by commas or colons. In this case the string of finders is evaluated from left to right and the current line is set by each finder that is followed by a colon. The final value of the range is determined by the last two finders. For example:-

```
/END/:/BEGIN/:/END/
```

specifies the ALGOL compound statement following the one containing the current line.

In the command summary, the term 'region' is used to mean either a single workspace line or a range of lines.

Default values in finders and ranges

To reduce typing, various components of finders and ranges may be elided. The rules are:-

i) in a finder, if no term precedes a leading '+' or '-', then an elided '.' is assumed,

ii) in a finder, if no term follows a '+' or '-', then a '1' is assumed,

iii) in a range, one or more finders may be elided, leaving only ',' or ':'. The last finder, if elided, defaults to '$', the last but one to '1',

iv) in a range, the comma or colon may be elided if no ambiguity results,

v) if a complete location specification is elided the current line '.' is assumed.

Examples of the use of these rules are as follows:-

Entered	Meaning	Rules
+1	.+1	i
+	.+1	i,ii
++	.+1+1	i,ii
,10	1,10	iii
-,+	.-1,.+1	i,ii
.,	.,$	iii
/abc//def/	/abc/,/def/	iv
,	1,$	iii
	.	v

Controls as locations

A control may be specified as a location in many, but not all, situations where a line can be specified. To specify the control 'A' in the context of a location, the form '@A' is used, (e.g., in '@AD').

There is no concept of a range of controls, so the form '@A,@G' is not permitted.

Patterns

A pattern is a sequence of elements enclosed by delimiters. In a 'search pattern' the delimiters must be '/' (for forward search) or '\' (for backward search), and the pattern may be preceeded by '~' if you wish it to match by exclusion. This type of pattern is used in finders and as the operand of X (execute).

Patterns are also used by the R (replace) and S (segment) operators to specify a substring of a line. In these cases the delimiter may be any character except ';'.

CHEF attempts to match the pattern against all substrings of a line starting with the 'left margin' (initially column 1) and finishing at the 'right margin' (initially the last column). You may alter these margin settings with the L (load) operator to force the search to take place within restricted column limits.

The following are examples of elements that can be used in constructing patterns:-

A	matches the character A,
.	matches any character at all,
[A-Z]	matches any character in the range 'A'-'Z' (With EBCDIC use '<','>' for '[',']'),
[A-Z0-9+)]	matches any character in the ranges 'A'-'Z', '0'-'9', or the characters '+' or ')',
~[0-9]	matches any character that is not in the range '0'-'9',
~A	matches any character except 'A'.

Any of these elements may be followed by a '*' to signify that the element may be present zero or more times. Thus the pattern '[A-Z][a-z]*' matches any word having an initial capital followed by zero or more lower case letters.

The following special elements may be used:-

^	matches the start of the line, (with EBCDIC use '@' for '^')
$	matches the end of the line.

The forms '//' and '\\' are used to signify a pattern that is the same as the one last specified.

Replacements

In the command 'R/abc/def/', the 'def' is the replacement. The characters '^', '$' and so on have no special significance in a replacement, but the character '&' specifies the matched string. For example, the command ',RA/Pago/&-&/' will replace all occurrences of 'Pago' by 'Pago-Pago'.

Note that the command 'R/abc//' specifies a null replacement.

CHEF command syntax

A CHEF command consists of some or all of the following elements, in the order given:-

i) a location, specifying the scope of action of the command,

ii) an operator specifying what action is required,

iii) a modifier for that action,

iv) an appropriate operand.

The location preceding the operator indicates where a text modification is to be made, or the source of text to be printed or copied to a file. Some commands restrict the type of location that may be specified. For instance, the I (insert) operator requires that its location be a workspace line - one cannot insert new text after a control. The E (edit) operator, and a number of others, permit no location at all.

The operator is a single upper or lower case letter, except for the query operator which is '?'. The modifier, permissible for some operators, is a single letter or other character that modifies the action of the operator.

A variety of possible operands may follow the operator and its modifier. These include locations, file names, integers, text patterns and so on.

Extra spaces may be used to separate the above four elements of a command, but may not be used within each element.

Several commands may be concatenated on one line, if desired, separated by the character ';'.

Default commands

A command consisting simply of a location, with nothing following it, is treated as a P (print) command. This is useful as a quick means of specifying a line or range to be printed. If the location is elided as well, i.e., only a null line is

entered, the command is treated as

```
>.+1P
```

and the line that follows the current line is printed. This can be used to step quickly through a region of the workspace, printing one line at a time.

If CHEF commands are concatenated, these default interpretations do not apply when the location specifies a region of the workspace. Thus the effect of '..;10;..' is merely to set the current line to 10. However the form '..;@A;..' causes control A to be printed.

File names

File names may be used as operands for many commands. They are always preceded by the F modifier.

Each workspace has a current file name. When an E (edit) or N (new) command is used to copy a file into the workspace, the current file name is set. To refer to the current file name use the symbol '.'. Thus the command 'WF.' copies the workspace to the current file.

The current file name may also be set with the L (load) operator (including the null value) and it may be interrogated with '?F'.

Console protocol

CHEF displays the character '>' to request command input. A number of commands, separated by ';', may now be entered and execution starts when the line is terminated by RETURN (or ENTER).

The command C (change) and I (insert) without modifier, cause CHEF to enter input mode. When input is to a control line, a single line of text will be accepted (terminated by RETURN or ENTER). When input is to the workspace, you may type as many lines as you wish but the input must finish with a line whose only non blank character is a '.' in the first position (a 'dot stop' line).

CHEF signals errors by printing '?'. With experience you will find this sufficient in most cases, but you can obtain a diagnostic message by replying with '?'.

The 'escape' character

The escape character is '#'. The most common use for it is when you want the special characters used in pattern or

replacement strings to have their ordinary meanings. This applies to characters such as '^', '$', '/' and so on. To escape the special meaning, precede the character with '#'. Thus '#*' is equivalant to an ordinary '*' in a pattern. The character '#' can itself be escaped. Thus to express the pattern element 'any number of #', use the combination '##*'.

In most cases, special characters used in situations where their special meaning does not apply do not have to be escaped. Thus a '$' at the beginning of a pattern is not treated as the end-of-line symbol (unless it is the only pattern character).

A second use for '#' is to express a character in a pattern or replacement string by means of its octal code. Thus '#012' is equivalent to the ASCII character 'linefeed'. There must always be exactly three octal digits. Implementations in a hexadecimal environment (e.g., MTS) are likely to use two hexadecimal digits instead.

Macro substitution

A control line may be copied into the command line using the special character '%'. For example, if control A contains the string 'abc', this string will be substituted for each occurrence of the combination '%A' in the command line.

Text from the console may be included in like manner, using the combination '%%'. Each occurrence of '%%' in the command line calls for one line of text from the console, terminated by RETURN (or ENTER). No prompt character is printed.

As an example of the use of the macro substitution facility, the command 'R/%A/%%/' fetches the pattern to be matched from control A and the string to be substituted from the console.

The substitution mechanism is quite general: macro expansion takes place incrementally as the commands are interpreted and you may nest macro calls to an arbitrary depth.

'Programs' of commands may be constructed with macro substitution: the string of commands stored in control A, for example, may invoke further commands in control B with the macro call '%B'. You may even create an indefinite loop by invoking the commands recursively with '%A'. The K operator is available for terminating such loops as demonstrated by the following example. This program prints the contents of selected controls and is started by the command '%C':-

control	content
A	Enter list of controls (form GHI..)
B	@DK; @DS/[A-Z]//-ED; @FC@%E; @FR/^/%E - /; @F; %B
C	@A; @DC; %B
D,E,F	(working space)

The justification operator (J)

This operator justifies the lines of the given range within columns which are the 'left verge' (initially 1) and the 'right verge' (intially 65). It takes its source text from the left margin (initially column 1) to the right margin (initially the last column). Text justification assumes the following simple rules:-

i) an indented line remains indented,

ii) a blank line remains blank,

iii) a line indented beyond the threshold (initially column 6) is frozen,

iv) a line indented beyond the threshold and having the centering symbol (initially ':') in column one is centered (deleting the ':'),

v) a sentence ending in '.', '!', ':' or '?' and followed by an extra blank has that extra blank preserved,

vi) otherwise the text of the current line together with that of the preceding line is justified.

As an example, the commands

```
/^ *$/;/^   *~ /://-J
```

justify the following paragraph.

The segment operator (S)

This operator copies segments of the specified line to controls. If there is no modifier, these segments are the head, match, and tail substrings defined with respect to the first occurrence of the pattern that forms part of the operand. If the modifier is A then the segments are defined by all occurrences of the matched pattern. If you do not wish to save all the segments, you may replace any of the controls with an invalid control character such as '-' or '.'. For example, if the current line is 'axbcxdefxghi':-

>S/x/ABC causes controls A, B and C to contain 'a', 'x' and 'bcxdefxghi' respectively and

>SA/x/A.B.C causes controls A, B and C to contain 'a', 'bc' and 'def' respectively.

The undoing operator (U)

You can recover from certain editing mistakes with the U operator. The action of U, which takes no location or operands, is to retore the workspace to the state it had before the last command that altered it. Only certain 'willing' operators (C, D, I, J, R, T) can be undone and you must not have used any 'unwilling' operators (E, N, Q, QQ, U, X) in the intervening time. CHEF acknowledges your U command, in favourable cases, by printing the command string it uses to restore the workspace.

The execute command (X)

If you use an F modifier after the X operator, the editor opens the file, whose name follows F, and executes the commands in it. Nested XF commands are not permitted.

If there is no modifier and the operand is a search pattern or tagger, the effect of the command is to execute the remainder of the command string for all lines of the location that match the specified pattern or tagger (or which do not match, if the pattern or tagger is negated). Thus the following will print the line numbers of all lines containing the word BEGIN.

```
>,X/BEGIN/;PN
```

and the following will print all lines of the workspace having no leading blanks

```
>,X~/^ /;P
```

Command summary

The following abbreviations are used (examples in brackets).

```
 c    = control                    ( @A )
 ch   = character
 f    = finder                     ( 10 or /abc/+1 )
 f0   = finder or 0
il i2 = integers
 n    = file name
 p    = pattern                    ( /abc/ or \def\ )
 pcl  = pattern and control list   (/abc/PQR or /abc/P-R )
 pn   = pattern and new string     ( /abc/def/ )
 r    = region                     ( 10 or 10,15 )
 sp   = search pattern             ( /abc/ or ~\def\ )
 t    = tagger                     ( 'A or ~"B)
(.)   = the default location is the current line
(6)   = default value for a CHEF parameter.
```

Locn.	Op./Mod	Opnd.	Action
Change existing lines (C)			
c r (.)	C F	n	lines from a file
	D	c r (.)	lines from workspace (deleted)
	-	c r	lines from workspace
	-	-	lines from console
(. is set to the last line entered)			
Delete lines (D)			
c r (.)	D -		delete lines (empty control)
(. is set at next line, or end of workspace)			
Edit (E)			
-	E F	n	edit file n, set current name
	-	-	emtpy workspace, cur. name null
(. is set at end of workspace)			
Help (H)			
-	H -	ch	help message for that character
		-	general help message
Insert lines after a specified line (I)			
f0 (.)	I F	n	lines from a file
	D	c r (.)	lines from workspace (deleted)
	-	c r	lines from workspace
	-	-	lines from console
(. is set to last line entered)			

Locn.	Op./Mod	Opnd.	Action
Justify text to within specified verges (J)			
r (.)	J -	i1,i2	left verge, right verge
		i2	right verge
		-	current verges (1,65)
(. is set to the last line justified)			
Kill current command line if location specifies empty line (K)			
c f (.)	K -	c f	take new commands from operand
		-	no new commands
Load a parameter (L)			
-	L F	n	current file name (may be null)
	L	i1	left verge (1)
	R	i1	right verge (65)
	T	i1	indentation threshold (6)
	V	-	verification toggle
	:	ch	centering symbol (:)
	^	i1	left margin(1)
	$	i1	right margin (252 or 128)
New work space (N)			
-	N F	n	load file, set cur. name
	-	-	new empty workspace
(. is set at end of workspace)			
Print to the console, or a file (P)			
c r (.)	P A	-	text with line numbers and tags
	C	-	byte count only
	F	n	copy to the file n
	L	-	print in lucid mode
	N	-	line numbers of region limits
	-	-	print to the console
(. is set to the last line printed)			
Quit (Q)			
-	Q -	-	quit this workspace
	Q	-	quit all workspaces
	S	command	execute the system command
	S	-	temporary exit to the system
Replace a string in one or more lines (R)			
c r (.)	R -	pn	replace first match in all lines
	A	pn	replace all matches in all lines
(. is set to last line in which replacement occurs)			
Save segments in control lines (S)			
c f (.)	S -	pcl	head, matched, and tail segments to the control lines in order
	A	pcl	save all segments w.r.t pattern

Locn.	Op./Mod	Opnd.	Action
Tag line with a character (T)			
r (.)	T -	ch	set tag to specified character
	-	-	reset the tag
(. is set to the last line tagged)			
Undo a willing operator (U)			
-	U -	-	willing - C D I J R T
			neutral - H K L P QS S V W Z
			unwilling - E N Q QQ U X
View a region of the workspace (V)			
f (.)	V -	il	window is .-il,.+il (18 or 8)
	-	-	window as previously set
Write the workspace to a file (W)			
-	W F	n	copies workspace to file n
(. is not changed)			
eXecute CHEF commands (X)			
r (.)	X -	sp t -	executes remainder of command line repeatedly for matching lines (/ = forwards, \ = backwards)
(. set by last command executed)			
-	X F	n	executes commands in file n
Query a parameter (?)			
-	? F	-	current file name
	L	-	current left verge
	R	-	current right verge
	T	-	current indentation threshold
	:	-	current centering symbol
	V	-	state of verification toggle
	^	-	current left margin
	$	-	current right margin
	//	-	current pattern
	-	-	current error message

JELP/cs:chef