Identification

option, delopt, printopt
C. Marceau

Purpose

This section describes the commands which set and read options. Option sets options. Delopt deletes options. Printopt prints values of options.

The reader should be familiar with BX.12.00 on the use of options in Multics. BX.12.01, on the representation of options in storage, is a prerequisite for understanding the implementation of options commands. The commands use the procedures described in BY.9.01 - BY.9.05.

Option - usage

\[
\text{option (p|s opt1 on|off -spec- ... optn on|off -spec-)}
\]

- \(p|s\) indicates whether the options are permanent, or are set for the duration of the console session.
- optj is the name of the option to be set.
- on|off indicates whether the option is to be set ON or OFF.
- -spec- is a character string, the specification for the option. If the user types no specification (i.e. omits the argument entirely) and the option is already set, the specification (if any) for the previous setting of the option is copied in this setting. If the user types no specification, and the option is not already set, this setting has no specification. If the option is already set with a specification, and the user wishes this new setting to have no specification, he types ...

Notes:
The option command must not be used to set options named "on" or "off".
If any option is set more than once in an option command, the last setting will take precedence over all previous settings.

After it sets an option, the command prints a message to the user verifying what it has just done. If the brief option is on, the command omits verification. Verification allows the user to check that he has indeed set the options as he intended.

Note:

The specification for an option should be kept short. If a user wishes to give an option a long specification, he should write a data file instead and use the path name of the file as a specification. This will make it easier for him to change the specification for the option, and it expedites the work of the option commands. The maximum length for a specification is 512 characters. The maximum length for an option name is 64 characters.

Implementation

The command sets one option at a time. It notes whether the setting is to be "p" (permanent) or "s" (for the duration of the console session). If the setting is for the console session, the command modifies the options stack by calling

```plaintext
modopt (opt1, switch, spec, 2)
```

where switch = "1"b or "0"b. See BY.9.03 on modopt.

If the setting is to be permanent, the command calls

```plaintext
modopt (opt1, switch, spec, 1).
```

After setting the option, the command notifies the user of what it has done.

Delopt - usage

```plaintext
delopt (opt1 opt2 ... optn)
```

The command tries to delete each option. If delopt cannot find some option to delete it, the command prints a message:

```plaintext
opti not found
```
Note

If the same option name appears more than once in the argument list, the first appearance causes the option to be deleted, and all subsequent appearances cause the "optj not found" comment.

Implementation

Delopt deletes the options one at a time. To delete optj, delopt calls

    delete_opt (optj)

delete_opt deletes optj from the options stack (if optj was ever set). See BY.9.05 for a description of delete_opt.

Printopt - usage

printopt (n1 n2 ... nN)

prints the names of all options and their settings in frames n1, n2, ... nN of the options stack.

nj is a character (see below) or the number of a frame in the options stack. If n=0, printopt prints options from the current frame. Printopt understands some letters as input, in place of frame numbers:

p means permanent settings (frame 1)

s means session settings (frame 2)

c means current frame (n=0)

* means all frames (no numerical equivalent)

If the user specified "*" as an argument, printopt prints out all the settings for each option. Otherwise it prints out the settings in those frames specified by the user.
Sample format:

<table>
<thead>
<tr>
<th>option</th>
<th>frame</th>
<th>switch</th>
<th>specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>brief</td>
<td>1</td>
<td>on</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>off</td>
<td></td>
</tr>
<tr>
<td>sort</td>
<td>1</td>
<td>on</td>
<td>size</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>off</td>
<td></td>
</tr>
<tr>
<td>alpha.brief</td>
<td>1</td>
<td>unset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>on</td>
<td></td>
</tr>
</tbody>
</table>

Implementation

printopt calls

\[
\text{option\_frameno}(k)
\]

to obtain \( k \) = the number of the current stack frame, and prints

\[
\text{current frame is } k
\]

then prints the column headings "option", "frame", etc. (See BY.9.04 for a description of option\_frameno.) printopt scans the argument list, converting letter arguments into numbers. If it encounters "*" it ignores all other arguments. It eliminates all numerical arguments >\( k \). Nonsense arguments (punctuation characters, strings of letters, and the like) prompt an error comment:

\[
\text{glob incorrect argument}
\]

where \( \text{glob} \) is the offending argument. printopt continues with any valid arguments.

Next printopt calls

\[
\text{option\_names}(1, \text{name})
\]

to obtain the name of the first option in the options stack. (See BY.9.04 for a description of option\_names.) If the (revised) argument list contains only one argument, \( n \), printopt then calls

\[
\text{read\_opt}(\text{name}, n, \text{switch}, \text{spec}, \text{set})
\]
and prints $name$, $n$, switch ("on" or "off"), and $spec$ (the specification) if set = "1"b (the option is set in frame $n$.) If set = "0"b, printopt prints "unset" instead of the switch.

If the user specified more than one frame ($n_1$, $n_2$, etc.) the command calls

\[ \text{option\_values (name, sws, spec, n).} \]

(See BY.9.04 for a description of option\_values.) $sws$ is an array of switches, $specs$ an array of specifications. $sws(i)$ and $specs(i)$ represent the value of $name$ in frame $i$. $n$, returned by option\_values, is the number of the first frame in which $name$ is set.

Printopt prints $name$, then for each $n_j$ prints $n_j$, $sws(n_j)$, and $specs(n_j)$. For each $n_j \leq n$, printopt prints "unset" instead of the switch. All $n_j > k$ are ignored.

Each option is treated in the same manner until the values of all options have been printed. If there are $m$ options set, the call

\[ \text{option\_names (m + 1, name)} \]

should result in condition (options_401) being signalled. When this condition is signalled, printopt knows that all options have been printed, and returns to its caller.