To: Distribution

From: Steve Herbst, Tom Casey

Subject: Changes to abbrev

Date: 05/01/78

This MTB proposes adding some new features to abbrev, among them the editing of command lines and abbrev request lines.

One of our most important design goals is ease of use. The expansion of user-defined abbreviations in command lines greatly reduces the amount of repetitive typing a user has to do. Even with abbreviations, it is sometimes necessary to type long and complex command lines. A typing error is usually not apparent until the entire line has been typed and sent to the command processor. When the error is discovered, the entire line must be retyped.

The uninstalled audit_ I/O module is able to provide editing of all input lines, not only command lines and abbrev request lines. Unfortunately, audit_ adds some overhead to every line by making extra calls and touching extra segments.

It is relatively easy for abbrev to save the last line it intercepted, for editing and re-execution. Jay Goldman's abbrev, which has this feature, has been used for years by a substantial portion of the MIT community.

The proposed editing syntax is patterned after qedx. Since the text to be edited is a single line, searches and buffer manipulation are unnecessary. Two editing commands are sufficient:

p	Print the line.
s	Substitute for all occurrences of a substring.

Three more operations are added by abbrev:

x	Expand abbreviations in the line.
e	Execute the line.
E <cl>	Expand <cl> and pass it to the command processor.

These editor commands can be used either on the saved line or on a fresh line before it is executed. To invoke the editor on the saved line, the user types the .$ abbrev request. To invoke the editor on a fresh line, the user types the line followed by a $. In both cases, the user can replace $ by a personally defined edit control string. Except for its response to this edit control string, abbrev's editor is invisible to the casual user.

Automatic editing mode can be turned on by saying "abbrev -edit_auto". In auto edit mode, the editor is entered whenever the command processor returns an error code due to a syntax error in the line or "Command not found". This feature is optional and compatible.

A version of abbrev implementing most of the new features resides in >udd>PDO>Goldman>o. It should be initiated with the names abbrev, ab, and abbrev_. This version lacks two of the features described in this MTB. It does not edit abbrev request lines, and it does not follow the qedx rules for regular expressions in subsitutions and list requests.

Send comments and suggestions to Herbst.Multics.

SUMMARY OF CHANGES

1. Implement the editing features.

2. Make the abbrev command accept control arguments to set and print modes of operation for the abbrev facility.

3. Store a string of break characters in the user's profile, making break characters static rather than per-process. Add a set of abbrev requests to manage break characters. Make the abbrev$set_break and abbrev$reset_break commands obsolete.

4. Add new useful forms of the .l request.

5. Rename the subroutine entry point abbrev_$expanded_line to abbrev_$expand_cl. This entry point expands command lines. Add a new entry point, abbrev_$expand_non_cl, that does not expand begin-type (command name) abbreviations. The new entry point is for expanding input lines that are not command lines, for example the pathnames in editor read and write requests.

NEW CONTROL ARGUMENTS

-edit_on, -en
-edit_off, -ef
-edit_auto, -ea

set abbrev's editing mode. See the discussion of EDITING below.

-edit_string STR, -es STR

change the edit control string to be STR, up to four characters in length. This string, $ by default, causes abbrev to enter edit mode. The .STR abbrev request invokes the editor on the saved command line, and STR at the end of a newly typed command line invokes the editor on that line.

-request_char C, -rc C

change the request character to be the single character C. The request character, which is . by default, begins abbrev requests.

-long, -lg

print status information about the abbrev facility. Included are the pathname of the current profile, the break characters defined for this profile, the edit control string and request character, and whether editing and expansion are turned on.

-brief, -bf

print a single line of status information for abbrev, consisting of the request character, the edit control string, and whether editing and expansion are turned on.

-revert

turn off the abbrev facility. This control argument to abbrev is equivalent to the .q request inside abbrev.

-off

turn off expansion mode.

-on

turn on expansion mode.

SUBROUTINE INTERFACES

The entry point abbrev_ expands a command line (if expansion is turned on) and passes it to the command processor.

The entry point abbrev_$expand_cl expands a command line and returns the result. The entry point abbrev_$expand_non_cl does the same but does not expand begin-type (command name) abbreviations.

BREAK SETTING

The following new abbrev requests manage break characters:

.bl
: List the abbrev break characters defined for the current profile.

.ba STR
: Append STR to the string of break characters. White space characters blank and tab are always break characters.

.bd STR
: Delete the characters in STR from the string of break characters.

.br
: Reset the break string to its default, the characters currently listed in the MPM.

NEW LIST REQUESTS

The following requests list by abbrev name:

.lb {AB1 AB2 ... ABj}

List begin-type abbreviations, those that are expanded only at the beginning of command lines.

.l^b {AB1 AB2 ... ABj}

List abbreviations that are not begin-type.

The following list by string starting the abbrev name:

.la STR1 {STR2 ... STRj}

List abbreviations starting with any of the specified character strings.

.lab STR1 {STR2 ... STRj}

List begin-type abbreviations starting with any of the specified strings.

.la^b STR1 {STR2 ... STRj}

List abbreviations that are not begin-type and start with any of the specified strings.

These list by qedx regular expression matching the name:

.ls REG-EXP1 {REG-EXP2 ... REG-EXPj}

List abbreviations matched by any of the specified regular expressions. Regular expressions have the same syntax as in qedx.

.lsb REG-EXP1 {REG-EXP2 ... REG-EXPj}

List begin-type abbreviations matched by any of the specified regular expressions.

.ls^b REG-EXP1 {REG-EXP2 ... REG-EXPj}

List abbreviations that are not begin-type and are matched by any of the specified regular expressions.

These list by regular expression matching the expansion:

.lx REG-EXP1 {REG-EXP2 ... REG-EXPj}

List abbreviations whose expansions are matched by any of the specified regular expressions.

.lxb REG-EXP1 {REG-EXP2 ... REG-EXPj}

List begin-type abbreviations whose expansions are matched by any of the specified regular expressions.

.lx^b REG-EXP1 {REG-EXP2 ... REG-EXPj}

List abbreviations that are not begin-type and whose expansions are matched by any of the specified regular expressions.

EDITING

.e on

Turn on editing of command lines and abbrev request lines. With editing on, edit mode can be entered by means of the .<EDIT-CONTROL-STRING> (default .$) abbrev request or by <EDIT-CONTROL-STRING> at the end of a line.

.e auto

Turn on auto editing. Edit mode is also entered automatically when the command processor gets a syntax error or cannot find a command or active function.

.e off

Turn editing off. This includes auto editing.

.E <LINE>

execute the abbrev request line <LINE> or expand and execute the command line <LINE> without changing the saved line.

.<EDIT-CONTROL-STRING> EDIT-REQUESTS

(default .$) invoke the editor on the saved command line.

EDIT-REQUESTS are valid in edit mode and on the .$ request line. The valid edit requests are:

p	print the line being edited.
s/STR1/STR2/	substitute all occurrences of the string STR1 in the line being edited with STR2. This request is identical to the s request in qedx. All three delimiters are required, but they do not have to be /'s. Any character not appearing in STR1 or STR2 can be used as a delimiter.
x	expand abbreviations in the line being edited, as if it were a command line.
e	execute the line being edited.
E <LINE>	execute the command line or abbrev request line <LINE>. The line <LINE> is saved and can be edited after its return by typing E.<EDIT-CONTROL-STRING>.
q	quit edit mode and return to command level.

Multiple editing requests are allowed on a line, as in qedx. The e and q requests must each be the last request on an editor request line.

The command line editor types very brief error messages. The message "??" indicates a syntax error on the editor request line. The message "NO" indicates that an editor request could not be performed, as when STR1 is not found by the s request.

A command line is saved after its execution completes or is abnormally terminated by the release command. If the execution of a command line is interrupted (by a QUIT, for example), that line is not available for editing until release is typed.

Auto editing takes place only if the I/O module managing user_input supports a resetread operation. Auto editing does not take place, for example, in an absentee job or in an exec_com while &attach is in effect.

When edit mode is entered automatically, a message is printed and a resetread is performed.

The abbrev requests .r and .f are retained as obsolete. The .r request is equivalent to ".e on" and .f is equivalent to ".e off".

The rest of this MTB is a narrative description of the modified abbrev facility, as it should appear in the MPM document on the abbrev command.

MPM DESCRIPTION:

The abbrev command causes the abbrev_ subroutine to intercept lines input to the command processor.

Lines beginning with the abbrev request character, a period (.) by default, are processed within abbrev_ as abbrev request lines. Abbrev requests modify how command lines are to be processed, edit command lines and other abbrev request lines, and manage the user's abbreviation profile. The user profile is a segment in the default working directory:

```
>user_dir_dir>Project_id>Person_id>Person_id.profile
```

There are abbrev requests to define, delete, and selectively list abbreviations. The abbrev request character can be changed by saying "abbrev -request_char X". The request character is defined on a per-process basis.

Lines not beginning with the abbrev request character are command lines. These are expanded by replacing abbreviations with their values, and passed to the command processor.

The command processor called by abbrev_ is whatever command processor was in effect before the abbrev command was invoked. By default, this is the system's command_processor_. It can be changed before abbrev is invoked by calling the cu_$set_command_processor subroutine, or after abbrev has been invoked by calling the abbrev_$set_cp subroutine.

Abbreviations are character strings up to eight characters in length delimited by break characters. The set of break characters is stored in the profile and can be changed by abbrev requests. By default, the break characters are:

```
$ " ' ` . ; | ( ) < > [ ] { }

HT, VT, FF, SPACE, NEWLINE
```

The abbrev request to add a break character first checks to make sure that no abbreviations currently defined contain the new character.

There are two types of abbreviations. Begin-type abbrevs, marked by a "b" when they are listed, are only expanded at the beginning of a command line or following the command line delimiter (;). Other abbreviations are expanded anywhere in the line.

SAVING LINES

The editing feature is enabled by saying "abbrev -edit_on". This causes abbrev to always save a copy of the most recently executed line. A line is not saved until it has been executed. A command line is not saved until the command processor returns or the "cleanup" condition is raised in an invocation of abbrev. The latter occurs when the command line is interrupted, for example by a QUIT, and the release command is invoked.

A common practice is to interrupt a command line known to be wrong by issuing a QUIT, type "release" and edit the saved command line. It can happen, however, that the command line has already completed when the user issues the QUIT, even though the terminal is still printing its output. In this case, the command line has already been saved and typing "release" causes "release" to become the saved line. Therefore, one should always type the .s request after QUIT to see the saved line before releasing.

EDIT MODE

The edit control string, dollar sign ($) by default, is used to enter edit mode. By saying "abbrev -edit_string XXXX", the user can change the edit control string to be any string of one to four characters. The edit control string is defined on a per-process basis.

Edit mode is entered one of three ways. The .<edit_control_string> (eg., .$) abbrev request enters edit mode, editing the saved line. Edit control string at the end of a command line or abbrev request line causes that line to be edited in edit mode. The third way is via automatic editing, enabled by saying "abbrev -edit_auto". With auto editing on, any command line that causes the command processor to return an error code (syntax or "Segment not found.") is edited in edit mode. The message ".<edit_control_string> " is printed and a resetread operation is performed on user_input whenever this happens. Edit mode is only entered automatically if the I/O module managing user_input supports the resetread operation, which it does not do in absentee jobs or in exec_com's with &attach in effect.

Edit mode makes an internal copy of the line to be edited. Edit requests modify this internal copy, which is not to be confused with the saved line. Editing changes the internal copy. The saved line is only changed in editing mode by the e and E operations, which call the command processor. When the q edit request is used to exit edit mode, the internal copy is destroyed.

A program_interrupt handler is established when edit mode is entered. Invoking the program_interrupt command after a QUIT re-enters edit mode.