Published: 05/26/67
(Supersedes: BY.11.03, 10/07/66)

Identification

Delerr - a subroutine to erase the last complete error description found in a user's error segment.
D. Widrig, K. J. Martin

Purpose

To provide the user a convenient method for deleting the last error description in the error segment, the subroutine delerr is used. It is expected that the user will invoke this procedure after examination of the last error description is complete.

Usage

```
call delerr;
```

Implementation

Delerr has a conceptually simple implementation. The relative pointer in error_out (err_ptr→error_out.recent) which points to the most recent error-description structure is reset to point to the previous one. (The error description structure is defined in BY.11.01.) The most recent error-description structure is freed and delerr returns. Any errors result in a call to seterr and the standard error signalling action.

To assist users in debugging programs, an option "no_error_delete", is recognized by delerr. If this option is on, delerr does not delete the last error description. Instead, delerr calls seterr to record the attempted deletion and sets the element error.attempted_delete in both error descriptions to "1"b to avoid later attempts to delete them. If error.attempted_delete = "1"b when delerr is called, delerr goes to the next most recent error description and checks error.attempted_delete in it. An error description structure in which error.attempted_delete = "1"b is considered to be logically deleted.

Even if the option no_error_delete is off, delerr checks error.attempted_delete of the most recent error description. If error.attempted_delete is "1"b it is inappropriate to delete this error description. Delerr traces back through the error-description structures (using the relative

pointer eptr->error.last_ptr) until it either finds one which has not been logically deleted or until the segment error_out has been exhausted. Assuming an error description is found which has not been logically deleted, delerr rethreads the relative pointers, eptr->error.last_ptr, to exclude that error description and then frees that error description (the space for it was allocated by seterr).

Note that if the option no_error_delete was never set, the most recent error description will be deleted on calls to delerr. In this case the actual implementation matches the conceptual implementation outlined in the first paragraph of implementation.

Upon return to command level, the user may elect to print the contents of his error segment using printerr (BY.11.04). With the above-mentioned option in effect, the user will have a complete record of all his errors and a record of which error descriptions were logically deleted. It is anticipated that this feature will improve debugging of long and convoluted programs.