To: Distribution

From: Richard J.C. Kissel

Date: May 19, 1977

Subject: A new standard editor

This MTB is a proposal for a new standard editor for Multics. MTB-334 raised a number of issues with respect to the current editors on Multics. The motivation for this proposal is to provide answers and solutions to those issues.

This MTB proposes a number of new features which are not in qedx, although most of them are in ted. A partial list of features follows:

a) The command interface has been made more standard.

b) The syntax requires white space after every request and that buffer names must be delimited with ":". This makes the syntax extensible and less prone to typing errors.

c) The use of the options request allows the user to tailor the default behavior of the editor for ease of use.

d) The editor will be recursive and a subroutine interface to the entire editor, or to selected parts of it, will be available.

The command interface to the proposed editor is as follows:

```
edit {path} {-control_args}
```

where:

1. path

   is the pathname of a segment which is to be read into buffer "0". The default pathname for buffer "0" is set to path. If path is not specified the default pathname for buffer "0" is set to null. If path does not exist an error is given. The editor is always entered in edit mode ready to accept requests.

2. control_args are chosen from the following:

   -options (-op) "op_string"
      a string of options and values enclosed in quotes, see the options request description for the available options.

   -macro_path (-mp) path
      where path is the pathname of an edit macro, with the suffix ".edit" assumed.

   -arguments (-ag) A1...An
      are macro arguments to be put in buffer "$args" one per line. If given, this must be the last argument, since everyting that follows will be taken as macro arguments. They will also be available individually in buffers "$argi" (1<=i<=n). The number of arguments specified will be available (as a character string) in buffer "$nargs".

   -no_push (-np)
      This control argument will cause this invocation of the editor to be started with the current editor environment. Normally, (-np not specified) the current editor environment is pushed on a stack and a new environment is created for each invocation. If a current environment does not exist, an error will be given. The "path" argument may not be specified if the "-np" control argument is given. (The current buffer, all the other buffers, the buffer stack, the options in effect, etc., are all part of the editor environment. At present any previous editor environments will not be accessible from the current environment. This could be changed by the addition of level numbers to buffer names.)

The following is a description of the proposed editor requests and syntax:

Note: In the syntax specifications the symbol "▯" will be used to mean white space (tab or space) and the symbol "⌶" will be used to mean a newline character. At the end of a request "▯" may also mean a newline character. Also, the editor will allow requests to be either upper or lower case letters.

The following capital letters will be used in the syntax specifications. They are fully explained when first used.

N	no address allowed.
Z	a simple address.
P	an address pair (single line default).
B	an address pair (entire buffer default).
X	speed_type expansion.
S	special character processing.
Y	abbrev expansion.
O	substitute request options.
D	digit in a substitute request.
T	terminator.
E	escape sequence processing.
I	input type.

Addressing:

A simple address "Z" has the following format:

line_addr¦char_addr

where line_addr and char_addr may be composed of the following: (The capital letters, "S" and "X", are defined below.)

Symbol	line addr meaning	char addr meaning
$	last line in a buffer	last char on a line
.	current line in a buffer	current char on a line
±n	n lines before (+) or after (-) the current line	n chars before (+) or after (-) the current char
>SX/reg_exp/	forward search for a line containing a match for reg_exp with wraparound	forward search for a match for reg_exp in the current line with wraparound
<SX/reg_exp/	backward search with wraparound	backward search in the current line with wraparound
+SX/reg_exp/	forward search without wraparound	forward search without wraparound
-SX/reg_exp/	backward search without wraparound	backward search without wraparound
n	line number n in the current buffer	char number n on the current line

'n	absolute line number n in the current buffer	not allowed

Note: The absolute line numbers are set when a segment is first read into a buffer and do not change. A request would be provided to reassign all the absolute line numbers in a buffer. These are currently unimplemented features and this specification serves merely to reserve the "'" character.

Also, "/" stands for any delimiter that is legal in a substitute request (see below), that is, any character which is not a letter, digit, white space, or newline.

The following defaults apply for simple addresses:

1) if "line_addr" is not specified then ".¦char_addr" is assumed.

2) if "char_addr" is not specified for the first address of an address pair (see "P" and "B" below) then "line_addr¦1" is assumed.

3) if "char_addr" is not specified for the second address of an address pair then "line_addr¦$" is assumed.

4) if "reg_exp" is not specified, the last "reg_exp" is used, or an error is given if there is not one.

5) "/reg_exp/" is taken to mean ">/reg_exp/", <u>but</u> the delimiters must be "/" rather than any other punctuation.

The capital letters may be replaced in either order as follows:

Speed Type Expansion

X:

null
: use the mode specified by the "speed_type" option of the option request.

x
: expand "reg_exp" (and "replacement_string" in the substitute request and input lines in the input requests) using the speed_type processor for the request before use by the editor.

v
: do not use the speed_type processor expansion for

this request.

Special

S:

null
: use the mode specified by the "special" option of the option request.

s
: all special regular expression symbols are recognized unless they are preceded by "\c".

n
: the special symbols ".","*","^", and "$" do not have their special meanings in "reg_exp" and "&" does not have its special meaning in "replacement_string" for a substitute request (unless preceded by "\c"). That is, interpret the "reg_exp" and "replacement_string" as simple strings.

An address pair, "P", has one of the following formats:

```
:buff_name:Z,Z
:buff_name:Z;Z
```

where the two simple addresses specify a set of characters in the specified buffer. The "," means that both addresses are relative to the current address (value of ".¦."), and the ";" means that the second address is relative to the first address. (This is the same meaning that these separators have in qedx).

The following defaults apply:

1) "buff_name" not specified, "::Z,Z" or "::Z;Z", may be written as "Z,Z" or "Z;Z" respectively and addresses characters in the current buffer.

2) second address not specified, ":buff_name:Z," or ":buff_name:Z;", may be written as ":buff_name:Z" or ":buff_name:Z;", and means ":buff_name:Z,Z" or ":buff_name:Z;Z" respectively. (Note that the ";" must be specified. Also, default 1 may be used to write "Z,", "Z", or "Z;" if the current buffer is meant).

3) no addresses specified, ":buff_name:," or ":buff_name:;", may be written as ":buff_name:" (or ":buff_name" if there is no ambiguity) and means ":buff_name:.¦1,.¦$". (Note that by using default 1 one need not type any address if the current buffer is

meant).

An address pair, "B", has exactly the same format as "P", but default 3 is changed from ".¦1,.¦$" to "1¦1,$¦$".

An address is a complete entity and no white space is allowed within it.

Basic Editor Requests:

Read: (note: see below for the definition of "Y")

PrY▯path_name▯

Read the segment specified by "path_name" and append it after the addressed range in the specified buffer. The default pathname for the buffer is set to "not trusted" unless the buffer is empty, in which case it is set to "path_name". The value of ".¦." is set to the first character of the last line read. Notice that "path_name" may not contain white space.

Write: (note: see below for the definition of "Y".)

Bw▯

Write from the specified buffer to the default pathname if it is trustworthy.

BwY▯path_name▯

Write from the specified buffer to "path_name". Set the default pathname of the specified buffer to "path_name" if the entire buffer is written, otherwise set it to "not trusted". Notice that "path_name" may not contain white space.

Quit:

NqƗ

Either quit or print a warning about any buffers which were modified since they were last written and stay in edit mode.

Nqf⌶

Quit force, no warning.

Quit Write: (note: see below for the definition of "Y".)

Nqw⌶

Same as a "q" request but write the current buffer to the default pathname first, if trustworthy.

NqwY▯path_name⌶

Like the "q" request but write the current buffer to path_name. The default pathname of the current buffer is not affected.

Quit No-Pop:

Nqnp⌶

Quit, but do not pop the current editor environment from the environment stack. Thus, the current environment will be available for further use. The "Quit" and "Quit Write" requests will pop the current environment from the environment stack unless the "-no_push" control argument was given for this editor invocation.

The following replacement may be made for the capital letter:

Abbrev Expansion

Y:

null
: use the mode specified in the "abbrev" option of the option request.

x
: call the abbrev processor to expand "path_name" before use by the editor.

v
: do not call the abbrev processor for this request.

Delete:

Pd▯

Delete the addressed range.

Print:

P=▯

Print the addresses of the addressed range.

Ppn▯

Print the addressed range without addresses.

Ppl▯

Print the addressed range with addresses.

Pp▯

Print the addressed range using the mode specified in the "print_mode" option of the option request.

Note that "P¦" means the same as "Pp¦".

Substitute: (note: see below for the definition of "S", "X", "O", and "D".)

PsSXOD/reg_exp/replacement_string/▯

The following replacements may be made, in any order, for the capital letters ("S" and "X" are as specified before):

Options

O: (Note: any combination of these may be specified)

null
: do a normal substitute (i.e. qedx-like).

b
: substitute brief, do not print an error message if no match is found for "reg_exp".

q
: substitute query, ask user a question before each substitution is made.

p
: print all lines changed by the substitution.

Digit

D:

null
: do nothing special (unless otherwise specified).

n
: where "n" is a decimal number, only replace the n<u>th</u> match of "reg_exp" with the replacement_string.

Note that the "/" stands for any delimiting character not in the set {0...9,A...Z,a...z,space}, that is, not a digit, letter, white space, or newline. Also, failure to find a match for "reg_exp" does not cause an exit from a macro as now happens in qedx; however, the "$error_message" buffer is filled in and the error name may be used to take appropriate action.

Input: (note: see below for the definition of "T", "E", and "X".)

PaTEX[optional_line]

: enter input mode and append lines from "user_input" (normally the terminal) after the addressed range.

PiTEX[optional_line]

: enter input mode and insert lines from "user_input" before the addressed range.

PcTEX[optional_line]

: enter input mode and replace the addressed range with lines from "user_input".

The capital letters may be replaced in any order by the following ("X" is as specified before):

Terminator

T:

null
: the input is terminated by "‡.‡".

l
: line, input a single line, either the optional line, or the next line if there is no optional line.

punctuation
: any character, "x", that is not a letter, number, white space, or newline may be used to bracket the input. That is, the next occurence of "x" will terminate the input.

Escape (note: escape sequences are defined later.)

E:

null
: use the mode specified by the "escape_mode" option of the option request.

e
: process all escape sequences normally.

n
: no escape sequences are recognized.

Line Movement: (note: see below for the definition of "I" and "E".)

BmIEP▯

move <u>from</u> address range "B" <u>to</u> address range "P", and then delete address range "B". If "B" and "P" specifiy overlapping address ranges an error is given.

BkIEP▯

copy <u>from</u> address range "B" <u>to</u> address range "P". No change is made to address range "B".

The following replacements may be made, in either order, for the capital letters ("E" is as specified before):

Input

I:

null
: same as "a" below.

a
: address range "B" is appended after address range "P".

i
: address range "B" is inserted before address range "P".

c
: address range "P" is replaced (changed) by address range "B".

Help:

Nhelp⌇
Nhelp▯feature⌇

These requests provide general help or help on the specified feature of the editor. They work by calling the system help command with certain control arguments set.

<u>Intermediate Editor Requests (Buffers)</u>:

Buffer Status:

Nx▯

print the status of all buffers.

Nxm▯

print the status of all buffers which have been modified since their last write.

Nx:▯

print the status of the current buffer.

Nx$▯

print the status of all special editor buffers. The special editor buffers are described below.

Nx:buff_name▯

print the status of the named buffer.

Change Current Buffer:

Nub:buff_name▯

use "buff_name" as the current buffer. If ":buff_name" is omitted the top buffer on the stack will be used.

Nup:buff_name▯

push the current buffer on a stack and use "buff_name" for the current buffer.

Nul▯

list the contents of the buffer stack.

Nue▯

make the buffer stack empty, the current buffer does not change.

Buffer Create:

Nbc:buff_name▯

create an empty buffer with name "buff_name". If ":buff_name" is not specified, an error is returned. If "buff_name" is already a buffer, an error is returned.

Buffer Delete:

```
Nbd:buff_name▯
```

delete the specified buffer. If ":buff_name" is not specified, an error is returned. If "buff_name" is the current buffer, an error is returned.

Buffer Options:

```
Nbo:buff_name▯option▯value{,option▯value}}
```

where what follows the request is an options list.

The following options are available (short forms of the option names will be provided):

Option	Value	Meaning
auto_save	n	Every n lines in input mode a \w will be done.
set_path	path_name	set the default pathname to "path_name". If "path_name" is null then the default pathname is deleted.
trace	off	default
	on	if this buffer is executed as a macro, print each request before it is executed.

Note that if ":buff_name" is omitted the current buffer is assumed.

Global Requests:

```
BgSX/reg_exp/▯requests}
```

execute the requests which follow the "/" on all lines in "P" which have a match for "reg_exp".

```
BgSX^/reg_exp/▯requests}
```

As above but on all lines in "P" which do not have a match for "reg_exp".

"S" and "X" are as defined before, and "/" represents any delimiter that is legal in a substitute request. Another global request may not be among those to be executed by the global request. This request serves to sequentially identify a set of lines or partial lines (that match or do not match "reg_exp") in "B" at which the group of requests will be executed. Thus, the value of ".¦." and of relative numeric addresses will change for the group of requests as each line in the identified set is made the current line.

Options Request:

```
No□option□value{,option□value}}
```

where what follows the request is an options list.

The following options are available (short forms of the option names will be provided):

Option	Value	
escape_mode	e, n	see "E" for meaning
print_mode	l, n	(for the "p" request)
verbose	off	default
	on	print things
special	s	special characters, default
	n	no special characters, make ".", "*", "^", "$", "&" normal unless escaped
speed_type	v	default, do not use speed_type expansion.
	x	use speed_type expansion.
abbrev	v	default, do not use abbrev expansion.
	x	use abbrev expansion
short_err	on, off	
long_err	on, off	

err_name	on, off
err_line	on, off

The "verbose" mode will print out "INPUT" or "EDIT" when the appropriate mode is entered and will print out any lines changed by a substitute request.

The error messages have a short form, a long form, an internal editor error name, and the line on which the error occurred, as parts. The various options control the printing of these parts. Also, the entire error message will be available in a special editor buffer, "$err_message", one part per line; and in the four buffers: "$err_short", "$err_long", "$err_name", and "$err_line", one part per buffer.

In general, a buffer name starting with "$" will be assumed to be a special read-only editor buffer and will contain useful information. These buffers cannot be made the current buffer and the status requests (except "x$") ignore these buffers.

A list of planned special buffers follows (short forms of the buffer names will be provided):

$args	all macro arguments.
$argi	1<=i<=$nargs, individual macro arguments.
$nargs	number of macro arguments.
$last_rstr	last "replacement_string" in an "s" request.
$last_rexp	last "reg_exp".
$last_line	the last line input.
$error_message	complete error message.
$err_short	short error message.
$err_long	long error message.
$err_name	internal error name.
$err_line	line causing error.

Advanced Editor Requests (Programming):

Special:

N"comment_line⌇

The "comment_line" is a comment and is ignored.

Pn▯

The value of ".¦." is set to the first address in the range "P".

Pf/reg_exp/▯

This request searches for a match for "reg_exp" in address range "P". If a match is found, the address is printed, otherwise "0¦0" is printed. The value of ".¦." is not affected by this request.

Transfers Within the Current Buffer:

}Nlab(label)▯

Identify "label" with this line. This line may be transferred to by a "to" request. Notice that this must be the first request on a line.

Nto(label)▯

Transfer execution control to the line identified by "label".

NteT(err_name)▯optional_line}

This request defines a group of requests delimited by "T" which are to be executed if editor error name "err_name" occurs. If "err_name" is omitted, written "()", then any error not handled by an explicit request will cause execution of the group of requests. If "te" is preceded by a "lab" request and control is transfered through the label (rather than an error) the "te" and delimiter will be no-ops. This request is like a pl1 on unit. Executing this request activiates the error handling in the current buffer and all lower buffers reached through macro calls. Specifying an empty request group terminates the handling of the specified "err_name" at a given level and lower levels. Specifying a non-empty request group activates the new error handling at the current level and lower levels.

Conditional Request:

NifT(expression)[optional_line]

Execute the group of requests terminated by "T" if "expression" is the character string "true". Otherwise, continue execution after the group terminator. No "lab" requests are allowed in the request group.

The "T" terminates the request group as before. Currently, "expression" must evaluate to the string "true" or "false". If it does not, an error will be returned. The "expression" may be a combination of active functions and editor builtin functions which give this result. The order of evaluation is: editor builtin functions first, followed by active funtion expansion. Editor builtin functions always start with the "%" character and may only appear in an "expression". The currently implemented editor builtin functions are:

%in(buff_name,Z,Z,Z)

Where "true" is returned if the last address is within the range specified by the first two addresses in the specified buffer, and "false" otherwise. If "buff_name" is not specified the current buffer is assumed.

%empty(buff_name)

Where "true" is returned if the named buffer is empty and "false" otherwise. If "buff_name" is not specified the current buffer is assumed.

%addr(buff_name,Z)

Where the character representation of the address "Z" in the named buffer is returned, or "0¦0" is returned if "Z" is not an address in the named buffer.

In the future we hope to implement a much nore general expression evaluator which will allow variables and various arithmetic, string, and boolean operators as well as the functions above.

If done in a reasonable way this variable manager and expression evaluator will be usable by the rest of the system as well as the editor.

Also, the editor will be callable as an active function. This will allow the use of "exec_com" programming features by the editor.

Execute:

Ne▯command_line⌇

Send the command line to the Multics command processor.

Expand:

Nex▯abbrev_line⌇

Send the "abbrev_line" to the abbrev processor for expansion and then execute it as if the "ex" request had been removed. This allows editor abbrevs.

File Output:

Nfo:buff_name▯requests⌇

Execute the requests with buffer name as the output file, do an automatic "co" at the end. If "buff_name" does not exist it will be created, if ":buff_name" is omitted the current buffer is assumed.

<u>Escape Sequences</u>:

\B▯

Expands the specified part of the specified buffer in line including the last newline if any ("B" is an address pair as specified before). The entire escape sequence including the required white space is replaced. When parsing a request this escape sequence will be treated as a single entity, thus, it may be used to group requests on separate lines and have them treated as if they were on one line. This is especially useful in the global request and the file output request. If a syntax error occurs when executing requests from a buffer, the buffer is exited and further reqests are taken from the calling buffer.

\nB▯

Like \b but without the last newline.

\r▯

Replace the escape including the required white space with a line from the terminal including the last newline.

\rn▯

Like \r but without the last newline.

\?▯

Prints "INPUT, terminator is T" or "EDIT, you may type 'help'". "T" is as before, and the "you may type 'help'" part of the second message is only printed the first time this escape is used in an editor invocation.

\[active_func]▯

Expand the active function. Any escape sequences in "active_func" are expanded first. The entire escape sequence including the required white space is replaced.

\p:buff_name.suffix▯

Expands to the absolute pathname of the named buffer (in the process directory since it is a temp seg). If specified, the ".suffix" causes the name "buffer_entryname.suffix" to be added to the buffer segment (it will be removed when the editor exits). This allows commands that require a suffix to operate directly on the buffer segment. If ":buff_name" is omitted the current buffer is assumed. The entire escape sequence including the required white space is replaced.

\w▯

Write to the default pathname of the current buffer or give a message if there is not one. Normally this is only used in input mode since it is equivalent to "w" in edit mode.

\dp:buff_name▯

Expands to the default pathname of the specified buffer or the null string if there is not a default pathname. If ":buff_name" is not specified, the current buffer is assumed. The entire escape sequence including the required white space is replaced.

\c

Handle the next character with special processing.