Kooch - IPC

To: Distribution

From: Gabriel D. Chang

Subject: The Editor

Date: March 31, 1978

This MTB describes an attempt to improve on the current version of qedx, make it easier to learn and make it more powerful to use. It is hoped that this editor may be acceptable to the majority of Multics users.

With reference to MTB-334 Current Editor Problems written by Richard J. C. Kissel, the main drawback or inadequacy for the current version of qedx seems to be:

1. pathname does not appear in the command line.

2. novice users have difficulty learning to use the command.

3. "\f" seems an awkward way to terminate an input mode.

4. "dcl", "cwd", etc may cause a lot of grief.

5. some very useful ted features are missing.

The MTB-339 A New Standard Editor written by Richard J. C. Kissel published May 19, 1977 was an attempt to solve these problems.

However, the author feels the editor as designed in MTB-339 still has a number of undesirable features.

1. Without special consideration for novice users, the editor is even more difficult to learn than qedx:

   a) A generalized address range of the form :buff_name:L|C,L|C is more difficult to learn than L,L used in qedx.

Multics Project internal working documentation. Not to be reproduced or distributed outside the Multics Project.

b) The concept of modifiers on a simple request is not easy to understand.

c) The mandatory blank required after a request forces a user to remember the appropriate place to put in the blank.

2. The inclusion of speedtype expansion features into the editor is not a good idea, for the following reasons:

a) The syntax for speedtype expansion is drastically different from the syntax of the editor, the novice user will undoubtedly be confused by speedtype expansions.

Examples:

Let "the" be the expansion for "t",
and "truncate" be the expansion for "tc",

a mistype "dc"
and a substitution s/d/t/ will give "thec" in the expansion mode, or unexpanded "tc" in the text in the nonexpansion mode, neither of which is the desired result.

While a mistype "ent"
and a substituion s/t/d/ will result in a substition error in the expansion mode.

b) Speedtype expansion can be done easily after the entire file has been edited and written out.

c) Excluding the speedtype capabilities will make the editor cleaner and faster.

The fact that it is dangerous to expand the expanded text in speedtype requires invoking the procedure retain_symbols unnecessarily often.

3. Some modifications to requests are not important enough to warrant their inclusion.

a) Abbrev expansion.

Since the pathname of the file to be edited is now included in the command interface, the need to expand pathnames diminishes.

p) Escape sequence processing.

4. Compatibility.

The author feels compatibility to current qedx is important. Although it is not possible to design an editor totally compatible with qedx, an effort should be made to retain as much qedx syntax as possible. After all, qedx has been tested and used widely on the Multics system for a number of years, and has its share of virtues and followers. The author can see the real danger of designing a totally incompatible editor that discourages qedx users from converting over.

5. Minimization of typing.

It does not look like a very good tradeoff to require blanks after requests when most of the extensions are only made to the advanced editor.

With these considerations in mind, the proposed new editor is carefully partitioned into three levels, corresponding to the basic editor, the intermediate editor, and the advanced editor. Lower level users are not allowed to use higher level features, thus they are able to get away with learning only the features that will concern them, and be able to use their level features confidently.

The command interface is

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

where

1. path

is the pathname of a segment 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 a segment corresponding to path does not exist, the user will get a warning, and such a segment will be created by the first write request.

2. control_args are chosen from the following:

-level (-lv) n

where n=1 denotes the basic editor, n=2 denotes the intermediate editor, and n=3 denotes the advanced editor. When the level control argument is not given, the default is level 1. Basic users are not allowed to use intermediate features, and intermediate users are not allowed to use advanced features.

-no_special_char (-nosch)

specifies that the special characters (^ $. * % ?) do not have special meanings attached to them in regular string context or replacement string context.

-special_char (-sch)

specifies that the special characters retain their special meanings.

Default is -nosch for the basic editor and -sch for the intermediate and advanced editors. Also note that -sch and -nosch are mutually exclusive.

-brief (-bf)

specifies that the user does not wish to receive queries, reminders, warnings, and the user wants to keep the error messages short.

-macro_path (-mp) path

where path is the pathname of an edit macro, with the suffix ".edit" assumed if omitted. This control argument cannot be used with the basic editor.

-arguments (-ag) A1...An

are macro arguments to be put in the buffer "$args", one per line. If present, this must be the last argument, since everything that follows will be taken as macro arguments. They will also be available individually in the buffers "$argi" (1<=i<=n). The number of arguments specified will be available (as a character string) in the buffer "$nargs". Again, the control argument cannot be used with the basic editor.

The Basic Editor:

As mentioned above, the default level is one. This is the basic editor, and is aimed primarily for the novice users.

In the basic level, an effort has been made to give ample warning, guidance, and safety features so that a novice user may be instructed along the way, and any misunderstanding or slip of the finger may not cause disastrous effects.

The basic editor includes the following well known qedx requests:

a

c

d

e

g

i

n

p

q

r

s

w

=

.

The a, c, or i input request must be followed immediately by a newline character.

Use of the e request will get a query.

Deleting more than four lines at a time will get a query.

Use of the w request with a pathname will get a query.

The use of line range with a w request is not allowed.

Use of the q request with a modified buffer will get a query.

Use of the g request will get a query.

A maximum of one request per line will be allowed for a novice user.

The single character "." on a new line terminates an input request (a, c, i). No provision is given for the novice user to have an input request stopped in the middle of a line.

The user is reminded when switching between input and edit modes.

Strictly no buffer usage.

Some new requests are added, among them

k copy a number of lines after a given line number in the same buffer. The argument after k must be an absolute line number.

Example:

3,5k7

copies lines 3-5 and appends these lines after line 7 in the same buffer. Value of "." is set to line 10.

m move a number of lines after a given line number in the same buffer. The argument after m must be an absolute line number.

Example:

3,5m7

moves lines 3-5 after line 7 in the same buffer. Value of "." is set to the new line 7.

l print a number of lines with line numbers.

> forward search without wraparound.

< backward search without wraparound.

j join the remainder of the request line to the end of a number of lines. This provision helps to insert comments at the ends of lines.

Example:

5j /* the pointer here must be null */

joins the string " /* the pointer here must be null */" to the end of line 5. (Note: the runoff features with which this document is made may have fouled up the tab and the spacing, but the readers should not have trouble identifying the intent of this example.)

The Intermediate Editor:

All the requests in the basic editor and those in qedx are honored in the intermediate editor. The builtin safeguards and restrictions for the basic editor will be lifted in the intermediate editor, with the exception of the mandatory new line after an input request.

However, the use of the w request with a pathname will get a query if no blank separates the w and the pathname.

The backslash f is still honored to enable the user to stop the input request in the middle of a line.

More than one request can be made on a single line. Blanks will not be required to separate each request. However, the requests will be stacked, and will only be processed when no syntax error has been detected for the entire request line. This is consistent with the concept of "minimization of typing". Thus, familiar nuisances like "dcl" and "cwd" will be diagnosed, and can be avoided.

Other modifications to the intermediate editor include:

The k and m requests are extended to enable the user to copy or move a number of lines from the same or different buffer. The argument after k must be an absolute line number or a buffer name.

Example:

3,5k7

has the same meaning as the example given in the basic editor.

3,5ky

copies lines 3-5 and appends these lines at the end of buffer y. If buffer y does not exist, one will be created. Value of "." in the current buffer is 5, and the value of "." in buffer y is the value of "$" in the buffer + 3.

Single character buffer names "0" - "9" immediately following the k and m requests must be enclosed within parentheses. If single digit numbers appear after the k or m request, the user will get a query.

To extend the concept of the "last regular expression", "%" will be used to represent the last address range and "%" in a replacement string is used to represent the last fully expanded replacement string.

Example:

Assume the value of "." is 35.

The request

```
-5,.5s/string1/string2/
```

sets the value of "." to 40, and the value of "%" to 30-40.

Example:

Consider these four consecutive requests:

```
40,$s/arg1/arg_1/

%s/arg2/arg_2/

39s//%/

%s/args/arg_1,%/
```

The second request does the substitution s/arg2/arg_2/ for the same line range (40 to the end of file) as the first request.

The third request does the substitution s/arg2/arg_2/ for line 39.

The fourth request does the substitution s/args/arg_1,arg_2/ for line 39, although using "%" as the line range here is hardly necessary.

Note: The value of the last address range is defined after a substitute request, and remains defined when there is no addition or deletion of lines in the buffer and the value of "." must remain unchanged. Any reference to an undefined "%" will be diagnosed. The value of the last fully expanded replacement string is defined after the first substitute request, and changes when there is a different substitute request.

To extend the notion a step further, the concept of the "last substitution" is introduced.

The request z performs the last substitution for a different line range.

Example:

Consider these two requests:

```
-5,.5s/string1/string2/p
```

The user may find to his dismay he should have typed -5,.6s.... He can simply correct his mistake by typing

```
.1z
```

In combination with //, the z request very closely simulates the substitute query capability outlined in MTB-339.

The value of "last substitution" is defined after the first substitute request and changes when there is a different substitute request.

The use of z redefines // to "string1" used in the last substitution.

The delimiter following the replacement string is optional, if the replacement string is followed immediately by a non-escaped newline character.

With the exception of \C, \B, \R, and \F, upper case character requests will not be recognized as their lower case equivalents, thus freeing 26 upper case letters for future extensions.

Advanced Editor:

All the features and requests of the intermediate editor are accepted by the advanced editor.

In order to satisfy more sophisticated users, the editor provides character addressing capabilities and some relatively useful programming features. In order not to confuse the lower level users, discussion of these features will only be included in the advanced section of the documentation. This MTB just outlines the two features built into this advanced editor in this MTB to give the readers a flavor of the capability and the ease of extensibility that can be made to the editor, and leave the detailed design of the advanced editor to a future date. But it should be pointed out that almost all of the advanced editor requests (programming) proposed in MTB-339 will be implemented with perhaps a slight change in syntax.

1. Character addressing will be implemented using the scheme proposed in MTB-339, using the vertical bar to separate the line and character portions of an address.

2. More advanced request names with more than one character must be delimited by colons, and the arguments they take may be enclosed within parentheses.

   Thus, :label: (a) may mean a label with name a; and :if_exists_line:329 may be the first part of a conditional statement, etc.

Conclusion:

It can be seen, the proposed editor extends the current qedx in two directions: making it easier to learn at one end, and making it more powerful at the other end. Undoubtedly an ambitious user can progress from one level to the next without too much difficulty, while contented users can remain at their own levels indefinitely if they so desire.

It is hoped that this proposed editor be adopted as the Standard Multics Editor.

Please address any comments on this MTB to:

Gabriel Chang
Honeywell Information Systems Inc
575 Technology Square
Cambridge, Ma. 02139

or send mail to Chang.Multics

or call 617-492-9314
         HVN-261-9314